Knowledge-Based Systems 280 (2023) 110992

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Video surveillance-based fall detection system using object-level feature thresholding and Z−numbers

Anima Pramanik [a,*], Sobhan Sarkar [b], Sankar K. Pal [a]

[a] Center for Soft Computing Research, Indian Statistical Institute Kolkata, 203 Barrackpore Trunk Road, Kolkata 700108, India
[b] Information Systems & Business Analytics, Indian Institute of Management Ranchi, Prabandhan Nagar, Nayasarai Rd, Ranchi, Jharkhand 835303, India

ARTICLE INFO

Keywords:
Fall detection system
Deep model
Rule base
Feature thresholding
Z-numbers

ABSTRACT

A new algorithm, namely Z−numbers-based deep feature thresholding (Z−DFT) is described for handling the issue concerning uncertainty that arises while classifying various fall and no-fall events in complex scenarios in both indoor (viz., home and hospital) and outdoor (viz., construction area and road). The Z−DFT consists of four phases: (i) object detection and tracking, (ii) feature extraction, (iii) feature thresholding and rule generation, and (iv) Z−numbers-based analysis for quantifying the reliability of the detected falls. Unlike state-of-the-art methods, Z−DFT uses both OpenPose and object-level features. This enables better modeling of both indoor and outdoor falls. New object-level features considered are change in area, aspect ratio, speed variation, and change in direction of the detected objects. The detection of fall locations involves two phases, viz., probable location and specific location. The probable location corresponding to each OpenPose and object-level feature is determined based on its statistical information. The commonality of these probable locations results in the specific location which is determined by framing linguistic rules using all the features. Z−numbers computed with features further reflect the reliability of detection. The characteristic features of Z−DFT are demonstrated over eighteen real-time videos acquired from YouTube8M and UR Fall data, along with its superiority claimed over ten state-of-the-art algorithms.

1. Introduction

The event, 'Fall' is considered as a dreadful anomaly, as it can affect a person physically as well as psychologically. Fall can lead to a different level of injury even death ultimately, if immediate help/support is not provided. According to the World Health Organization (WHO) [1], fall is the second leading cause of unintentional major injuries or deaths worldwide. Annually, an estimated 684,000 individuals globally succumb to their death due to fall incidents. While senior citizens constitute a large majority of this population; disabled people, children, and traffic accidents also account for a major portion of the casualties. The consequence of fall in busy road is most dreaded as compared to the other type of falls [2]. Hence, it is essential to develop a real-time fall detection system for detecting both indoor (i.e., home and hospital) and outdoor (i.e., construction area and road) falls to mitigate their impacts, thereby enhancing the human-safety.

The fundamentalism of video surveillance system in traffic stems from the fact that even minor interruptions to its flow, it causes widespread congestion. Traffic accidents may result in pedestrian fall which causes pedestrian-related life-altering injuries and even fatalities. Therefore, it is necessary to develop and install a real-time fall detection system at road-intersections to reduce the response time of medical dispatch units, potentially saving the pedestrians. The pre-causationary action after fall detection can prevent multi-vehicle collision as well as the further endangerment of the fallen person in order to save life. While, in this study, we have developed a real-time fall detection system that can automatically detect both indoor (i.e., home) and outdoor (i.e., construction area and road) falls in complex scenarios. Moreover, this system can abate the severity of fall-related injuries by ensuring prompt and timely medical assistance. This enhances the human-safety.

Over the years, a wide number of techniques have been developed to build an efficient fall detection system. However, researchers have largely focused on single-fall detection system designed for senior citizens residing in old-age homes [3,4]. Several number of studies have been focused on multi-falls in construction area [5]. Both are restricted to simple scenarios. This necessitates the development of a generic fall detection system which can be used for both single- and multi-falls in a variety of complex scenarios. The general fall detection system consists of five phases, such as (i) **Phase 1:** learning the characteristics of fall events, (ii) **Phase 2:** installation of the surveillance camera for

* Corresponding author.
E-mail addresses: apramanik17@gmail.com (A. Pramanik), sobhan.sarkar@gmail.com (S. Sarkar), sankar@isical.ac.in (S.K. Pal).

https://doi.org/10.1016/j.knosys.2023.110992
Received 25 January 2023; Received in revised form 4 September 2023; Accepted 11 September 2023
Available online 20 September 2023

capturing every minute detail of a scenario, (iii) **Phase 3:** development of a fall detection algorithm, (iv) **Phase 4:** training/validation, and (v) **Phase 5:** testing the developed fall detection algorithm over live stream videos by connecting the system (where the algorithm is installed) to the surveillance camera through an IP address. We have contributed to the aforesaid third phase (i.e., development of a fall detection algorithm) and fourth phase (i.e., training/validation). Conventional fall detection methods consist of three stages, including object detection and tracking, feature extraction, and feature analysis for classifying the event as "Fall" or "No-Fall". Conventional learning techniques used for object detection and tracking are categorized into supervised (object with class information) and unsupervised (object without class information). Unsupervised learning-based techniques do not require labeled data for training. But these techniques are restricted to the occlusion-free scenarios. On the other hand, deep CNN (for supervised learning) has remained the most popular choice due to its proficiency in pattern recognition, but it is restricted to the training classes. An unprecedented open-source library for person class is already developed [6]. Thus, having been trained on a suitable amount of person data, deep CNN is effective in person detection and tracking, and feature extraction.

Various deep learning-based techniques are focused on fall detection, but, these techniques are restricted to huge amount of fall incident related data, which may not be available or affordable always. Therefore, it is required to adopt unsupervised learning-based feature analysis (based on feature thresholding) followed by supervised object detection for modeling falls in real-time scenarios. Although a large number of research [7,8] have focused on fall detection, they may fail to handle uncertainty issues arising between various 'Fall' and 'No Fall' events under complex scenarios. Moreover, the algorithm designed for indoor fall detection may not be applicable for outdoor fall detection. This is due to the nature of fall is varied from video to video. It creates uncertainty in describing the characteristics of falls. The said uncertainty issue should be addressed in order to detect the falls in both indoor and outdoor scenarios and differentiate the fall event from the no-fall event. To address the issues, we have developed a new algorithm, namely $Z-$numbers-based deep feature thresholding ($Z-$DFT). The $Z-$DFT has four-fold tasks: (i) object detection and tracking, (ii) feature extraction, (iii) feature thresholding and rule generation, and (iv) Z-numbers-based quantification for reliability analysis of detection [9]. For object detection, deep CNN-based detectors are broadly classified as one-stage detectors [10] and two-stage detectors [11]. One-stage detectors (e.g., YOLO and its variants) are good in having inference speed but achieve less detection accuracy as compared to two-stage detectors (e.g., RCNN and its variants) [12]. A two-stage detector, namely Granulated RCNN (G-RCNN) [11] is used in this study for object detection task. After detection, bounding boxes are fitted over the detected persons. The bounding-box information is further used for the tracking task.

After detection and tracking task, deep features are extracted from each detected person. Unlike state-of-the-art methods, in $Z-$DFT, both OpenPose and object-level features are considered. This enables an improved modeling of both indoor and outdoor falls. The spatial information for each detected person is fed to a deep model, namely OpenPose model [13] (consisting of multi-stage CNN) for extracting OpenPose features that estimate human pose. Posture features are very effective in modeling fall events where the body parts of each detected person are clearly visible in videos. People are clearly visible in videos representing the areas, including home, hospital, and construction area. On the other hand, in road, due to complex traffic, a person may be partially occluded by another person or vehicle. In this case, all body parts of that person may not be clearly visible in videos. Therefore, object-level features are considered along with OpenPose features to improve the modeling of falls. Various object-level features, such as change in area, aspect ratio, speed variation, and change in direction are considered in this study.

Both OpenPose and object-level features are analyzed for defining thresholds in order to detect the approximate/probable locations of falls. This enables an automated updation of thresholds based on the information of video contents, thereby holding the generalization capability. Using these features and their thresholds, two linguistic rules are defined for combining the probable locations in order to find the specific locations of falls. Further, to ensure the detection, Z-numbers are computed using the feature thresholding score which is obtained from all OpenPose and object-level features. Rule generation followed by fuzzy layer neural network-based classification can be effective in handling the uncertainty issue [14]. A recognition system combining the fuzzy set theory and approximate reason is effective in handling the imprecise patterns of data [15,16]. Z-numbers are the advanced version of these methods. A recent study [17] reveals how the concept of Z-numbers can be used in video processing to explain a scene with certainty in the form of natural language. Therefore, to obtain the reliability of detected falls, we have quantified the performance of rule-based system using Z-numbers. The generated rules combined with Z-numbers-based quantification handle the uncertainty arising between various 'Fall' and 'No Fall' events under complex scenarios.

The developed $Z-$DFT algorithm is embedded within the video surveillance system for fall detection, thereby making a system, called fall detection system. Effectiveness of the developed $Z-$DFT algorithm for fall detection has been demonstrated extensively over eighteen videos containing both 'Fall' and 'No Fall' events in indoor (i.e., home) and outdoor (i.e., construction area and road). A comparative study is done with some state-of-the-art methods to show the superiority of the developed $Z-$DFT algorithm. It is also proved that our developed $Z-$DFT algorithm is able to overcome other issues, including generalization capability in applying to multiple situations with complex scenarios.

Based on the aforesaid discussions, the novelties of our study can be summarized as follows: A new algorithm, namely $Z-$numbers-based deep feature thresholding ($Z-$DFT) is developed for handling the uncertainty issue that arises during the classification of various 'Fall' and 'No-fall' events under complex scenarios. Unlike state-of-the-art methods, in $Z-$DFT, both OpenPose and object-level features are used for enhancing the modeling of both indoor (viz., home and hospital) and outdoor (viz., construction area and road) falls. In $Z-$DFT, some features are derived from both OpenPose and object-level features. The threshold for each OpenPose and object-level feature is determined using its statistical information (i.e., mean and standard deviation) and is used to obtain the approximate/probable locations of falls. Two linguistic rules are defined using all the features and their thresholds for obtaining the specific locations of falls. Z-numbers are computed based on the feature thresholding scores to obtain the degree of reliability for fall detection.

The rest of the paper is organized as follows: Section 2 presents the related works. A conceptual framework of general fall detection system is described in Section 3. Section 4 presents the developed $Z-$DFT algorithm. In Section 5 the results are discussed. Finally, we conclude this study and state some future scopes in Section 6.

2. Related works

In general, the fall detection techniques have three phases: (i) object detection and tracking, (ii) feature extraction, and (iii) feature analysis. A brief review on object detection and tracking is presented in Section 2.1. The techniques used for feature extraction and feature analysis are majorly classified into two categories: (i) machine vision and (ii) feature-based analysis. Related studies on machine vision and feature-based analysis for fall detection are discussed in Sections 2.2 and 2.3, respectively.

2.1. Object detection and tracking

Some key literature on object detection and tracking are presented in the following sections.

2.1.1. Object detection

Literature on object detection are categorized into two classes, such as image processing [18] and deep learning [11,19]. In the former category, object detection is done by following two steps: (i) generation of the foreground region proposal and (ii) classification of objects belonging to the said region proposal. Background subtraction is done in [20] for obtaining the foreground regions using either temporal difference or symmetry (local) of an image patch. In [21], Histogram of gradients (HoGs) of convolution features is used for extracting the object regions. Various features, such as edge, spatial/color/temporal similarity, and Haar-like features are well in object detection [22]. The Gaussian Mixture Model (GMM) is one of the powerful background modeling algorithms [23]. It helps in pixel modeling with the help of weighted mixture of Gaussian. On the other hand, granulation (i.e., clustering) is the strong paradigm for object detection by handling the uncertainty issue arising between various object and no-object regions. Aforesaid image processing-based techniques are useful for region proposal but not for object classification.

With the presence of rich training data, deep network can provides promising results for person detection [12]. Deep network-based object detectors are categorized into two class, such as two-stage detectors and one-stage detectors [12]. Two-stage detectors include Faster RCNN [19] and its variants [12], whereas, one-stage detectors involve YOLO and its variants [24]. Although one-stage detectors are speeder, two-stage detectors achieve high accuracy. In this study, we have used two-stage detector for the detection task, as fall detection task is highly sensitive to the detection accuracy. In earlier study, we have developed a two-stage detector, namely granulated region convolutional neural network (G-RCNN) [11], which is an advanced version of Faster RCNN. A trade of between accuracy and speed is maintained in G-RCNN. Therefore, in this study, G-RCNN (a two stage-detector) is used for person detection.

2.1.2. Object tracking

Object tracking task is followed by the detection task. In object tracking, either pixel-level features or object-level features are used for data association. Region-based tracking [25,26] is unsupervised, where image parts are connected based on color, intensity, and texture-statistics. Contour-based [27,28] tracking is an unsupervised approach, where the bounding boxes corresponding to detected objects are updated dynamically at each time increment. These two approaches may not be applicable to occluded scenarios. Among various unsupervised approaches for tracking, model-based approach [29,30] is also popular. This produces good tracking accuracy but the major limitation is the requirement for an accurate geometric model of object. The issue of obtaining accurate geometric object model is solved by the deep network-based detection techniques. Therefore, model-based approach may show good tracking accuracy after getting the accurate detected regions. Thus, we have adopted model-based approach in this study for object tracking.

2.2. Machine vision for fall detection

Due to the significant improvement in machine learning and deep learning, machine vision-based techniques become predominant in fall detection [11]. A few related studies on machine learning- and deep learning-based fall detection are discussed in Sections 2.2.1 and 2.2.2, respectively.

2.2.1. Machine learning-based techniques

Popular machine learning techniques for fall detection are Support Vector Machine (SVM) [31,32], Decision Tree (DT) [33,34], Naive Bayes (NB) [35] and K-Nearest Neighbors (KNN) [36]. A fall detection instrument, namely 'iTUG' is developed for fall detection using logistic regression [37] based on the data collected from 69 participants (26 falls and 43 non-falls). In [38], a gradient boosting DT classifier is used to detect falls using human posture features. The method also has an inbuilt alarm to notify authorities about the fall event. In [39], multi-layer perceptron and random forest are trained using both the spatial and temporal information corresponding to pose features. All these methods are restricted to a huge amount of training data which may not be available always.

2.2.2. Deep learning-based techniques

Recently, deep learning-based methods gain much attention in the domain of machine vision due to their ability of self learning based on useful features [40]. Various studies [5,41–43] on deep learning are focused on indoor and outdoor falls. In [41], two deep networks, DCNN and LSTM are combined for fall detection. DCNN is used for extracting object location and feature extraction. Whereas, LSTM is used to classify the fall event by aggregating the temporal information corresponding to the extracted features. Another deep learning-based technique [43] combines gated recurrent units (GRUs) with CNN for fall detection. LSTM-prediction followed by OpenPose feature extraction is used in [5] for fall detection. A combination of two fold tasks, including (i) context-aware method for data acquisition using the infrared depth array sensors and (ii) deep learning (3D-CNN) for 'Fall' or 'No Fall' classification, was able to achieve high accuracy but mis-classifies alternative heat sources as persons, leading to inaccurate results [44].

Some other deep learning-based methods, including CNN + LSTMs and RNN + LSTM are well-known for fall detection [7]. Auto-encoders have also been gaining some attention in recent years due to the efficient manner in which they are able to learn data coding for unsupervised learning-based models [45]. CNN-based model is used in [46] for detecting a sequence of frames containing fall incident. Recurrent Neural Networks (RNNs) with underlying LSTM blocks is used in [47] for fall detection. A novel framework for fall detection, known as DeepFall is developed in [48]. DeeFall uses spatio-temporal convolutional auto-encoder for training using the spatial and temporal features. Non-invasive sensing method is used here for extracting the features from videos which contain traffic activities. In [42], LSTM/GRU model is trained using OpenPose skeleton for fall detection. However, aforesaid studies are focused on indoor falls designed for citizens residing in homes and(or) outdoor falls designed for persons working in construction area.

The uncertainty arising between various 'Fall' and 'No Fall' events is not addressed in both machine learning-based and deep learning-based techniques. In machine vision techniques, huge amount of training data is required which may not be available always. They also lack in generalization in practical applications as they are generally trained on simulated data. Moreover, for training of deep network, GPU having high capacity is required which may not be affordable always. The issues of 'unavailability of data' and 'GPU requirement' can be addressed in feature-based analysis. A detailed review on feature-based analysis for fall detection is presented in next section.

2.3. Feature-based analysis for fall detection

As feature-based analysis follow unsupervised learning, it is useful when there is an unavailability of training data. In feature-based analysis, features are computed using a set of parameters and evaluated with a set of rules (based on some thresholds) to model a event as 'Fall' or 'No Fall'. Therefore, features as well as thresholds are chosen carefully to enhance the efficacy of the feature-based model [49]. As feature-based methods are training independent, these methods are able to achieve results at a relatively faster pace than machine vision-based methods. Two-kinds of features, namely pose features and object-level features are very effective in modeling falls. A few related studies on pose features and object-level features based analysis for fall detection are discussed in Sections 2.3.1 and 2.3.2, respectively.

2.3.1. Pose features-based analysis

Several researches are done on pose estimation models for extracting the region containing human and corresponding pixel-related features [8]. In [50], a method is developed using a tri-axial accelerometer and gyroscope for capturing data and pose features-based analysis for fall detection. While the model is able to effectively detect falls, it has been trained on simulated data and thus may not be appropriate for real-life complex scenarios. A single-person fall detection method is demonstrated in [51] using vision-based capturing device and OpenPose feature thresholding-based classification. However, the model faces certain limitations when it encounters insufficient difference in color between the subject's clothes and the background. In [52], key points of human skeleton are extracted using OpenPose model and are used for formulating rules for fall detection. In [53], the MobileNetV2 network is trained with OpenPose features for fall detection. As 'Fall' is directly related to the characteristics of human pose, pose features are more representative than object-level features. But in case of complex scenarios, human pose may not be clearly visible in videos. In such cases, object-level features are effective in modeling falls. Some key literature on object-level feature-based analysis for fall detection are presented in the next section.

2.3.2. Object-level features-based analysis

Object-level features are very effective in modeling fall events under complex scenarios. A concept called 'Adaptive Directional Bounding Boxes' is used in [54] for obtaining the bounding box over the detected object. Features, such as aspect ratio and center-of-gravity corresponding to this bounding box are used for fall detection. Torso angle and centroid height are also effective in detecting the fall event [55]. Ratio of the observed silhouette area can also be helpful in fall detection [56]. Feature threshold based on the stationary time spent by the center-of-gravity of the silhouette is used [57]. In [58], human bounding volume is used for modeling four type of falls. A multi branch deep network is developed in [59] using context, object features, and detailed semantic part cues for fall detection.

Based on the aforesaid discussion, the research issues are summarized as follows.

(i) Most of the studies are focused on both single- and multi-falls in either indoor (i.e., home and hospital) or outdoor (i.e., construction area) having occlusion-free scenarios.
(ii) Very few studies have been concentrated on complex scenarios, but these are restricted to construction area.
(iii) Either OpenPose or object-level features are used for modeling falls.
(iv) No research on handling the uncertainty issue arising between various 'Fall' and 'No Fall' events under complex scenarios.
(v) Usage of simulated data restricts the model's usage in outdoor scenarios and real-life applications. Moreover, there is a significant shortage of training data in this field.

In order to address the aforementioned issues, the present study contributes as follows:

(i) A new fall detection algorithm, namely Z-numbers-based feature thresholding (Z−DFT) is developed by incorporating both OpenPose and object-level features along with rule generation and Z-numbers-based reliability analysis.
(ii) Unlike state-of-the-art methods, in Z−DFT, both OpenPose and object-level features are used for better modeling of both indoor and outdoor falls. Some key features, such as change in area, speed variation, and change in direction are derived using the knowledge of object-level features.
(iii) The detection of fall locations involves two phases, such as probable location and specific location. Thresholding criteria for each OpenPose and object-level feature is defined in order to obtain the probable/approximate locations of falls. Whereas, two linguisting rules are defined using all OpenPose and object-level features in order to obtain the specific locations of falls.
(iv) Z−numbers are computed using the feature thresholding score in order to obtain the reliability of detected falls.
(v) The defined linguistic rules along with the Z−numbers-based quantification handle the uncertainty issue arising between various 'Fall' (i.e., indoor and outdoor falls) and 'No fall' events under multiple situations with complex scenarios.

A detailed methodology of Z−DFT is presented in Section 4. The conceptual framework of general fall detection system is presented in the next section.

3. The conceptual framework of general fall detection system

The conceptual framework of general fall detection system is presented in Fig. 1. This framework has five phases: (i) gathering knowledge about the characteristics of fall event, (ii) installation of CCTV, (iii) development of fall detection algorithm, (iv) training/validation, and (v) implementation, explained as follows.

3.1. Characteristics of fall event

At first, it is imperative to comprehend the types of fall events occurred in outdoor (e.g., road) and indoor (e.g., home) areas. Therefore, a multi-disciplinary team (consisting of both safety professionals and technical experts) is formulated. Safety professionals have detailed knowledge about the characteristics of fall events; whereas, domain experts have knowledge about how to model the characteristics of such events. Safety professionals should know the orientation of the installed camera and characteristics of the fall event (which is to be modeled by technical experts). Based on the aforesaid two information, technical experts can develop algorithm for modeling the fall event. As said earlier, real-time fall detection is required in order to send immediate help to the fall location for mitigating the impact of fall. Therefore, initially, as per the knowledge of safety professionals, technical experts should gather knowledge about various type of falls and develop algorithm for modeling both indoor and outdoor falls in complex scenarios.

3.2. Installation of CCTV

Before start the analysis, proper places are identified for installing the CCTV to monitor the traffic scenarios. CCTV is installed at a slight elevation in the road side to capture every minute details. The number of cameras to be installed, is decided by the safety professionals based on the requirement. Here, cameras are installed to capture falls. The videos of falls are further used for validating the fall detection algorithm.

3.3. Development of fall detection algorithm

In this phase, the characteristics of falls is modeled by technical experts using the domain knowledge. The developed fall detection algorithm should have generalization capability, so that it can be applicable in both indoor and outdoor areas. Keeping this in mind, we have developed a semi-supervised algorithm, namely Z−DFT for fall detection, as explained in Section 4. The Z−DFT works on unsupervised rule generation followed by feature extraction, and supervised detection and tracking.

3.4. Training/validation

Next phase is the training/validating of the developed fall detection algorithm. As already defined, the developed Z−DFT algorithm is semi-supervised in nature. In Z−DFT, object detection is done in the supervised way, whereas, feature analysis is done in the unsupervised way. Therefore, no training is required for Z−DFT, Only validation of Z−DFT is done using some videos acquired from UR Fall data.

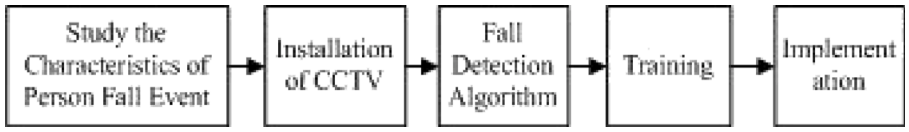


Fig. 1. Conceptual framework of general fall detection system for improving human-safety.

3.5. Implementation

This operation consists of two important steps: (i) testing the developed algorithm ($Z-$DFT) over real-time videos and (ii) adopting remedial actions in the post-detection of the fall event. The validated $Z-$DFT algorithm is tested over eighteen videos acquired from YouTube8M, and UR Fall data. After achieving the convincing testing accuracy, the developed algorithm can be tested on live stream-videos (24×7) using an IP address of the camera. After the detection of a fall, a document containing the information of location-stamp, time-stamp, and a video clip is generated. Moreover, an alarm is generated in the control room to make safety personals vigilant, so that they can provide an immediate help to the fall-location for mitigating its impact. The generated video clips contain pre-scenario and scenario of the fall event, thereby enhancing the capability for recognizing the reason of fall. The video clips can be used in future to aware people about the behavior related causes of fall. The training of persons is done to rectify their faults, so that the number of fall, related to the abnormal behavior of person is reduced. Moreover, after analyzing the reasons, correct preventive measures (related to road, vehicle, floor, and(or) environment conditions) should be taken by concerned authority to stop re-occurring the cause (related to road, vehicle, floor, and(or) environment) of fall. In this way, human-safety can be enhanced.

From the aforesaid discussion, it is evident that the effectiveness of fall detection system depends on the accurate detection of falls using a fall detection algorithm. The developed $Z-$DFT algorithm for fall detection is explained in the next section.

4. $Z-$DFT

An overview of the developed $Z-$DFT algorithm is exhibited in Fig. 2. Initially, frames are extracted from the input video. Object (i.e., person) detection and tracking is then performed over the video frames, as explained in Section 4.1. Thereafter, both OpenPose and object-level features are extracted from the detected and tracked objects, as described in Section 4.2. The extracted features are analyzed for generating rules in order to obtain the specific locations of falls. This is explained in Section 4.3. To further obtain the reliability of detections in terms of linguistic descriptions, Z-numbers are computed based on these features, as presented in Section 4.4.

4.1. Person detection and tracking

Frames that are extracted from video, are fed to the detector for person detection. After detection, person tracking is done over the consecutive frames. Methods used for person detection and tracking are stated in Sections 4.1.1 and 4.1.2, respectively.

4.1.1. Person detection

A two stage-detector, namely G-RCNN [11] is used for person detection. G-RCNN incorporates the concept of granulation (i.e., clustering) within the deep network for object detection task. In G-RCNN, both spatio-color and temporal information corresponding to the first pooling layer (of the deep CNN used in G-RCNN) are used for obtaining the spatio-temporal granules which represent the approximate object(s) location. Spatio-temporal granules are the common regions between spatio-color and temporal granules. Here, satio-color granules represent both static and foreground objects. Whereas, temporal granules define the foreground regions. Due to the noise, some undesired regions (corresponding to the background) might belong to the temporal granules, thus resulting in vagueness in the detection of foregrounds. Therefore, the commonality between temporal and spatio-color granules indicates the regions that correctly represent the foregrounds.

The deep network, namely granulated AlexNet (G-AlexNet) [11] is developed and used as a feature generator in G-RCNN. G-AlexNet has five convolution layers (namely Conv1, Conv2, Conv3, Conv4, and Conv5), three pooling layers (namely Pool1, Pool2, and Pool5), three granulation layers (namely, Granule1a, Granule1b, and Granule2), anchor layer, and three fully connected layers (namely, FC_1, FC_2, and FC_3). The G-AlexNet takes images/video frames as input and generates the reduced feature maps with several channels (number of channels is equal to the number of filters used in each layer) after the operation at each convolution layer. At each channel of Pool1 map (feature map generated after the average pooling operation at Pool1 layer), features corresponding to both spatio-color and temporal information are extracted and used in the Granule1a and Granule1b layers, respectively. In these two layers, both spatio-color and temporal granules are formed using quad tree decomposition and three point estimation, respectively. Then, the commonality between these two granules is considered as spatio-temporal granules formed over Granule2 layer. The spatio-temporal granules represent the approximate objects regions and are passed to the next layer (i.e., Conv2 layer) for extracting deep features corresponding to the object(s) region. This reduces the searching space, and increases the detection speed. The Pool5 map is the final reduced feature map obtained from G-AlexNet architecture. In G-RCNN, biases are used for each operation corresponding to each layer. Therefore, Pool5 map may contain some irrelevant features. To remove these irrelevant features, the pixel location (in the input image) corresponding to both Granule2 and Pool5 layers are considered as regions of interest (RoIs). The features corresponding ot these RoIs are used for classification task.

For the classification task, masks of different scales and aspect ratios slide over the RoI-map. These masks are called anchors. Nine anchors of sized (10×14, 14×10, 14×14, 28×20, 20×28, 28×28, 56×40, 40×56, 56×56) are used in this study. The two-dimensional pixel array for representing each anchor is called anchor feature map. Then, the anchor maps are resized with the input image (i.e., the size of input which is used to train the classifier used for the detection task). The sequence of anchor maps are fed to the classifier (SVM) through the three fully connected layers (FC_1, FC_2, and FC_3) for the task of person classification and bounding-box fitting. Here, ReLU [11] activation function is used. In G-RCNN, two type of losses, namely classification loss and bounding-box regression loss are combined to obtain the total loss. This loss is back propagated from the Pool5 to Conv1 layers for parameter tuning for obtaining better accuracy. Learning rate of 0.01 is used. Weights and biases of each layer are initialized with 0.001 and 0.0001, respectively. Gradient search algorithm is used here for training. After the detection, detected persons are tracked, as explained in the following section.

4.1.2. Person tracking

The tracking task is followed by the detection task. In this study, feature-based tracking is done for the detected persons. As said earlier, person detection is done over each frame using the G-RCNN. The location and class information corresponding to all detected persons are used for tracking. The process of tracking consists of two steps: (i)

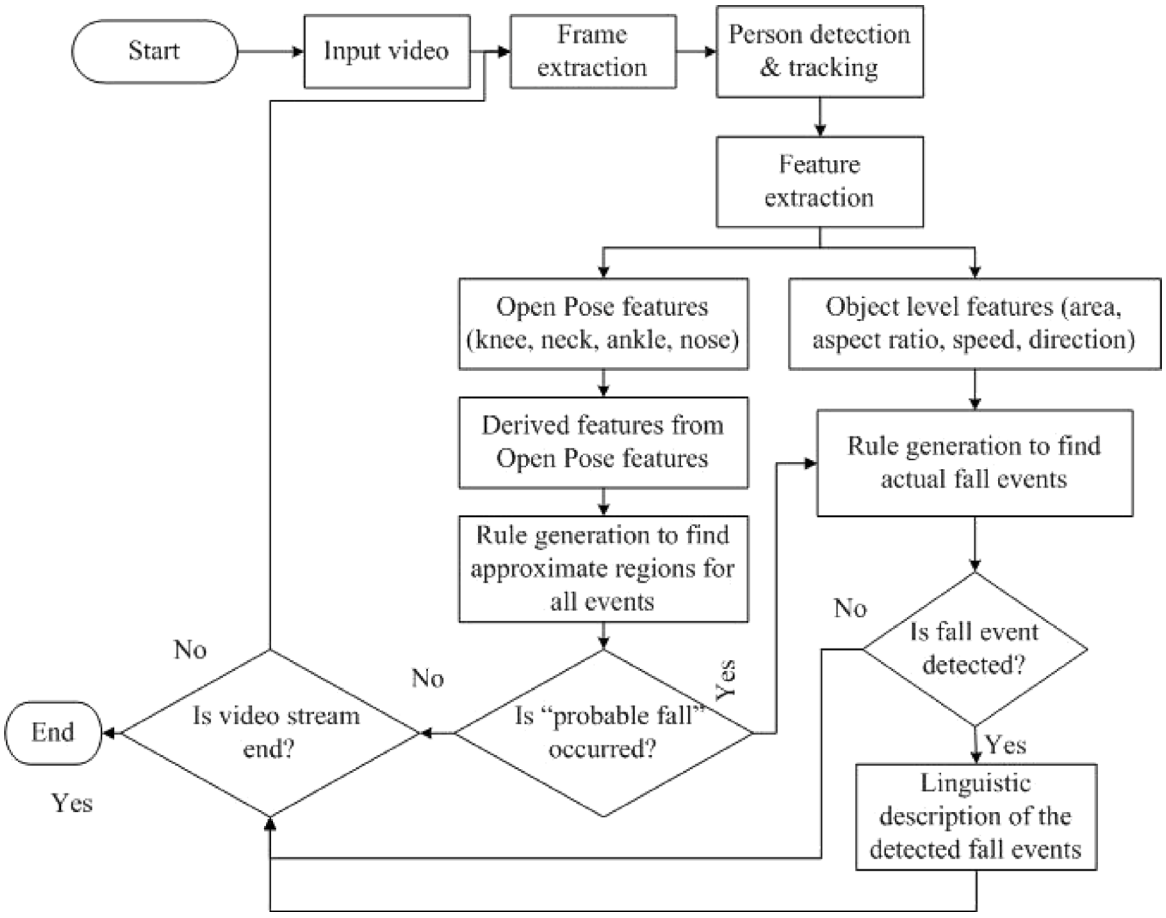


Fig. 2. Flowchart of $Z-$DFT algorithm.

the estimation of state-space which already obtained from the detection task, and (ii) the association of estimated state-space from frame to frame. After detection, each detected person is fitted by appropriate bounding-box. The information of eight-dimensional state space, such as co-ordinate of left corner, height, width, aspect ratio, and velocity of the detected bounding-box are used as features for tracking. Using the said information, data association is performed to generate the trajectories corresponding to the detected persons.

Association of the spatio-temporal information for a detected person over the consecutive frames is done to obtain the trajectory corresponding to the same person. The spatial information of a detected person in a frame is called the track-let of its trajectory. Let O_i, M, and T_j be the ith detected object in f_t, total number of track-lets present in f_{t-1}, and jth track-let present in f_{t-1}, respectively. Then, O_i is associated with a track-let (lying nearest in the previous frame) using Eq. (1).

$$O_i \leftarrow T_j \; if \; dis(O_i, T_j) = \wedge\left\{dis(O_i, T_1), dis(O_i, T_2), \ldots, dis(O_i, T_j), \ldots, dis(O_i, T_M)\right\} \quad (1)$$

where $dis(O_i, T_j)$ defines the distance between O_i and T_j, and $\wedge$ represents the minimum operation. In this way, data association is done for person tracking.

After person detection and tracking, both OpenPose and object-level features are extracted. A detailed description of each feature is presented in the next section.

4.2. Feature extraction

Two type of features/key-points, such as OpenPose and object-level features (i.e., change in area, aspect ratio, speed variation, and change in direction) are extracted for each detected person. Well known OpenPose model [13] is the open source model and can be used for multi-person $2D$ pose estimation. This model was developed with the aim of analyzing the real-time movements of multiple persons. Therefore, in this suty, using OpenPose model, a set of flow fields (i.e., OpenPose features) which encodes the degree of association between body parts of detected person is extracted. A total of 18 OpenPose features can be extracted for a person. But the number of OpenPose features generated for each detected person is varied with respect to the effect of visual quality (i.e., low illumination), occlusion, and training data. Most popular 'Common Objects in Context (COCO)' dataset provides the information of highest number of different joints. Therefore, the OpenPose model is trained on popular MSCOCO Dataset [60]. We have used the CPU version of OpenPose in an effort to demonstrate a more cost-effective and energy-efficient fall detection system, as GPU is not affordable always.

The $2D$ locations of 18 OpenPose features, including 'L-Eye', 'R-Eye', 'L-Ear', 'R-Ear', 'Nose', 'Neck', 'L-Shoulder', 'R-Shoulder', 'L-Elbow', 'R-Elbow', 'L-Wrist', 'R-Wrist', 'L-Hip', 'R-Hip', 'L-Knee', 'R-Knee', 'L-Ankle', and 'R-Ankle' are extracted for each detected person. A window of size $w \times h$ is slided over the detected person, and is fed to the OpenPose model as an input. The OpenPose is a multi-stage CNN which generates $2D$ locations of all OpenPose features along with their confidence scores for each input of size $w \times h$. The deep CNN, namely VGG16 is used in the OpenPose model. The VGG16 consists of thirteen convolutional, three pooling, and three fully connected layers. At the output of 10th convolutional layer of VGG16, feature maps corresponding to the aforesaid 18 OpenPose features are generated. Then, by the next layer, part affinity fields (PAFs) are created for each of these 18 key features, and considered as feature vectors. In the subsequent layers of VGG16, these feature vectors are concatenated corresponding to each OpenPose feature.

As said earlier, in order to obtain better modeling of falls, both OpenPose and object-level features are used. Object-level features that are considered in this study are change in area, aspect ratio, speed variation, and change in direction corresponding to each detected person. Aspect ratio is the ratio between width and height. Let $d_i^t(A)$, $d_i^t(AR)$, $d_i^t(SP)$, and $d_i^t(DIR)$ be the area, aspect ratio, speed, and direction corresponding to the ith detected person in the current (tth) frame, respectively. Let (x_i^t, y_i^t) be the x-coordinate and y-coordinate of the left corner for the bounding box fitted over the detected ith person in f_t, and (w_i^t, h_i^t) be the width and height of the same detected person. Then, $d_i^t(A)$, $d_i^t(AR)$, $d_i^t(SP)$, and $d_i^t(DIR)$ are defined as:

$$\begin{aligned} d_i^t(A) &= |(w_i^t \times h_i^t)| \\ d_i^t(AR) &= |\frac{w_i^t}{h_i^t}| \\ d_i^t(S) &= \sqrt{(Sx_i^t)^2 + (Sy_i^t)^2} \\ d_i^t(DIR) &= |\tanh(\frac{y_i^t}{x_i^t}) - \tanh(\frac{y_i^{t-1}}{x_i^{t-1}})| \end{aligned} \tag{2}$$

where w_i^t and h_i^t represent the width and height of ith person detected in tth frame (f_t). Sx_i^t and Sy_i^t represent the velocity components for ith person detected in f_t corresponding to the horizontal axis and vertical axis, respectively. Various object-level features, such as change in area, speed variation, and change in direction are computed using the spatio-temporal information of area, speed, and direction, respectively. This is explained in Section 4.3.2.

Both OpenPose and object-level features are analyzed for obtaining the specific locations of falls. This is explained in the next section.

4.3. Feature analysis and fall classification

The method of feature extraction is illustrated in the previous section. The next step is the feature analysis and event classification. The detection of fall location involved to phases: (i) detection of probable locations of falls and (ii) detection of specific locations of falls. In the first phase, each feature is analyzed for obtaining its suitable threshold. Based on the thresholds, probable locations of falls corresponding to both OpenPose and object-level features are determined. Thereafter, the commonality between the probable locations is obtained by defining two rules using the information of features and their thresholds. The said commonality represents the specific location of fall. Analysis of OpenPose and object-level features for fall detection are presented in Sections 4.3.1 and 4.3.2, respectively.

4.3.1. Analyses of OpenPose features

The primary OpenPose features for fall detection training have been fixed as ankle and nose. This is because, for a person, these features (i.e., ankle and nose) are easily detected due to the prominence and easy identification of face and feet. Using the OpenPose model, these features can be captured from any angle of positioning of person, i.e., the approximate position of the nose is detected even from the back of the person's head. However, in the event, where any one of these features is occluded, then other two features, namely knee and neck are taken into consideration. This can act as a safeguard in several cases including fall events in which the person's feet or face get occluded, or failure in detection of features due to unprecedented errors by the OpenPose model.

Out of the eighteen OpenPose features, 6 features, namely 'Nose', 'Neck', 'L-Knee', 'R-Knee', 'L-Ankle', and 'R-Ankle' are most relevant for representing the change in nature of the scene during the fall event. Therefore, these six OpenPose features are used for analysis purposes. The $2D$ (x, y) co-ordinates of the aforesaid six OpenPose features are separated from other OpenPose features based on the confidence scores. Let $y_i^t(Lankle)$, $y_i^t(Rankle)$, $y_i^t(nose)$, $y_i^t(Lknee)$, $y_i^t(Rknee)$ and $y_i^t(neck)$ be the location of y-coordinates corresponding to the 'L-Ankle', 'R-Ankle' 'Nose', 'L-Knee', 'R-Knee', and 'Neck' for detected ith person present in f_t, respectively. The top left position of the detected person is represented by $(0, 0)$ co-ordinate. Let $d_i^t(nlk)$, $d_i^t(nrk)$, $d_i^t(nla)$, and $d_i^t(nra)$ be the differences between neck and left knee, neck and right knee, nose and left ankle, and nose and right ankle, respectively, as defined:

$$\begin{aligned} d_i^t(nlk) &= |y_i^t(Lknee) - y_i^t(neck)| \\ d_i^t(nrk) &= |y_i^t(Rknee) - y_i^t(neck)| \\ d_i^t(nla) &= |y_i^t(Lankle) - y_i^t(nose)| \\ d_i^t(nra) &= |y_i^t(Rankle) - y_i^t(nose)| \end{aligned} \tag{3}$$

Let $d_i^t(nk)$ and $d_i^t(na)$ be the average distances between neck and knee, and nose and ankle, respectively. The $d_i^t(nk)$ and $d_i^t(na)$ are defined as

$$\begin{aligned} d_i^t(nk) &= \frac{d_i^t(nlk) + d_i^t(nrk)}{2} \\ d_i^t(na) &= \frac{d_i^t(nla) + d_i^t(nra)}{2} \end{aligned} \tag{4}$$

Two thresholds, namely Th_1 and Th_2 are determined for $d_i^t(nk)$ and $d_i^t(na)$ in order to obtain the probable fall location using the knowledge of four OpenPose features (i.e., neck, knee, nose, and ankle). These two thresholds are defined as:

$$\begin{aligned} Th_1 &= th_{nk} \left(= mean\left(d_i(nk)\right) - 3 \times \sigma_{nk}\right) \\ Th_2 &= th_{na} \left(= mean\left(d_i(na)\right) - 3 \times \sigma_{na}\right) \end{aligned} \tag{5}$$

where $mean\left(d_i(nk)\right)$ and σ_{nk} represent the mean and standard deviation of the position difference between neck and knee for ith object (i.e., person) over the P number of consecutive frames. Similarly, $mean\left(d_i(na)\right)$ and σ_{na} represent the mean and standard deviation of the position difference between nose and ankle for ith object (i.e., person) over the P number of consecutive frames. The base of thresholds selection is that the average distance between features should be minimum during the fall event. Therefore, for a detected person, if any one of the said OpenPose features is less than its threshold, then probable fall is found at the location of this person, and the location is called 'probable location' of falls.

A rule is defined by combining these two features (i.e., $d_i^t(nk)$ and $d_i^t(na)$) in order to get more probable fall location. The 'Rule 1' is defined as:

$$\begin{aligned} if\ d_i^t(nk) \le Th_1\ or\ d_i^t(na) \le Th_2\ , then\ \}Fall' \\ otherwise,\ \}No\ Fall' \end{aligned} \tag{6}$$

From the above discussion, it is evident that, using the defined rule, the more probable locations of falls are found. The searching space for obtaining the specific location of falls is reduced, thereby increasing the detection speed. Moreover, two thresholds, Th_1 and Th_2 are selected based on the statistical information of the OpenPose features. Therefore, these two thresholds will be updated automatically with the contents present in the consecutive frames. Thus it holds the generalization ability in handling both simple and complex scenarios.

The rule that is defined using OpenPose features is useful in detecting the fall event in indoor scenario (home) and construction area. But in case of road, mostly objects are occluded with each other. In such complex scenarios, OpenPose features may not be effective always in detecting the fall events. The reason is: the characteristics of fall in indoor scenario (belongs to simple traffic) is different than the characteristics of fall event in outdoor scenario (belongs to complex traffic). It creates uncertainty in various fall events. As traffic flow is also distracted during the anomalous event (fall), object-level features are effective in modeling the fall-event. The analysis of object-level features for fall detection is presented in the next section.

4.3.2. Analyses of object-level features

The analysis of object-level features is added with the analysis of OpenPose features in order to determine the specific location of falls. Therefore, to strengthen the decision defined in Eq. (6), object level features, namely change in area, aspect ratio, speed variation and change in direction of the detected persons are considered. During the fall event corresponding to a frame, change in area and aspect ratio of a detected person should be higher than the change in area and aspect ratio of the same person detected in the previous frame. Both change in area and aspect ratio are used to differentiate the sitting position from the standing pose. When fall occurs in a particular frame, the change in feature is entirely different from the remaining frames. In the other way, we can say that, during the fall event, their is an abrupt change found in object-level features. For complex scenarios having multiple persons, fall detection algorithm checks if the speed and direction corresponding to other detected persons are abruptly changed. The reason is: during fall, mostly, other persons move to the fall location to help the fallen person. Therefore, two features, such as speed variation and change in director for other detected persons are also analysis to check whether fall is occurred or not.

Let $dif_i^t(A)$ be the change in area of ith detected person for two consecutive frames (i.e., f_t and f_{t-1}). Let $dif_j^t(S)$ and $dif_j^t(DIR)$ be the speed variation and change in direction for jth foreground object for two consecutive frames (i.e., f_t and f_{t-1}). Here, $dif_i^t(A)$, $dif_j^t(S)$, and $dif_j^t(DIR)$ are defined as:

$$\begin{aligned} dif_i^t(A) &= |d_i^t(A) - d_i^{t-1}(A)| \\ dif_j^t(S) &= |d_j^t(S) - d_j^{t-1}(S)| \\ dif_j^t(DIR) &= |d_j^t(DIR) - d_j^{t-1}(DIR)| \end{aligned} \tag{7}$$

Let $dif_j^t(SDIR)$ be the product of $dif_j^t(DIR)$ and $dif_j^t(S)$. Three thresholds, namely Th_3, Th_4, and Th_5 are determined for $dif_i^t(A)$, $d_i^t(AR)$, and $dif_j^t(SDIR)$ in order to obtain the probable locations of falls using the knowledge of four object-level features. These three thresholds are defined as:

$$\begin{aligned} Th_3 &= th_A \left(= mean\left(dif_i(A)\right) + 3 \times \sigma_A\right) \\ Th_4 &= th_{AR} \left(= mean\left(d_i(AR)\right) + 3 \times \sigma_{AR}\right) \\ Th_5 &= th_{SDIR} \left(= mean\left(dif_i(SDIR)\right) + 3 \times \sigma_{SDIR}\right) \end{aligned} \tag{8}$$

where $mean((.))$ and $\sigma_{(.)}$ represent average and standard deviation of a set, say (.) corresponding to an object-level feature for P number of consecutive frames. The base of thresholds selection is that change in object-level features should be maximum during the fall event. Therefore, for a detected person, if ant one of the said object-level features is higher than its threshold, then probable fall is found at the location of this person, and the location is called probable location of falls.

A rule is defined by combining these three object-level features (i.e., $dif_i^t(A)$, $d_i^t(AR)$, and $dif_j^t(SDIR)$) in order to get more probable location of falls. The 'Rule 2' is defined as:

$$\begin{aligned} &if\ dif_i^t(A) \geq Th_3\ \&\ d_i^t(AR) \geq Th_4\ \&\ d_i^t(S) = 0\ \& \\ &(dif_j^t(S)_{j\neq i}^{n_t} \times dif_J^t(DIR)_{j\neq i}^{n_t}) \geq Th_5,\ then\ 'Fall' \\ &\qquad otherwise,\ \}No\ Fall' \end{aligned} \tag{9}$$

where n_t defines number of objects present in tth frame.

From the above discussion, it is also evident that, thresholds, Th_3, Th_4, and Th_5 are selected based on the statistical information of the object-level features. Therefore, these three thresholds will also be updated automatically with the contents present in the consecutive frames, thereby holding the generalization ability in handling both simple and complex scenarios. For a detected person, if the following rule is satisfied, then the location of this person is considered as specific location of fall, as defined:

$$\begin{aligned} &if\ o_i^t\ follows\ both\ Rule1\ and\ Rule2\ then\ \}Fall\ Must\ Occur' \\ &\qquad otherwise,\ \}No\ Fall' \end{aligned} \tag{10}$$

After classifying the event as 'Fall', the reliability of the decision is further ensured based on $Z-$numbers. This is explained in the next section.

4.4. Z-Numbers for ensuring the reliability of the detected fall event

Z-number is used to encode the knowledge of an event into words/phrases [61]. Z-numbers [62] have two tuples (i.e., $Z = (H; E)$): 'probability' (H) and 'reliability' (E). The first tuple is based on the probability of occurrence of an event (i.e., real-valued uncertain variable), whereas, the second tuple is defined using the linguistic terms corresponding to the same event based on the probability score. For each person concerning a detected fall, Z-numbers are computed based on the values of its features to ensure a higher degree of reliability for the detection. In the conventional way, $Z-$numbers are computed based on the probability score. As the rules are formulated in unsupervised way, crisp values are obtained corresponding to all OpenPose and object-level features. When the formulated rules satisfy the criteria based on OpenPose and object-level features and their thresholds, then corresponding event is classified as 'Fall'. Therefore, the values of both OpenPose and object-level features (i.e., $d_i^t(nk)$, $d_i^t(na)$, $dif_i^t(A)$, $d_i^t(AR)$, $\forall_j(dif_j^t(S))$, and $\forall_j(dif_j^t(DIR))$) are used in computing the normalized feature thresholding score (say, $d_i^t(NFT)$) in order to obtain the first tuple (H) of $Z-$numbers. The $d_i^t(NFT)$ is defined as:

$$\begin{aligned} &d_i^t(NFT) \\ &= \frac{(1 - d_i^t(N_{nk})) + (1 - d_i^t(N_{na})) + d_i^t(N_A) + d_i^t(N_{AR}) + Avg(\forall_j d_j^t(N_{SDIR}))}{5} \end{aligned} \tag{11}$$

where $d_i^t(N_{nk})(= \frac{d_i^t(nk)}{Th_1})$, $d_i^t(N_{na})(= \frac{d_i^t(na)}{Th_2})$, $d_i^t(N_A)(= \frac{dif_i^t(A)}{Th_3})$, and $d_i^t(N_{AR})$ $(= \frac{d_i^t(AR)}{Th_4})$ represent the normalized scores corresponding to $d_i^t(nk)$, $d_i^t(na)$, $dif_i^t(A)$, and $d_i^t(AR)$, respectively. $Avg(\forall_j d_j^t(N_{SDIR}))$ represents the average of normalized score corresponding to all detected foreground objects (except the falling person) in the tth frame, defined as:

$$Avg(\forall_{j\neq i}^{n_t} d_j^t(N_{SDIR})) = \frac{\sum_{j\neq i}^{n_t} \frac{(dif_j^t(S) \times dif_J^t(DIR))}{Th_5}}{n_t} \tag{12}$$

As it is defined earlier, the first tuple (H) of Z-numbers is based on the $d_i^t(NFT)$. One assumption is taken as: If $d_i^t(NFT)$ is high, then reliability of detection will also be high. During falls, the values of $d_i^t(nk)$, $d_i^t(na)$ should be minimum. Whereas, the values of $dif_i^t(A)$, $d_i^t(AR)$, and $Avg(\forall_{j\neq i}^{n_t} d_j^t(N_{SDIR}))$ should be high during falls. Therefore, to obtain the high value of $d_i^t(NFT)$, $(1 - d_i^t(nk))$, $(1 - d_i^t(na))$ are used in Eq. (11).

Based on the normalized feature thresholding score ($d_i^t(NFT)$) as defined in Eq. (11), some linguistic rules are defined to provide information about H and E in Z-numbers. The process of how Z-numbers are used in this study is illustrated using an example. If X =< Person Fall (PF) >: name of the variable, and Y : PF must happen in the frame F_t, if the normalized feature thresholding score for PF detection is greater than 0.9 and less than 1. Then, H: Context = < PF, normalized feature thresholding score > 0.9 and ≤ 1, and E : Relevance of H given X within the context of Y =<Certainty>. Here, 'Certainty' represents the linguistic term for E. Detailed results of Z-numbers for ensuring the correctly defined linguistic terms and classified event (i.e., Fall) are explained in Section 5.3.2. Incorporating Z-numbers with the OpenPose and object-level features-based rule generation constitutes the $Z-$DFT algorithm. The pseudo code of the developed $Z-$DFT for multi-person fall detection is shown in Algorithm 1. The effectiveness of the developed $Z-$DFT algorithm for fall detection is presented in the next section.

5. Results & discussions

Our study has three broad sub-objectives in order to show (i) the effectiveness of feature thresholding and rule generation in defining the probable and specific locations of falls, respectively, (ii) the utility of Z-numbers over feature thresholding and rule generation for ensuring the detection, and (iii) the superiority of $Z-$DFT algorithm over some state-of-the-art methods. The dataset details, experimental setup, experimental results, and safety intervention are discussed in the following sections.

5.1. Datasets

Most of the available fall datasets contain simulated falls instead of real-life falls. The characteristics of real-life fall is varied from one scenario to another. Whereas, in simulated data, fall scenario is designed by experts. Therefore, it may not be matched with the real-life scenario. Thus, simulated data may not be an accurate source for both training/validation and testing of the developed fall detection algorithm ($Z-$DFT). Therefore, in this study, to assess the effectiveness of the developed $Z-$DFT, two video datasets, namely UR Fall dataset (URFD) [63] and YouTube8M (YT8M) [64] are used. These two video datasets contain real-life videos with diverse visual entities. As already defined, in $Z-$DFT, feature thresholding is done in unsupervised way, Therefore, only validation is done to check the effectiveness of feature thresholding in $Z-$DFT. From URFD, 12 videos (containing indoor 'fall' and 'no fall' events) are used for both validation and testing purposes. Whereas, YT8M is a large-scale unlabeled dataset containing millions of videos having various kinds of possible traffic scenarios. Of them, 6 videos containing pedestrian fall are selected. YT8M has no annotated videos; therefore, it is used for testing purposes only to get the generalization ability of the developed $Z-$DFT algorithm. The validation and test videos (acquired from YT8M and URFD) consist of 730 to 1450 frames with a variety of real-life indoor (e.g., home) and outdoor (e.g., road and construction area) scenarios, including occlusion, multi-falls, and no fall.

As mentioned earlier, the detector, G-RCNN is used for the person detection. G-RCNN is trained over the data, namely PASCAL VOC 07 (VOC 07). This is a static image data. It contains $5K$ training and $5K$ testing images of 20 classes. Out of these, the samples corresponding to the object class, 'Person' is considered in the training process. A total of 524 training images are used. Out of these, the images contain 'person' is considered as positive class and rest are considered as negative class. On the other hand, OpenPose model is trained over MSCOCO data. The details of experimental setup is presented in the next section.

5.2. Experimental setup

The developed $Z-$DFT algorithm was coded using python 3.8.6 (Anaconda) in Windows 10 with an Intel(R) Core(TM) i7-10750H CPU @ 2.60 GHz-2.59 GHz. The libraries used are 'cv2' and 'NumPy'. For this study, the URFD dataset is divided into validation (60%) and test (40%) subsets. In the developed $Z-$DFT algorithm, five thresholds are used for detecting the fall event. How is the optimal thresholds selection done, is explained in Section 5.3.1.

5.3. Experimental results

Experiments are carried out to evaluate the efficacy of the developed $Z-$DFT algorithm for fall detection, in line with three objectives (refer to Section 5). Objective (i) aims to show the effectiveness of feature thresholding and rule generation in detecting the probable and specific locations of falls, as defined in Section 5.3.1. Objective (ii) shows how Z-numbers are effective in obtaining the reliability of classified fall event, as stated in Section 5.3.2. Objective (iii) demonstrates the superiority of the developed $Z-$DFT over some state-of-the-art methods, as presented in Sections 5.3.3, 5.3.4, and 5.3.5.

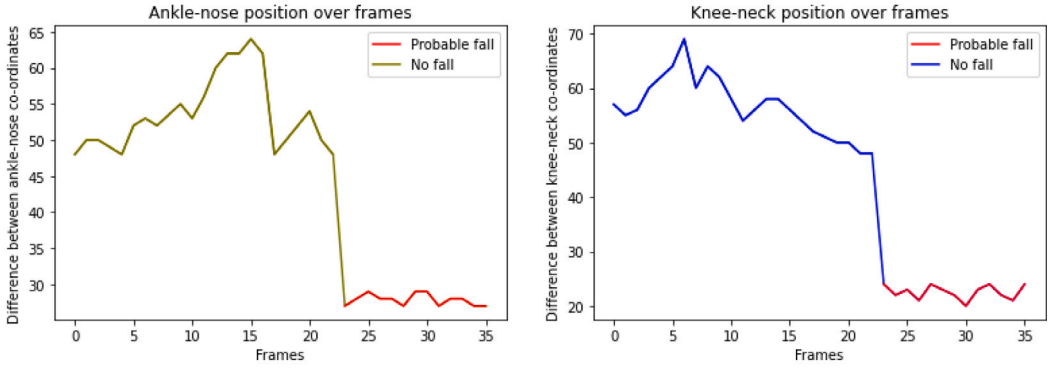
5.3.1. The effectiveness of feature thresholding and rule generation for obtaining the probable and specific locations of falls

Qualitative assessment of feature thresholding and rule generation are presented in this section. During the validation phase, feature thresholding is done to obtain optimal thresholds for both OpenPose and object-level features, as explained in Sections 5.3.1(i) and 5.3.1(ii), respectively. Visualization of specific locations of falls corresponding to formulated rules is presented in Section 5.3.1(iii).

(i) *Thresholds selection for OpenPose features:*
Openpose features corresponding to multi-persons are illustrated in Fig. 3. From Fig. 3(a), it is seen that 13 and 8 key points are extracted for left-sided and right-sided persons, respectively. For these two persons, different number of key points are extracted. This depends on the poses of detected persons and orientation of camera. Four OpenPose features, namely ankle, knee, neck, and nose are required for fall detection. The relevant pair of features for the multi-persons are detected in Fig. 3(b). From this figure, it is seen that for the left-sided person, two OpenPose features, such as ankle and nose are detected. Whereas, for the right-sided person, two other OpenPose features, such as neck and knee are detected. This depends on the human pose-orientation which is visible in videos. It is also evident that, these four features may not be detected all time. Therefore, any one of the two derived OpenPose features such as (i) distance between neck and knee (i.e., $d_i(nk)$) and (ii) distance between nose and ankle (i.e., $d_i(na)$) for the ith detected person are useful in obtaining the specific locations of falls based on two thresholds, Th_1 and Th_2.
The graphs of $d_i(na)$ and $d_j(nk)$ for ith and jth detected persons over P number of consecutive frames are shown in Fig. 4(a) and Fig. 4(b), respectively. In Figs. 4(a) and 4(b), the x- and y-axis represent the frames over time and the distance between relevant features, respectively. These graphs are represented by line charts with different colors. These colors have been used to distinguish the progression of coordinate before and after fall. If the distance between the nose and ankle, or knee and neck is higher, it means person is walking or standing. Person lean forward when stumbling. Therefore, distance between nose and ankle, or knee and neck becomes less during the fall. Hence, the lower values of thresholds, Th_1 and Th_2 are preferable. Keeping this in mind, the thresholds, Th_1 and Th_2 are selected using the information of (mean - 3× standard deviation). A number of $10(= P)$ consecutive frames are used in this study for optimal thresholds selection.
As seen in Fig. 4(a), at 23rd frame, the value of $d_i(na)$ is decreased abruptly. Based on the statistical information (i.e., $mean = 48.8$ and $\sigma = 2.96$) of previous 10 frames for ith detected person, the threshold, $Th_1 = 39.92$ is selected. Whereas, in the same frame, $d_i(na)$ is $14.3(< 39.92)$. Therefore, in this frame, probable fall may occurs. As seen from Fig. 4(a), olive-colored line represents the normal scenario. Whereas, in case of fall scenario, this line is turned to red-color. The result corresponding to $d_i(na)$ and optimal value of threshold, Th_1 is presented in Fig. 7(b) (refer to the left-sided person). On the other hand, in Fig. 4(b), at 23rd frame, the value of $d_j(nk)$ for jth detected person is also decreased abruptly. Based on the statistical information (i.e., $mean = 47.1$ and $\sigma = 1.83$) of 10 consecutive frames for jth detected person, the threshold, $Th_2 = 41.61$ is selected. Whereas, in the same frame, $d_j(nk)$ is $24.6(< 41.61)$. Therefore, in this frame, probable fall may occurs. As seen from Fig. 4(b), blue-colored line represents the normal scenario. Whereas, in case of fall scenario, this line is turned to red-color. The result

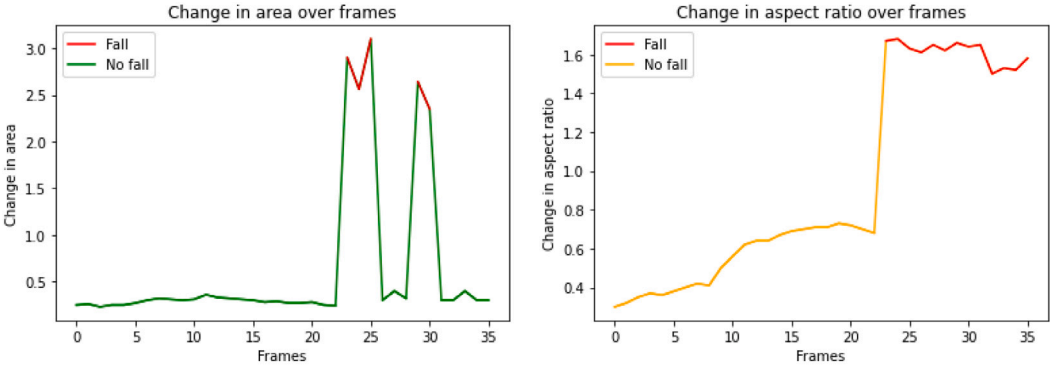
(a) All features (b) Relevant features

Fig. 3. OpenPose features.



(a) (b)

Fig. 4. Graphs for the normalized distances, $d_i(na)$ and d_i^{nk} over the consecutive frames.



(a) (b)

Fig. 5. Graphs for the (a) change in area ($dif_i(A)$) and (b) aspect ratio ($d_i(AR)$) over consecutive frames.

corresponding to $d_i(nk)$ and optimal value of threshold, Th_2 is presented in Fig. 7(b) (refer to the right-sided person). From these two graphs, it is evident that the thresholds corresponding to the 23rd frame are selected as $Th_1 = 39.92$ and $Th_2 = 41.61$. As these thresholds are automatically updated based the statistical information corresponding to each detected object (person), it holds the generalization ability in thresholds selection for OpenPose features.

(ii) *Thresholds selection for object-level features:*
As said earlier, four object-level features, such as change in area, aspect ratio, speed variation, and change in direction are also effective in detecting the fall locations. Thresholds, Th_3 and Th_4 are used for thresholding two object-level features, such as change in area and aspect ratio, respectively. Whereas, a derived feature using both speed variation and change in direction is also used in fall detection. Threshold, Th_5 is used for the combination of speed variation and change in direction. How are the best configuration results found for these three thresholds, is presented in this section.
The graph of change in area ($dif(A)$) vs. frames is presented in Fig. 5(a). It is found that, during the fall event, falling person first bend down and then lie on the floor/road. Therefore, during the fall event, an abrupt change in area is found. From Fig. 5(a), it is evident that, an abrupt change in area is found at 23rd frame for ith detected person. Based on the statistical information (i.e., $mean = 0.38$ and $\sigma = 0.07$) of change in area for ten

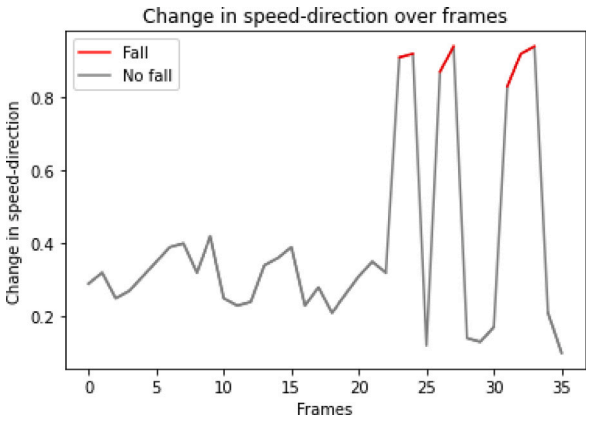


Fig. 6. Graph for $dif_j(S) \times dif_j(DIR)$ over consecutive frames.

consecutive frames corresponding to ith detected person, the threshold, $Th_3 = 0.59$ is selected. Whereas, in the same frame, $dif_i(A)$ is 2.9(> 0.59). Therefore, in this frame, a fall may occurs using the knowledge of change in area. As seen from Fig. 5(a), green-colored line represents the normal scenario. Whereas, in case of fall scenario, this line is turned to red-color. Similarly, the graph of aspect ratio vs. frames is presented in Fig. 5(b). As already defined, during the fall event, the aspect ratio of the falling person is increased abruptly. From Fig. 5(b), it is evident that, an abrupt change in aspect ratio is found at 23rd frame. Based on the statistical information (i.e., $mean = 0.65$ and $\sigma = 0.09$) of aspect ratio for ten consecutive frames corresponding to ith detected person, the threshold, $Th_4 = 0.92$ is selected. Whereas, in the same frame, $d_i(AR)$ is 1.67(> 0.92). Therefore, in this frame, a fall may occurs using the knowledge of aspect ratio. As seen from Fig. 5(b), orange-colored line represents the normal scenario. Whereas, in case of fall scenario, this line is turned to red-color.

The movements of trajectories for other moving persons will also change during the fall event. The multiplication of speed variation and change in direction (for movements) corresponding to the other foregrounds is computed in order to check is there any abrupt change in behavior of traffic flow during the fall event. Graph between the multiplication scores (of speed variation and direction change) vs. frames for ith foreground is shown in Fig. 6. From Fig. 6, it is evident that, an abrupt change in the said multiplication scores is found at 23rd frame. Based on the statistical information (i.e., $mean = 0.32$ and $\sigma = 0.08$) of the multiplication scores for ten consecutive frames corresponding to ith detected person, the threshold, $Th_5 = 0.56$ is selected. Whereas, in the same frame, $dif_j(S) \times dif_j(DIR)$ is 0.92(> 0.56). Therefore, in this frame, a fall may occurs using the knowledge of said multiplication score. As seen from Fig. 6, gray-colored line represents the normal scenario. Whereas, in case of fall scenario, this line is turned to red-color. The values of aforesaid thresholds, Th_3, Th_4, and Th_5 are varies from video to video based on their contents. Therefore, in the developed $Z-$DFT algorithm, thresholds are selected in an unsupervised way. Hence, it holds the generalization capability. Based on the optimal thresholds (i.e., Th_3, Th_4, and Th_5) for object-level features, results corresponding to the probable location of fall(s) are shown in Fig. 7(a) and Fig. 7(b), respectively.

As already defined, feature thresholding is done to obtain probable locations of falls corresponding to each OpenPose and object-level features. This reduces the searching space for obtaining the specific locations of falls. Therefore, these features are combined to formulate rules in order to obtain the specific locations of falls, as explained in the next section.

(iii) *Visualization of person fall detection:*
To strengthen the fall detection results, rules are formulated using both OpenPose and object-level features for obtaining the specific locations of falls. By keeping this in mind, two rules are defined in Eqs. (6) and (9) using OpenPose and object-level features, respectively. As seen in Fig. 7(b), for the left-sided person, the difference between the position of ankle and nose is used in Eq. (6) for classifying the event as 'Fall' or 'No fall'. In the same figure, for right-sided person, the difference between neck and knee is used in Eq. (6) for classifying the event as 'Fall' or 'No fall'. The rule-based system using OpenPose features is defined to obtain specific locations of falls. On the other hand, for both left and right sided persons, all defined object-levels features (i.e., change in area, aspect ratio, speed variation, and change in direction) are checked to see whether these features satisfy the rule defined in Eq. (9). If yes, then the event is classified as 'Fall'. In order to obtain the exact (specific) location of 'Fall', both rules defined in Eqs. (6) and (9) must be satisfied.
Results are shown in Fig. 8. Fig. 8(a) indicates there is no fall in the frame, therefore, detected persons are marked by blue-colored bounding boxes. Whereas, Fig. 8(b) represents the fall event for multi-person. Here, fallen persons are marked by red-colored bounding boxes. From this figure, it is also evident that each fallen person is assigned with a unique ID. As an example, another fall event is shown in Fig. 9. This figure represents a single fall scenario. Two scenarios, 'No fall' and 'Fall' are presented by Fig. 9(a) and Fig. 9(b), respectively. After fall detection, its corresponding video frame is further used for validation task to enhance the detection accuracy. To further ensure the detection, Z-numbers are computed, as explained in the next section.

5.3.2. *The usefulness of Z-numbers along with formulated rules for ensuring the detection*

For each detected object (i.e., person) concerning a fall, Z-numbers are computed to ensure the detection. The first tuple (H) of Z-numbers is defined by two linguistic terms (LTs), namely 'Absolutely Important' (AI), and 'Important' (I) based on the feature thresholding score which is computed using the values of all OpenPose and object-level features. The second tuple (E) of Z-numbers represents the reliability of H. Three linguistic terms (LTs), namely 'Likely' (LY), 'Most Likely' (ML), and 'Certainty' (CY) are assigned to the second tuple, E based on this score and assigned linguistic term corresponding to H. This is about the reliability of the decision. Ranges of output scores for the LTs assigned to E are as follows: 'LY': (0, 0.2, 0.5); 'ML' : (0.5, 0.7, 0.9); and 'CY' : (0.9,0.9, 1). The transformation rules of LTs for (H, E) are defined using the information of normalized feature thresholding scores. The transformation rules for (H, E) for falls are reported in Table 1. It is evident from the table that $d_i^t(NFT) > 0.9$ is absolutely important for the decision, detected class = 'Fall', and corresponding LT, 'CY' (certainty) is assigned to E. It indicates that the reliability of the decision is certain. Similarly, the transformation rules for 'No fall' event are also shown in this table. In this way, Z-numbers provide a degree of reliability corresponding to detected falls.

For a scenario, if the defined two rules (refer to Eqs. (6) and (9)) are satisfied along with the condition, $d_i^t(NFT) > 0.9$ (Z: X = Fall, H = AI, and E = CY), then it is ensured that the event is 'Fall'. This result prove the effectiveness of Z-numbers in ensuring the reliability of the detected fall event. Thus, Z-numbers combined with the deep feature thresholding and rule generation handle the uncertainty issue arising between various 'Fall' and 'No Fall' events under complex scenarios. The effectiveness of $Z-$DFT over some state-of-the-art methods is depicted in the next section.

(a) (b)

Fig. 7. Probable fall detection results: (a) Fall Scenario 1 and (b) Fall Scenario 2.

(a) (b)

Fig. 8. Specific locations of multi-falls: (a) No Fall and (b) Fall.

(a) (b)

Fig. 9. Single fall detection result: (a) No Fall and (b) Fall.

Table 1
Z-number for Person Fall detection.

Variable (X)	Constraint (H)	Reliability (E)
Fall	AI, $d_i^I(NFT) > 0.9$	CY
Fall	AI, $0.5 < d_i^I(NFT) < 0.9$	ML
Fall	AI, $d_i^I(NFT) < 0.5$	LY
Fall	I, $0.5 < d_i^I(NFT) < 0.9$	CY
Fall	I, $d_i^I(NFT) < 0.5$	ML
No Fall	AI, $d_i^I(NFT) < 0.5$	CY
No Fall	I, $0.5 < d_i^I(NFT) < 0.9$	ML

5.3.3. Comparative study

To validate the developed Z−DFT, a comparative study between the Z−DFT and some state-of-the-art methods for fall detection is conducted over some videos acquired from the YouTube8M and URFD data. Accuracy has been considered as the performance metric for this comparative study. Various well-known fall detection algorithms, including SVM-based (SVMFD) [31], KNN-based (FD-KNN) [36], CNN-based (CNNFD) [65], Deep neural network-based (DNN) [66], Trajectory weighted deep convolutional rank pooling-based (TWDCRP) [67], Deep learning and activity characteristics-based (DLAFD) [40], ARFD-Net [43], LSTM-based [5], GRU-based [42], and Multi-human fall detection (MHFD) [41] are used for the comparative study. All these algorithms are applied over YouTube8M and URFD datasets. Here, capturing device is single camera. In SVMFD, CNNFD, and LSTM-based methods, SVM, CNN, and LSTM are trained with the real-time images containing two classes 'Fall' and 'No Fall'. Whereas, in FD-KNN, silhouette ratio for a person is detected and used to train the KNN for fall detection. In DNN, deep network is trained using the human posture information for fall detection. Another network, namely TWD-CRP uses weighted trajectory-based ranked pooling infomration for fall detection. In DLAFD, Faster RCNN is used for object (i.e., person) detection, and then object level features are extracted and analyzed for fall detection. Whereas, in ARFDNet, latent feature pooling is used. In GRU-based method, LSTM is modeled with OpenPose features. In

Table 2
Results of comparative study over YouTube8M and URFD data.

Author	Data	Classification technique	Accuracy (%)
Z−DFT (Our method)	YouTube8M	Z−number-based deep feature thresholding	98.04
SVMFD	YouTube8M	SVM	66.1
FD-KNN	YouTube8M	KNN+Silhouette ratio	74.4
CNNFD	YouTube8M	CNN	69.02
DNN	YouTube8M	Human posture	81.47
TWDCRP	YouTube8M	Weighted trajectory + Ranked pooling information	94.23
DLAFD	YouTube8M	Faster RCNN feature based thresholding	86.5
ARFDNet	YouTube8M	Latent feature pooling	83.69
LSTM-based	YouTube8M	LSTM+OpenPose	61.8
GRU-based	YouTube8M	OpenPose feature thresholding	69.3
MHFD	YouTube8M	YOLO+3DCNN	70.26
Z−DFT (Our method)	URFD	Z−number-based deep feature thresholding	99.63
CNNFD	URFD	CNN	89.31
DNN	URFD	Human posture	83.86
TWDCRP	URFD	Weighted trajectory + Ranked pooling information	96.04
DLAFD	URFD	Faster RCNN feature based thresholding	93.58
ARFDNet	URFD	Latent feature pooling	95.12
LSTM-based	URFD	LSTM+OpenPose	92.1
GRU-based	URFD	OpenPose feature thresholding	98.2
MHFD	URFD	YOLO+3DCNN	99.66

MHMD, YOLO+3D-CNN is used for multi-fall detection. In our developed algorithm, Z−numbers-based deep feature thresholding (Z−DFT) is done for fall detection.

The comparative results over YouTube8M and URFD data are presented in Table 2. From Table 2 (refer to 1st row), it is found that the developed Z−DFT outperforms the state-of-the-art when YouTube8M is used. From the same table (refer to 12th row), it is also evident that Z−DFT is comparable with the state-of-the-art when URFD is used. The reason is: URFD is created in a controlled environment, whereas, YouTube8M is a real-life data containing various issues, including occlusion, low illumination, and uncertainty. Therefore, the state-of-the-art deep models (i.e., CNNFD, DLAFD, ARFDNet, LSTM-based, DNN, TWDCRP, and GRU-based) for fall detection perform good in URFD but not in YouTube8M. Moreover, LSTM, GRU, and MHMD-based methods require huge amounts of real data which may not affordable always. On the other hand, SVMFD and FD-KNN are based on conventional machine learning techniques (SVM and KNN), these are limited to the simple traffic flow. The comparison between Z−DFT, SVMFD, and FD-KNN is done to see the effectiveness of feature-based rule-base system over the conventional machine learning techniques. From the Table 2, it is seen that the Z−DFT is superior to SVMFD (refer to 2nd row) and FD-KNN (refer to 3rd row) in terms of detection accuracy.

In DNN, TWDCRP, CNNFD, DLAFD, ARFDNet, and Z−DFT, deep feature thresholding-based rule-base system is done for fall detection. Therefore, the comparison between Z−DFT and aforesaid state-of-the-art methods (DNN, TWDCRP, CNNFD, DLAFD, and ARFDNet) is done to see the effectiveness of the combination of deep feature thresholding-based rules generation with Z−numbers for fall detection in both indoor and outdoor scenarios. In the aforesaid state-of-the-art methods, either OpenPose features or object-level features are used for rule generation. Whereas, in Z−DFT, both OpenPose and object-level features are used to strengthen the decision of fall detection. Moreover, in Z−DFT, Z−numbers are computed based on the deep feature thresholding score to ensure the detection. The combination of Z−numbers with deep feature thresholding-based rule base system handles the uncertainty issue arising between various fall and no-fall events under complex environment.

As already defined, object (i..e, person) detection is prerequisite of fall detection, correct prediction of the location of person over the frames also affects the fall detection accuracy. From the Table 2, it is seen that the Z−DFT is superior to DNN, TWDCRP, CNNFD, DLAFD, and ARFDNet. It means, persons are correctly detected by our developed Z−DFT as compared to CNNFD, DLAFD, and ARFDNet. Hence, the deep features corresponding to a person, generated in our developed algorithm is more accurate than other three. Moreover, rule-based system defined in Z−DFT is accurate. Further, the use of Z−numbers over rule-based system, strengthen the decision. Thus, our developed Z−DFT can be effectively used for both indoor and outdoor fall detection. As per the authors' knowledge, this is the first time when the concept of Z-numbers is used for fall detection. As an example, some pictorial views of the resultant falls over YouTube8M and URFD datasets (corresponding to the outdoor and indoor scenarios, respectively) are shown in Fig. 10. The computational complexity of the Z−DFT is stated in the next section.

5.3.4. Computational complexity

The developed Z−DFT consists of four stages: (i) ODT, (ii) feature extraction, (iii) feature analysis and classification, and (iv) computation of Z-numbers for ensuring the reliability of detection. The computational time for each of these stages is added to obtain the overall computational complexity of the developed Z−DFT. The computational time of any algorithm is proportional to its computational complexity. The computational complexity of each stage is provided below.

(i) **Computational complexity for object (i.e., person) detection and tracking:** OD is done using deep network G-RCNN, and tracking task is done by making a proper assignment between same objects detected over two consecutive frames. G-RCNN consists of G-AlexNet architecture. As said earlier, G-AlexNet consists of five convolutional, three pooling, one RoI, two anchor generation, and three fully connected layers. At the first convolution layer of G-AlexNet, 96 filters each having size 11×11 with stride = 4, and padding = 0 are used for the convolution operation. Let $N \times N$ be the size of the input fed to the G-AlexNet for the generation of feature maps. Computational complexity for this operation is defined as:

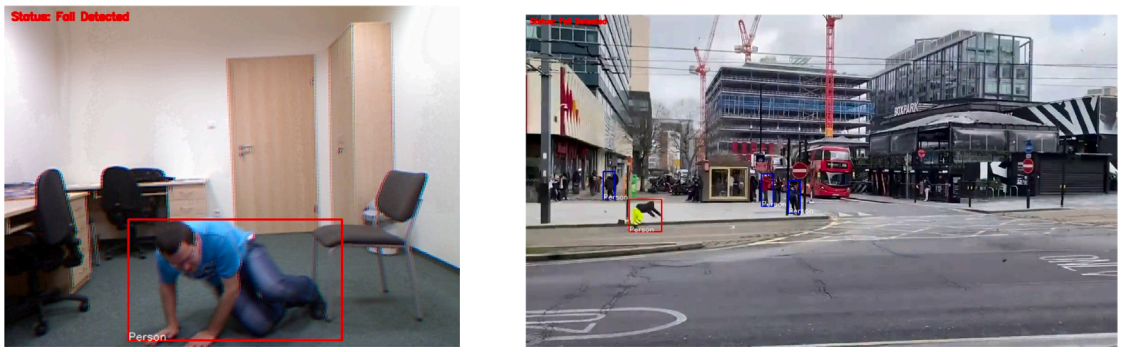
$$O(96 \times ((N - 11 + 2 \times 0)/4) + 1 \times ((N - 11 + 2 \times 0)/4) + 1)$$
$$= O(96 \times (N - 7)/4 \times (N - 7)/4) = O(N^2) \quad (13)$$

Therefore total computational complexity for the convolution operation at five convolutional layers is $5 \times O(N^2)$. At each pooling layer, a filter size of 3×3 with stride = 2 is used. Computational complexity for first pooling layer operation is defined as:

$$O((N - 11)/2 \times (N - 11)/2) = O(N^2) \quad (14)$$

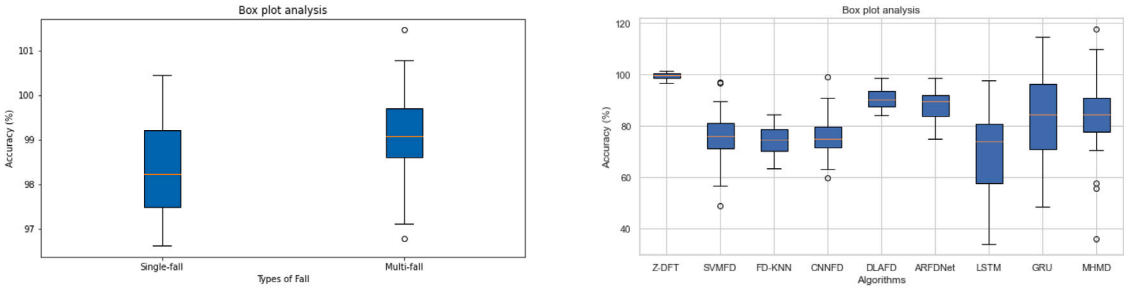
The total computational complexity for the pooling operation at three pooling layers is $3 \times O(N^2)$. Computational complexity for the formation of spatio-temporal granules over Pool1 layer is defined as:

$$O(N) + O(1) \quad (15)$$



(a) URFD frame (b) YouTube8M frame

Fig. 10. Sample results for falls.



(a) $Z-$DFT results (b) Comparative results

Fig. 11. Robustness checking using Box plot.

Computational complexity for the generation of RoI and anchor-feature map is $O(1)+O(1)=O(2)$. Finally, computational complexity for three fully connected layers, classification and bounding-box regression is defined as:

$$O(1)+O(1)+O(1)+9\times O(1)+9\times O(1)=21\times O(1) \tag{16}$$

The total computational complexity for detection task is defined as:

$$\begin{aligned} O(G-RCNN) &= 5\times O(N^2)+3\times O(N^2) \\ &\quad +O(N)+O(1)+O(2)+21\times O(1) \\ &= 8\times O(N^2)+O(N)+24\times O(1) \end{aligned} \tag{17}$$

The computational complexity for tracking task is defined as:

$$O(tracking)=O((n\times(n-1))/2) \tag{18}$$

where, n represents the number of person detected in a frame. Therefore, total computational complexity for detection and tracking task is as follows:

$$8\times O(N^2)+O(N)+24\times O(1)+O((n\times(n-1))/2) \tag{19}$$

(ii) **Computational complexity for feature extraction:** As already defined, OpenPose and object level features are extracted for each detected person. Here, four type of object-level features are extracted. If n number of persons are detected in any frame, then computational complexity for extracting features corresponding to these detected persons is $O(5n)$.

(iii) **Computational complexity for feature analysis and classification:** It consists of two steps: computation for the analysis of OpenPose features and computational complexity for analysis of object-level features. In the analysis of OpenPose features, initially, the position difference between two relevant features is computed, and then check, if this position difference exceeds threshold value or not. Therefore computational complexity for OpenPose feature analysis and classification is $O(2)$. For object-level feature analysis corresponding to each detected person, change in area, speed, and direction are computed. The computational complexity of this operation is $O(3)$. Using the object-level features, one rule is defined for fall detection. Therefore, total computational complexity for object-level feature analysis and classification is $O(4)$. Hence, total computational complexity for feature analysis and classification is $O(2)+O(4)=O(6)$.

(iv) **Computational complexity for Z-numbers computation for ensuring the reliability of the classified fall event:** Based on the normalized feature thresholding score (refer to Eq. (11)) (defined in the rule base system), the transformation rule (refer to Table 1) for Z-numbers is defined. The computational complexity for this operation is $O(1)$.

The total computational complexity of the developed $Z-$DFT is defined as:

$$\begin{aligned} &8\times O(N^2)+O(N)+24\times O(1)+O((n\times(n-1))/2) \\ &\quad +O(5n)+O(6)+O(1) \\ &=8\times O(N^2)+O(N)+O((n\times(n-1))/2)+O(5n)+O(31) \end{aligned} \tag{20}$$

Table 3 shows the comparative results between the developed $Z-$DFT and some other state-of-the-art, including SVMFD, FD-KNN, CNNFD, DLAFD, ARFDNet, LSTM-based (LSTMb), GRU-based (GRUb), and MHMD in terms of run time. It is observed from the table that the developed $Z-$DFT is superior to SVMFD, DLAFD, ARFDNet, LSTM, GRU, and MHMD and comparable with FD-KNN and CNNFD. In our developed $Z-$DFT granulation is done over the convoluted feature map for accurate detection of person even in case of occlusion. Correct detection of person also affects the fall detection accuracy. Moreover, Z-numbers are computed based on the deep feature thresholding score to ensure the reliability of the class of the detected event. Due to the aforesaid tasks, our developed $Z-$DFT takes larger time as compared

Table 3
Results of required computation time for fall detection algorithms.

Algorithms	SVMFD	FD-KNN	CNNFD	DLAFD	ARFDNet	LSTMb	GRUb	MHMD	$Z-$DFT
Speed(fps)	8	13	11	9	10	7	6	9	11

to FD-KNN and CNNFD. From the previous comparative study, it is already proved that the $Z-$DFT is superior to the aforesaid state-of-the-art in terms of fall detection accuracy. Therefore, in order to make a trade off between fall detection accuracy and speed, $Z-$DFT is the superior choice.

5.3.5. Robustness checking

A total of 18 videos acquired from YouTube8M and URFD datasets are utilized to investigate the robustness of the $Z-$DFT for fall detection. Consequently, a total of 18 accuracy are obtained that help in the generation of a box-plot for robustness checking of the developed $Z-$DFT. As already defined, the $Z-$DFT can detect both 'single fall' and 'multi fall'. The robustness of $Z-$DFT in terms of box plot analysis for both single and multi fall detection is shown in Fig. 11(a). It is evident from this figure that the developed $Z-$DFT for multi-fall detection shows the higher accuracy with lowest range of dispersion as compared to single-fall detection. The model ($Z-$DFT) is able to achieve an average accuracy of 98.87% for single fall and 99.2% for multi fall. Another box plot analysis is done for the $Z-$DFT, SVMFD, FD-KNN, CNNFD, DLAFD, ARFDNet, LSTM, GRU, and MHMD over both YouTube8M and URFD data to prove the robustness of the $Z-$DFT over others. Result is exhibited in Fig. 11(b). From this figure, it is evident that the developed fall detection algorithm (i.e., $Z-$DFT) for both YouTube8M and URFD data shows maximum accuracy with the lowest degree of dispersion as compared to SVMFD, FD-KNN, CNNFD, DLAFD, ARFDNet, LSTM, GRU, and MHMD. Hence, the robustness of $Z-$DFT is proved.

6. Conclusions

Fall events on roads (i.e., outdoor) are increasing each year. It is happened due to either collision, near-miss, poor road condition, lack of maintenance, incompetency, and(or) negligence of human. Moreover, in home, the number of falls relates to aged persons increases day by day. Therefore, human-safety becomes pertinent nowadays. The real-time detection of falls and adaptation of on-time corrective and preventive actions are required for improving the human-safety. Previous studies are mainly focused on indoor (i.e., home) and outdoor (i.e., construction area) falls based on simulated data. Moreover, no research has been conducted on handling the uncertainty issue arising between various 'Fall' and 'No Fall' events under complex scenarios. In order to address these issues, we have developed a new algorithm, namely $Z-$numbers-based deep feature thresholding ($Z-$DFT) for fall detection in both indoor and outdoor areas.

To demonstrate the efficacy of the developed $Z-$DFT, comparisons with some state-of-the-arts (i.e., SVM-based (SVMFD) [31], KNN-based (FD-KNN) [36], CNN-based (CNNFD) [65], Deep neural network-based (DNN) [66], Trajectory weighted deep convolutional rank pooling-based (TWDCRP) [67], Deep learning and activity characteristics-based (DLAFD) [40]), ARFDNet [43], LSTM-based [5], GRU-based [42], and Multi-human fall detection (MHFD) [41] are conducted using two datasets, namely YouTube8M [64] and URFD [63]. Speed (fps) and accuracy (%) are used as performance metrics for the evaluation. From the comparative study, it is seen that the developed $Z-$DFT is applicable for the detection of single- and multi-falls in both indoor and outdoor areas. This flexibility enables its usage in a wide variety of instances. Moreover, it is also seen from the experiments that the developed $Z-$DFT is superior in terms of both accuracy and speed. We have also demonstrated that the developed $Z-$DFT is robust than the said state-of-the-art. Therefore, our study aims to hold contributions in both theoretical and piratical aspects.

6.1. Theoretical contributions

The conceptual framework of the general fall detection system comprises five phases: (i) analysis of the characteristics of falls, (ii) installation of CCTV, (iii) development of a fall detection algorithm, (iv) training/validation, and (iv) implementation. We have contributed in the aforesaid fourth and fifth phases. The developed $Z-$DFT has four phases: (i) object detection and tracking, (ii) feature extraction, (iii) feature thresholding and rule generation, and (iv) $Z-$numbers-based analysis for ensuring the detection. Unlike state-of-the-art methods, both OpenPose and object-level features are used in $Z-$DFT for better modeling of falls. Some features are derived using the OpenPose (i.e., position of ankle, nose, neck, and knee) and object-level (i.e., change in area, aspect ratio, speed variation, and change in direction) features. The derived features are further analyzed for determining optimal thresholds in order to obtain the probable locations of falls. These features are combined using two formulated rules for obtaining the specific locations of falls. Further, $Z-$numbers are computed using the feature thresholding scores in order get the degree of detection. As per the author's knowledge, this is the first time when the concept of $Z-$numbers is used for fall detection.

6.2. Practical implication

From the practical point of view, fall detection system is very useful in enhancing the human-safety. Using the developed $Z-$DFT algorithm, fall events can be detected from live stream videos. After the detection of a fall, a video clip corresponding to the same event is generated and stored in the database. This video clip can be used in future for analyzing the probable cause(s) of the event. In case of both indoor (i.e., home) and outdoor (i.e., road) scenarios, the video clips are used to aware the concerned persons about their abnormal behavior related to falls. The training of these persons is done to rectify their faults, thereby reducing the number of fall events related to the abnormal behavior. Whereas, if the cause is related to the environment, then correct preventive measures should be taken by the concerned authority to stop re-occurring the causes of falls. In addition, after the on-time detection of fall, an alarm is generated in the control room for sending an immediate help to the fall location in order to mitigate the impact of fall. All these enhance the human-safety.

6.3. Effectiveness of the developed $Z-$DFT algorithm

The effectiveness of the developed $Z-$DFT algorithm for fall detection are: (i) uncertainty issue arising between various indoor and outdoor falls, and no-falls under complex scenarios is addressed, (ii) OpenPose features combined with object-level features enable $Z-$DFT in modeling both indoor and outdoor falls using a single unified system, (iii) As feature thresholding is done using the statistical information of video contents, $Z-$DFT holds the generalization ability in detecting both indoor and outdoor falls, (iv) the usefulness of $Z-$numbers ensures the reliability of the detected fall event, and (v) the developed fall detection system enhances the human safety.

6.4. Limitation and future scopes for the study

The present study has a few limitations. Only four OpenPose features are considered in this study. More number of features may be used for enhancing the detection accuracy. Moreover, the scope of the present study is limited to the monocular vision only. Future

studies may consider the development of a fall detection algorithm for stereo vision. Additionally, rough concept [68,69] may be used with $Z-$numbers to enhance the detection capability. In this context, a decision support system [70] may also be developed for better managing the fall detection.

Algorithm 1: $Z-$DFT Algorithm.

Input: Tuple ($\{L\}$) containing x and y coordinates of all $k = 18$ features for n number of detected person in the current video frame (t^{th}).
Output: $\{Output\}$: Fall or No Fall.
Initialize the set $\{L\}_t, t = 0, 1, ..., 6 \leftarrow \phi$
Initialize the set $\{Output\} \leftarrow \phi$
for $j = 0$ *to* n **do**
  for $i = 0$ *to* k **do**
    Compute $d_i^t(nlk)$, $d_i^t(nrk)$, $d_i^t(nla)$, and $d_i^t(nra)$ using Eq. (3).
    Compute $d_i^t(nk)$ and $d_i^t(na)$ using Eq. (4).
    if $d_i^t(nk) \leq Th_1$ *or* $d_i^t(na) \leq Th_2$ **then**
      Fall may happen
      Compute $d_i^t(A)$, $d_i^t(AR)$, $d_i^t(S)$, and $d_i^t(DIR)$ using Eq. (2).
      Compute $dif_i^t(A)$, $dif_i^t(S)$, and $dif_i^t(DIR)$ using Eq. (7).
      if $dif_i^t(A) \geq Th_3$ *and* $d_i^t(AR) \geq Th_4$ **then**
        if $d_i^t(S) = 0$ *and* $(dif_j^t(S)_{j\neq i}^{n_t} \times dif_J^t(DIR)_{j\neq i}^{n_t}) \geq Th_5$ **then**
          Fall event is detected
          Compute $d_i^t(NFT)$ using Eq. (11).
          Compute Z-number based on $d_i^t(NFT)$.
          if *Z-number for a fall event follow rule (AI,* $d_i^t(NFT)$>0.9*)* **then**
            Fall must happen
            $\{Output\} \leftarrow$ Fall event
          end
        end
      end
    end
  end
end
return *Output*
End

CRediT authorship contribution statement

Anima Pramanik: Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Formal analysis, Data curation, Conceptualization. **Sobhan Sarkar:** Writing – review & editing, Writing – original draft, Visualization, Supervision, Formal analysis, Conceptualization. **Sankar K. Pal:** Writing – review & editing, Supervision, Conceptualization.

Declaration of competing interest

The authors have no relevant financial or nonfinancial interests to disclose. The authors have no conflicts of interest to declare that are relevant to the content of this article. There are no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript. The authors have no financial or proprietary interests in any material discussed in this article.

Data availability

Data will be made available on request.

Acknowledgements

Prof. Sankar K. Pal acknowledges National Science Chair, Science and Engineering Research Board, Department of Science and Technology (SERB-DST), Government of India.

References

[1] World Health Organization, Falls, 2021, https://www.who.int/news-room/fact-sheets/detail/falls.
[2] S.S. Khan, J. Hoey, Review of fall detection techniques: A data availability perspective, Med. Eng. Phys. 39 (2017) 12–22.
[3] K. Singh, A. Rajput, S. Sharma, Human fall detection using machine learning methods: A survey, Int. J. Math. Eng. Manag. Sci. 5 (1) (2020) 161–180.
[4] S.K. Yadav, K. Tiwari, H.M. Pandey, S.A. Akbar, A review of multimodal human activity recognition with special emphasis on classification, applications, challenges and future directions, Knowl.-Based Syst. 223 (2021) 106970.
[5] M. Taufeeque, S. Koita, N. Spicher, T.M. Deserno, Multi-camera, multi-person, and real-time fall detection using long short term memory, in: Medical Imaging 2021: Imaging Informatics for Healthcare, Research, and Applications, Vol. 11601, International Society for Optics and Photonics, 2021, 1160109.
[6] M. Everingham, L. Van Gool, C.K. Williams, J. Winn, A. Zisserman, The pascal visual object classes (voc) challenge, Int. J. Comput. Vis. 88 (2) (2010) 303–338.
[7] E. Alam, A. Sufian, P. Dutta, M. Leo, Vision-based human fall detection systems using deep learning: A review, Comput. Biol. Med. (2022) 105626.
[8] J. Gutiérrez, V. Rodríguez, S. Martin, Comprehensive review of vision-based fall detection systems, Sensors 21 (3) (2021) 947.
[9] S. Sarkar, A. Pramanik, J. Maiti, An integrated approach using rough set theory, anfis, and z-number in occupational risk prediction, Eng. Appl. Artif. Intell. 117 (2023) 105515.
[10] J. Redmon, A. Farhadi, Yolov3: An incremental improvement, 2018, arXiv preprint arXiv:1804.02767.
[11] A. Pramanik, S.K. Pal, J. Maiti, P. Mitra, Granulated RCNN and multi-class deep sort for multi-object detection and tracking, IEEE Trans. Emerg. Top. Comput. Intell. 6 (1) (2021) 171–181.
[12] S.K. Pal, A. Pramanik, J. Maiti, P. Mitra, Deep learning in multi-object detection and tracking: state of the art, Appl. Intell. 51 (2021) 6400–6429.
[13] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, Y. Sheikh, OpenPose: realtime multi-person 2D pose estimation using part affinity fields, IEEE Trans. Pattern Anal. Mach. Intell. 43 (1) (2019) 172–186.
[14] S. Mitra, S.K. Pal, Logical operation based fuzzy MLP for classification and rule generation, Neural Netw. 7 (2) (1994) 353–373.
[15] D.P. Mandal, C. Murthy, S.K. Pal, Formulation of a multivalued recognition system, IEEE Trans. Syst. Man Cybern. 22 (4) (1992) 607–620.
[16] A. Pathak, S.K. Pal, Fuzzy grammars in syntactic recognition of skeletal maturity from x-rays, IEEE Trans. Syst., Man, Cybernet. 16 (5) (1986) 657–667.
[17] A. Pramanik, S.K. Pal, J. Maiti, P. Mitra, Traffic anomaly detection and video summarization using spatio-temporal rough fuzzy granulation with Z-numbers, IEEE Trans. Intell. Transp. Syst. 23 (12) (2022) 24116–24125.
[18] N. Blanc, B. Steux, T. Hinz, LaRASideCam: A fast and robust vision-based blindspot detection system, in: 2007 IEEE Intelligent Vehicles Symposium, IEEE, 2007, pp. 480–485.
[19] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, Adv. Neural Inf. Process. Syst. 28 (2015) 91–99.
[20] R. Kalsotra, S. Arora, Background subtraction for moving object detection: explorations of recent developments and challenges, Vis. Comput. 38 (12) (2022) 4151–4178.
[21] A. Pramanik, C. Djeddi, S. Sarkar, J. Maiti, et al., Region proposal and object detection using hog-based CNN feature map, in: 2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy, ICDABI, IEEE, 2020, pp. 1–5.
[22] K.-Y. Park, S.-Y. Hwang, An improved haar-like feature for efficient object detection, Pattern Recognit. Lett. 42 (2014) 148–153.
[23] Z. Chen, T. Ellis, A self-adaptive Gaussian mixture model, Comput. Vis. Image Underst. 122 (2014) 35–46.
[24] R. Huang, J. Pedoeem, C. Chen, Yolo-lite: a real-time object detection algorithm optimized for non-GPU computers, in: 2018 IEEE International Conference on Big Data, Big Data, IEEE, 2018, pp. 2503–2510.
[25] A. Yilmaz, O. Javed, M. Shah, Object tracking: A survey, Acm Comput. Surveys (CSUR) 38 (4) (2006) 13–es.
[26] H. Lee, D. Kim, Salient region-based online object tracking, in: 2018 IEEE Winter Conference on Applications of Computer Vision, WACV, IEEE, 2018, pp. 1170–1177.
[27] N. Saunier, T. Sayed, A feature-based tracking algorithm for vehicles in intersections, in: The 3rd Canadian Conference on Computer and Robot Vision, CRV'06, IEEE, 2006, p. 59.
[28] C.I. Patel, R. Patel, Contour based object tracking, Int. J. Comput. Electr. Eng. 4 (4) (2012) 525.
[29] A. Sobral, A. Vacavant, A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos, Comput. Vis. Image Underst. 122 (2014) 4–21.
[30] W. Chen, L. Cao, X. Chen, K. Huang, An equalized global graph model-based approach for multicamera object tracking, IEEE Trans. Circuits Syst. Video Technol. 27 (11) (2016) 2367–2381.

[31] C. Doukas, I. Maglogiannis, P. Tragas, D. Liapis, G. Yovanof, Patient fall detection using support vector machines, in: IFIP International Conference on Artificial Intelligence Applications and Innovations, Springer, 2007, pp. 147–156.

[32] S. Sarkar, S. Vinay, R. Raj, J. Maiti, P. Mitra, Application of optimized machine learning techniques for prediction of occupational accidents, Comput. Oper. Res. 106 (2019) 210–224.

[33] F.-Y. Leu, C.-Y. Ko, Y.-C. Lin, H. Susanto, H.-C. Yu, Fall detection and motion classification by using decision tree on mobile phone, in: Smart Sensors Networks, Elsevier, 2017, pp. 205–237.

[34] M. Rella Riccardi, F. Mauriello, S. Sarkar, F. Galante, A. Scarano, A. Montella, Parametric and non-parametric analyses for pedestrian crash severity prediction in great britain, Sustainability 14 (6) (2022) 3188.

[35] P. Mazurek, R.Z. Morawski, Application of naïve Bayes classifier in fall detection systems based on infrared depth sensors, in: 2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS, 2, IEEE, 2015, pp. 717–722.

[36] C.-L. Liu, C.-H. Lee, P.-M. Lin, A fall detection system using k-nearest neighbor classifier, Expert Syst. Appl. 37 (10) (2010) 7174–7181.

[37] J. Seo, T. Kim, J. Lee, J. Kim, J. Choi, G. Tack, Fall prediction of the elderly with a logistic regression model based on instrumented timed up & go, J. Mech. Sci. Technol. 33 (8) (2019) 3813–3818.

[38] W.-Y. Cai, J.-H. Guo, M.-Y. Zhang, Z.-X. Ruan, X.-C. Zheng, S.-S. Lv, GBDT-based fall detection with comprehensive data from posture sensor and human skeleton extraction, J. Healthc. Eng. 2020 (2020).

[39] B.-H. Wang, J. Yu, K. Wang, X.-Y. Bao, K.-M. Mao, Fall detection based on dual-channel feature integration, IEEE Access 8 (2020) 103443–103453.

[40] W. Min, H. Cui, H. Rao, Z. Li, L. Yao, Detection of human falls on furniture using scene analysis based on deep learning and activity characteristics, IEEE Access 6 (2018) 9324–9335.

[41] M.E.N. Gomes, D. Macêdo, C. Zanchettin, P.S.G. de Mattos-Neto, A. Oliveira, Multi-human fall detection and localization in videos, Comput. Vis. Image Underst. 220 (2022) 103442.

[42] C.-B. Lin, Z. Dong, W.-K. Kuan, Y.-F. Huang, A framework for fall detection based on openpose skeleton and lstm/gru models, Appl. Sci. 11 (1) (2020) 329.

[43] S.K. Yadav, A. Luthra, K. Tiwari, H.M. Pandey, S.A. Akbar, ARFDNet: An efficient activity recognition & fall detection system using latent feature pooling, Knowl.-Based Syst. 239 (2022) 107948.

[44] S. Tateno, F. Meng, R. Qian, Y. Hachiya, Privacy-preserved fall detection method with three-dimensional convolutional neural network using low-resolution infrared array sensor, Sensors 20 (20) (2020) 5957.

[45] M.M. Islam, O. Tayan, M.R. Islam, M.S. Islam, S. Nooruddin, M.N. Kabir, M.R. Islam, Deep learning based systems developed for fall detection: A review, IEEE Access 8 (2020) 166117–166137.

[46] A. Núñez-Marcos, G. Azkune, I. Arganda-Carreras, Vision-based fall detection with convolutional neural networks, Wirel. Commun. Mobile Comput. 2017 (2017).

[47] M. Musci, D. De Martini, N. Blago, T. Facchinetti, M. Piastra, Online fall detection using recurrent neural networks, 2018, arXiv preprint arXiv:1804.04976.

[48] J. Nogas, S.S. Khan, A. Mihailidis, Deepfall: non-invasive fall detection with deep spatio-temporal convolutional autoencoders, J. Healthc. Informat. Res. 4 (1) (2020) 50–70.

[49] D. Razum, G. Seketa, J. Vugrin, I. Lackovic, Optimal threshold selection for threshold-based fall detection algorithms with multiple features, in: 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO, IEEE, 2018, pp. 1513–1516.

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

[51] M. Kreković, P. Čerić, T. Dominko, M. Ilijaš, K. Ivančić, V. Skolan, J. Šarlija, A method for real-time detection of human fall from video, in: 2012 Proceedings of the 35th International Convention MIPRO, IEEE, 2012, pp. 1709–1712.

[52] W. Chen, Z. Jiang, H. Guo, X. Ni, Fall detection based on key points of human-skeleton using openpose, Symmetry 12 (5) (2020) 744.

[53] M. Gao, J. Li, D. Zhou, Y. Zhi, M. Zhang, B. Li, Fall detection based on OpenPose and MobileNetV2 network, IET Image Process. 17 (3) (2023) 722–732.

[54] A. Yajai, S. Rasmequan, Adaptive directional bounding box from RGB-D information for improving fall detection, J. Vis. Commun. Image Represent. 49 (2017) 257–273.

[55] L. Yao, W. Min, K. Lu, A new approach to fall detection based on the human torso motion model, Appl. Sci. 7 (10) (2017) 993.

[56] M.A. Mousse, C. Motamed, E.C. Ezin, Video-based people fall detection via homography mapping of foreground polygons from overlapping cameras, in: 2015 11th International Conference on Signal-Image Technology & Internet-Based Systems, SITIS, IEEE, 2015, pp. 164–169.

[57] U. Pratap, M.A. Khan, A. Jalai, Human fall detection for video surveillance by handling partial occlusion scenario, in: 2016 11th International Conference on Industrial and Information Systems, ICIIS, IEEE, 2016, pp. 280–284.

[58] M.A. Mousse, B. Atohoun, Saliency based human fall detection in smart home environments using posture recognition, Health Inf. J. 27 (3) (2021) 14604582211030954.

[59] B. Wan, D. Zhou, Y. Liu, R. Li, X. He, Pose-aware multi-level feature network for human object interaction detection, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 9469–9478.

[60] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C.L. Zitnick, Microsoft coco: Common objects in context, in: European Conference on Computer Vision, Springer, 2014, pp. 740–755.

[61] S.K. Pal, D. Bhoumik, D. Bhunia Chakraborty, Granulated deep learning and Z-numbers in motion detection and object recognition, Neural Comput. Appl. 32 (21) (2020) 16533–16548.

[62] L.A. Zadeh, A note on Z-numbers, Inform. Sci. 181 (14) (2011) 2923–2932.

[63] M. Kepski, UR fall detection dataset, 2022, http://fenix.univ.rzeszow.pl/~mkepski/ds/uf.html. (Online Accessed 19 June 2022).

[64] S.S. Thomas, S. Gupta, V.K. Subramanian, Event detection on roads using perceptual video summarization, IEEE Trans. Intell. Transp. Syst. 19 (9) (2017) 2944–2954.

[65] W. Liu, J. Guo, Z. Huang, W. Qiu, Falling-Action Analysis Algorithm Based on Convolutional Neural Network, Atlantis Press, 2016.

[66] Y. Fan, M.D. Levine, G. Wen, S. Qiu, A deep neural network for real-time detection of falling humans in naturally occurring scenes, Neurocomputing 260 (2017) 43–58.

[67] Z. Zhang, X. Ma, H. Wu, Y. Li, Fall detection in videos with trajectory-weighted deep-convolutional rank-pooling descriptor, IEEE Access 7 (2018) 4135–4144.

[68] S. Sarkar, S. Baidya, J. Maiti, Application of rough set theory in accident analysis at work: a case study, in: 2017 Third International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN), IEEE, 2017, pp. 245–250.

[69] A. Pramanik, S. Sarkar, J. Maiti, P. Mitra, Rt-gsom: rough tolerance growing self-organizing map, Inf. Sci. 566 (2021) 19–37.

[70] S. Sarkar, M. Chain, S. Nayak, J. Maiti, Decision support system for prediction of occupational accident: a case study from a steel plant, in: Emerging Technologies in Data Mining and Information Security: Proceedings of IEMIS 2018, Volume 2, Springer, 2018, pp. 787–796.