A Skeleton-Free Fall Detection System From Depth Images Using Random Decision Forest

Ahmed Abobakr, *Graduate Student Member, IEEE*, Mohammed Hossny, *Member, IEEE*, and Saeid Nahavandi, *Senior Member, IEEE*

***Abstract*—Interest in enhancing medical services and healthcare is emerging exploiting recent technological capabilities. An integrable fall detection sensor is an essential component toward achieving smart healthcare solutions. Traditional vision-based methods rely on tracking a skeleton and estimating the change in height of key body parts such as head, hips, and shoulders. These methods are often challenged by occluded body parts and abrupt posture changes. This paper presents a fall detection system consisting of a novel skeleton-free posture recognition method and an activity recognition stage. The posture recognition method analyzes local variations in depth pixels to identify the adopted posture. An input depth frame acquired using a Kinect-like sensor is densely represented using a depth comparison feature and fed to a random decision forest to discriminate among standing, sitting, and fallen postures. The proposed approach simplifies the posture recognition into a simple pixel labeling problem, after which determining the posture is as simple as counting votes from all labeled pixels. The falling event is recognized using a support vector machine. The proposed approach records a sensitivity rate of 99% on synthetic and live datasets as well as a specificity rate of 99% on synthetic datasets and 96% on popular live datasets without invasive accelerometer support.**

***Index Terms*—Decision forest, fall detection, posture recognition, RGB-D, skeleton-free, support vector machine (SVM).**

I. Introduction

FALLS are considered a major health problem and the second leading cause of accidental or unintentional injury deaths worldwide. According to a global estimation, 37 million falls of which 646 000 are fatal occur each year. A fall is defined as an event that results in a person coming to rest inadvertently on the ground or floor or other lower level structures [1]. Everyone can be at risk of having a fall, but the natural aging process often places adults older than 65 years of age at an increasing risk of having a fatal fall due to the presence of long-term health conditions [2]. Frequent loss of balance, unconsciousness, dizziness, visual and muscle impairments, and slipping are the major contributory factors of a fall [3].

For elders, falls have serious psychological consequences such as losing confidence and independence in addition to other long-term effects. Furthermore, Tinetti *et al.* [4] have demonstrated that delayed medical assistance and staying on the floor for long periods increase the risk of both physical and psychological complications. Therefore, an automatic and independent fall detection for aged care facilities has been a goal for medical health research for decades. Additionally, obtaining a robust fall detection system is essential toward achieving a smart healthcare environment. The recent technological advancements have made obtaining smart medical services an accelerating multidisciplinary area of research.

This paper proposes a vision-based integrable and automated fall detection system. The design goals of the proposed solution are

1) *Increased robustness* in terms of accommodating wide variety of lighting conditions and anthropometric variations as well as resilience to occlusions.
2) *Reduced complexity* that ensures real-time response on embedded devices.
3) *Absolute privacy* measures that extend to implementing the system everywhere, even in bathrooms, which are a prime place of frequent falls due to slipping.

Toward these goals, the proposed system is composed of three main processing modules: an image acquisition module, a posture recognition module, and an activity recognition module. Fig. 1 shows an overview of the processing stages of the proposed system.

We chose RGB-D sensors for the image acquisition module. Traditional RGB cameras measure color intensities with a high dependence on illumination. This imposes difficulties in performing foreground separation, which is an essential preprocessing step. On the other hand, depth imaging technologies estimate the distance of three-dimensional (3-D) points in the scene away from the imaging plane. This feature has dramatically simplified a wide range of vision tasks. Depth cameras use infrared structured light [5] to acquire depth information enabling illumination independence. Additionally, depth measurements simplify performing 3-D foreground segmentation using simple thresholding. The popular Microsoft Kinect RGB-D sensor has been used extensively for medical applications. Webster and Celik [6] systematically reviewed and demonstrated the use of Kinect in elderly care and stroke rehabilitation [7]. Although

Manuscript received December 25, 2016; revised April 8, 2017, July 28, 2017, and November 14, 2017; accepted November 18, 2017. Date of publication December 27, 2017; date of current version August 23, 2018. This work was supported by the Institute for Intelligent Systems Research and Innovation, Deakin University, Australia. The database was created with funding from NSF EIA-0196217. *(Corresponding author: Mohammed Hossny.)*

The authors are with the Institute for Intelligent Systems Research and Innovation, Deakin University, Geelong, VIC 3216, Australia (e-mail: aabobakr@deakin.edu.au; mohammed.hossny@deakin.edu.au; saeid.nahavandi@deakin.edu.au).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors. The supplementary files include an mp4 video file. The file size is 36.7 MB.

Digital Object Identifier 10.1109/JSYST.2017.2780260

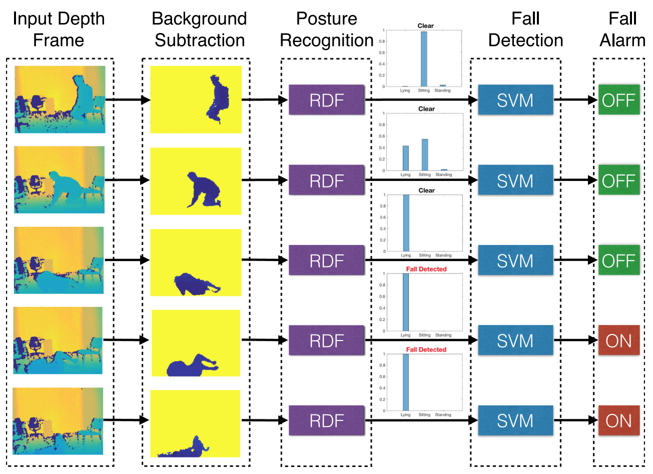

Fig. 1. Proposed fall detection method. First, a calibrated background frame is subtracted. Second, an RDF model is applied on the foreground body pixels to recognize the articulated posture. Third, an SVM model analyzes the change in lying posture confidence to detect the occurrence of fall events.

Kinect v2, the second version of Microsoft Kinect, produces less noisy depth images, we chose to constraint the proposed solution to Kinect v1. The reason behind this is the increased power consumption and cooling requirements of Kinect v2, which is not suitable for deployment on embedded devices [8]. These requirements are dictated by the time of flight imaging technology in Kinect v2 as compared to the structured light technology in Kinect v1.

The second module is a posture recognition component. It identifies the adopted body posture in each depth frame through an efficient pixel-wise representation and a random decision forest (RDF) classifier. Typically, an input depth frame is preprocessed by a background subtraction algorithm. Then, each visible body pixel is densely represented using the depth comparison feature (DCF) and evaluated using the RDF classifier. Hence, each body pixel contributes by voting with a certain level of confidence for being a part of a certain pose. Finally, the majority of pixel votes determines whether the current pose being lying, sitting, or standing.

The third module is a support vector machine (SVM) classifier to detect the occurrence of falls via analyzing the pattern of the lying posture confidence overtime.

The rest of this paper is organized as follows: Section II presents a brief survey on recent fall detection methods. Section III presents the proposed skeleton-free posture recognition method. Section IV describes the method of building SVM models for fall detection based on the output of the posture recognition module. Experiments and results are discussed in Section V. Finally, conclusion and future advancements are highlighted in Section VI.

II. Fall Detection: A Brief Survey

Enhancing healthcare and medical services is a multidisciplinary area of research that receives high interest. Fall detection, in particular, has witnessed much focus due to the physical and psychological complications of falls on elders [4]. Recently, Mubashir *et al.* [9] have categorized fall detection approaches

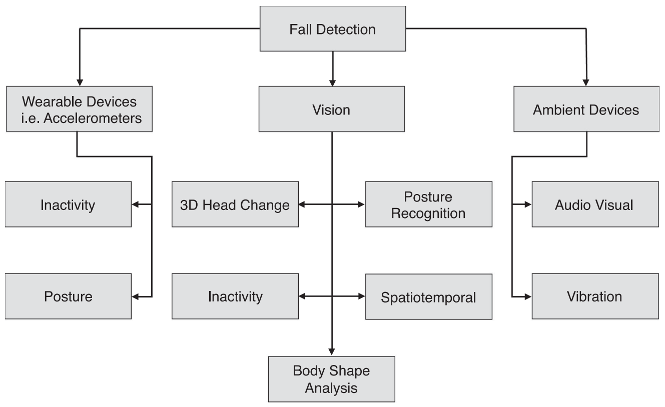

Fig. 2. Fall detection approaches as categorized by Mubashir *et al.* [9]. The proposed method combines ideas from both the posture recognition and the spatiotemporal categories to detect the fall action.

into three main categories: wearable devices based, ambience device based, and vision-based, as shown in Fig. 2.

The most popular wearable devices based approach is analyzing a wide range of physical activity measurements captured by an accelerometer attached to the human body. Mathie *et al.* [10] used a single waist-mounted accelerometer to monitor different parameters of human movements such as posture orientation and energy of movement. In their approach, abnormalities in parameter readings initiate the fall alarm. Inertial frame velocity measures acquired via accelerometer were compared with a predefined threshold to identify fall activities in [11]. Lai *et al.* [12] distributed multiple triaxial accelerometers over certain body parts to sense injuries resulting from accidental fall occurrence. However, relying on wearable devices does have limitations such as battery lifetime and being easily disconnected and forgotten. Furthermore, several studies have concluded that older adults prefer nonwearable sensors [13], [14].

Another approach relies on analyzing audio, video, and vibrational information sensed by ambient devices. Li *et al.* [15] performed preprocessing enhancements and sound source localization. Then, a feature extraction and acoustic classification using the K nearest neighbor classifier into falls or nonfalls were employed. This was also supported by measuring audio signals using eight omnidirectional microphones. Further improvements were done on this system relying on the Kinect sensor microphone localization capabilities in [16]. Zhuang *et al.* [17] classified sound signals using an SVM with a Gaussian mixture model kernel. Floor vibration based fall detection systems were investigated in [18] and [19]. Fusion of floor vibration and audio signals was used by Zigel *et al.* [20] to detect fall events. The drawback of relying on ambient devices is frequent false alarms resulting from noisy audio and vibration signals of falling objects [9].

Interest in vision-based systems is emerging toward achieving effective computerized healthcare via exploiting the recent technological achievements. Visual information processing approaches are further divided into body shape change, posture, key joint altitude, and spatiotemporal analysis methods. These approaches require computationally efficient processing of images or videos in order to achieve real-time performance. The use of a traditional two-dimensional (2-D) stereo camera for

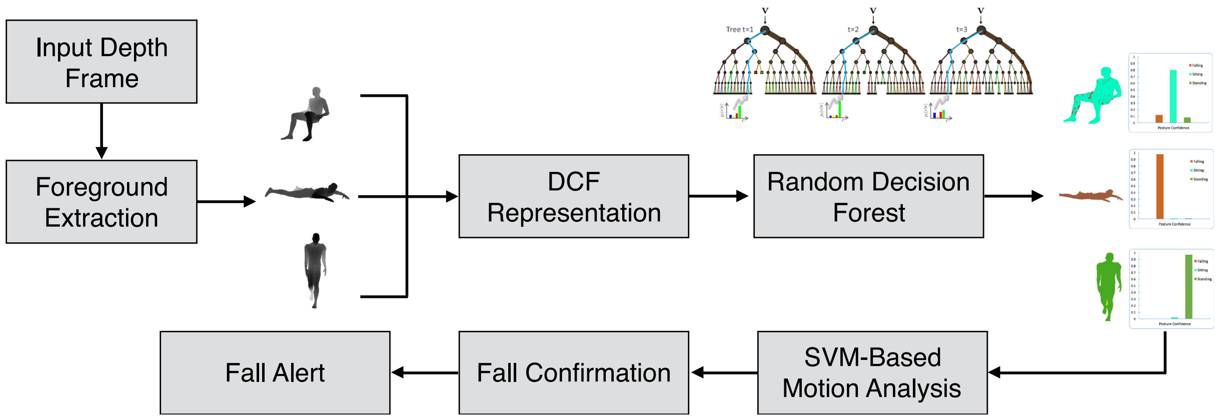

Fig. 3. Fall detection system architecture. Input depth frame captured using Kinect is first preprocessed by foreground extraction. Then, the DCF feature extractor is applied. Feature representations of depth pixels are evaluated using an RDF classifier. The change in lying posture confidence levels is analyzed using a support vector machine classifier. Finally, an alert is triggered in case of a fall is confirmed.

fall detection was demonstrated in [21]–[23] via posture analysis, [24] by analyzing shape deformation, and [25] via 3-D head tracking.

However, 2-D approaches are plagued by the difficulty of separating subjects from backgrounds, also known as the foreground separation step [26]. Thus, they either suffer from poor performance, need uniform backgrounds, or need stationary cameras with static backgrounds for background subtraction [14], [26]. Therefore, a growing trend is to use the depth cameras as they provide much richer geometrical information and facilitate preprocessing tasks such as background subtraction and objects delineation. Furthermore, privacy concerns and low/no light challenges limit applications of fall detection systems relying on RGB cameras.

The Kinect depth sensor has been commonly used for fall detection [14], [27]–[31]. Stone and Skubic [14] characterized person vertical state in depth frames over time. Then, a temporal segmentation was used to identify ground events. Five features were extracted for each ground event and an RDF was used to compute a confidence that a fall preceded on a ground event. In this method, the vertical state characterization was done based on predefined measurements. Also, the whole body should be visible, as any segmentation artifacts of ground events will relatively affect the feature extraction process. Bian *et al.* [27] developed a two-stage system. First, the 3-D coordinates of body joints were interpolated from segmented body parts. Second, an SVM classifier was used to detect the fall based on the positions of the extracted joint trajectory over time. Limitations of this method are high dependence on head position and predefined threshold values.

The most common vision-based approach to recognize a fallen posture is by extracting 3-D skeleton data and tracking body parts such as the head and the hip. Skeleton data could automatically be obtained using joints localization via body parts tracking or joint offset regressors [5], [27], [32]. There are several limitations to this approach documented in the literature. First, the joints tracking module provided by the Kinect SDK has a limited range from 1.5 to 4 m [14], [33]. Also, the Kinect sensor would fail to extract the skeleton during the rapid fall motion [14], [27]. Second, joints tracking based methods require ground floor calibration and cannot detect falls correctly when the person lies down on furniture [27]. Third, local body part detectors and offset joint regressors have a major challenge with missing body parts occluded by objects in the environment or other body parts (self-occlusion). Finally, these methods rely on massive amounts of training data to achieve a generic model that fits all postures [5], [26].

To address these limitations, the proposed method analyzes the body posture holistically and does not rely on skeleton extraction, joints tracking, joint altitude thresholds, or ground plane calibration as shown in Fig. 3. The proposed solution builds on Buys *et al.*'s [26] and Shotton *et al.*'s [5] contributions. We extended Buys *et al.*'s data generation module to articulate postures in distinctive labels. We also tuned the window parameters of Shotton *et al.*'s DCF feature to densely represent the whole posture.

III. Methods: Holistic Posture Recognition

The classical pose estimation relies on a presumed skeletal model [34] of the observed subjects. This model provides *a priori* information on number of body parts and the relationship between them. To analyze a depth frame, this method is divided into three pipelined problems.

First, a classification problem is formulated to segment a depth frame into body parts. Second, a regression problem is formulated on a selected set of body parts (i.e., joints) to localize the 3-D coordinates of each joint. Third, an expectation maximization problem is formulated to keep track of the localized joints and hence maintain the extracted skeleton even if some body parts are hidden. This allows the extracted skeleton to accommodate a wide range of postures. However, this comes at a price of increased complexity for simple yet critical application domains such as fall detection, which requires learning only three postures.

The posture recognition module is trained to discriminate among variations of the lying, sitting, and standing postures. We analyze the whole posture by allowing each body pixel in a single depth frame to vote for the occurrence of a certain posture. This module uses an RDF model that is trained on DCF features extracted from each body pixel in the depth frame.

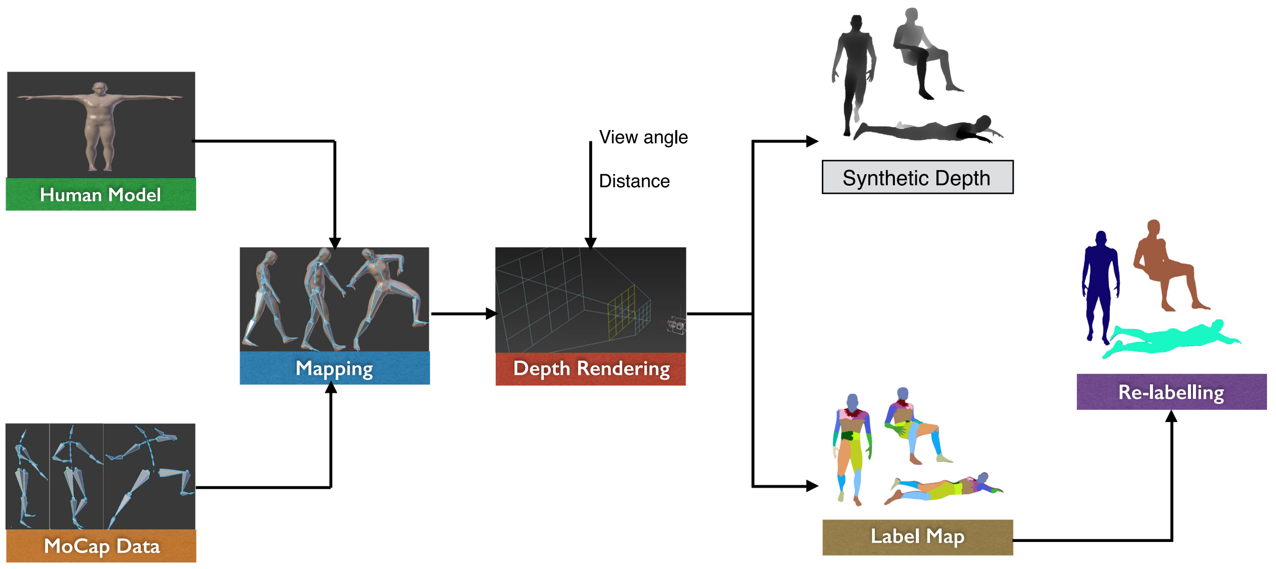

Fig. 4. Customized data generation pipeline for the holistic posture recognition. Motion frames of high degree of dissimilarity are mapped onto a 3-D model. The rendering process is parameterized with the camera view angle and distance, simulates the existence of multiple cameras, and produces depth and label image pairs. Finally, obtained body part labels are mapped to a single label representing the current pose, so that, all body pixels are voting for the same pose.

The motivation for the proposed method is twofold. First, it allows overcoming the problem of occluded body parts efficiently without memory filters. Second, it maintains high classification accuracy with low resolution frames (80 $\times$ 60 compared to 640 $\times$ 480). Nevertheless, collecting a pixel-labeled dataset of human poses remains a difficult and time-consuming task. Therefore, we extend the implementation of the state-of-the-art data synthesis pipeline [26]. The idea of using synthetic data has been proven effective and sufficient for achieving high generalization performance in [5] and [26].

A. Training Data

The building blocks of the data generation pipeline are shown in Fig. 4. Typically, motion capture (MoCap) data are mapped into virtual human models synthesizing virtually infinite and diverse datasets for different application domains. For instance, for a fall detection application, different fall scenarios could be recorded using an MoCap system for a single subject and mapped or replicated onto different 3-D human models.

1) Articulating Postures to Human Models: The data generation process starts with creating realistic 3-D virtual human models. We have created a single male model of weight 90 kg and height 190 cm for generating training images. MoCap data are used to articulate this model. MoCap data are a variation of human activities recorded using marker-based MoCap systems with real humans [35]. The only limitation of the data synthesis approach is the availability of MoCap data. However, a wide collection of human activities is recorded and made available for the research community in CMU's Graphics Lab Motion Capture Database [35]. This dataset is organized into subjects such as running, walking, dancing, etc. We manually select a subset of subjects that contains variations of the poses of interest, which are standing, sitting, and lying on the ground. The chosen motions are used to articulate the created model. During the mapping process, body pixels are grouped and assigned to the respective body part identified by body joints. Body pixels are animated by subsequent transformations applied to body joints, as shown in Fig. 5.

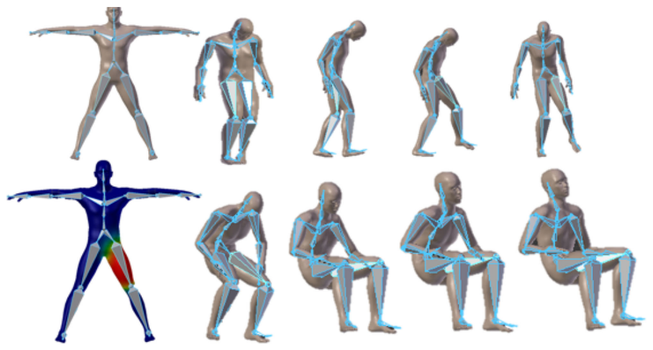

Fig. 5. Articulating a virtual human model. In Blender, the skeletal structure of a motion capture file is fitted onto the 3-D model and animation information is transferred to the model.

2) Rendering Synthetic Depth Frames: To achieve better generalization behavior with unseen scenarios, a training dataset that features maximum invariance possible is generated. Geometrical data of the virtual model are rendered using camera view angles from 0 to 360 with a step of 45^{o}. The null-valued pixels are set to the maximum depth of 10 m. Each synthetic depth map has an associated ground truth labels image generated where each body pixel in the depth map has a corresponding body part label in the labels map. This approach was used in training the skeletal tracking system of the Kinect [5] and for training a body parts detector in [26]. The resulting depth-label image pairs could further be used for training joints regression models.

3) Posture Labeling: However, the argument of this work is that, instead of using body parts or joints tracking based methods to recognize the current pose, we are going to perform a holistic posture analysis. Therefore, as a key contribution to this work, the resulting depth and label maps are clustered into three poses. Then, body pixels are relabeled and assigned the same label that characterizes the current pose, either standing, sitting, or lying, as shown in the last stage of Fig. 4.

This rendering pipeline could be considered as a generative model that could synthesize a virtually infinite set of images. Hence, allowing large-scale training of supervised models for a wide range of applications, including but not limited to, fall detection. Increasing the number of training samples has been proven beneficial in terms of obtaining a generalizable classifier and reducing the misclassification error. However, the computational resources become a challenge while expanding the training set. Consequently, a compensation should be made through experiments to optimize the proper amount of training data that are sufficient for generalization.

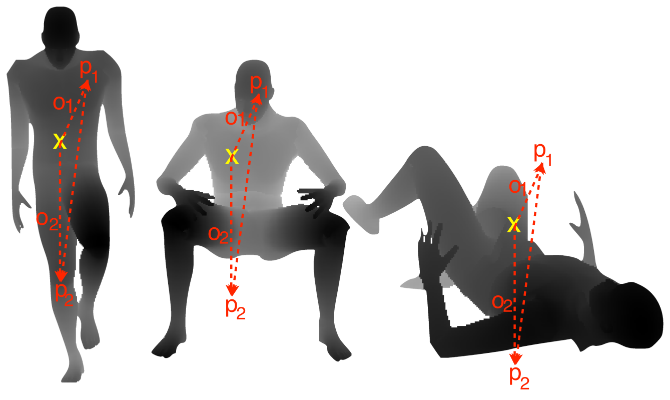

Fig. 6. Applying the DCF to discriminate among standing, sitting, and lying postures. For a body pixel x, the DCF is the depth difference between two other points p_1 and p_2 computed using shared pixel location offsets o_1 and o_2. Thus, the DCF has different responses according to the pixel x being classified and the current posture.

B. Feature Extraction

One essential step toward achieving a highly accurate vision system is to optimize a feature set that represents and discriminates visual information. The features being computationally efficient is an additional constraint to ensure real-time performance. Therefore, we use the simple DCF to represent body pixels. For a body pixel p in an input depth image I, the feature response is computed as

$$f(I, p|\Theta) = d_I\left(p + \frac{o_1}{d_I(p)}\right) - d_I\left(p + \frac{o_2}{d_I(p)}\right) \quad (1)$$

where $d_I(p)$ is a function returns the depth value at pixel p in the input image I, and $\Theta = (o_1, o_2)$ is a randomly sampled 2-D offset pair. The maximum radius for generating offset pairs is optimized to 300 pixel meters for both x and y coordinates of o_1 and o_2 to provide more spatial context for the feature to represent the whole body posture. The classification outcome should be the same regardless of how far the pixels are from the imaging plane. Hence, the depth normalization factor $\frac{1}{d_I(P)}$ is applied to the generated offsets to ensure depth invariance.

For each input depth image, background pixels are set to the maximum depth of 10 m. Then, 2000 randomly sampled body pixels are used in training. For each of these pixels, the DCF is computed using a list of 2000 location offsets $\hat{\Theta}$ creating a feature vector v of 2000 DCF values per pixel. Location offsets are common for all pixels. Thus, the same 2-D location offset will have different feature responses depending on the given body pixel and the articulated posture, see Fig. 6. This setting allows the features to discriminate between different postures. The storage cost of an image feature vector is approximately 4 MB. The resulting pixel-wise feature vectors are used for training an RDF.

C. Training With Random Decision Forests

RDF [36] is an ensemble of decision tree predictors [37]. RDF has been proven fast and efficient for handling different data analysis problems such as classification, regression, clustering, and dimensionality reduction in [38]. It has a linear evaluation complexity and can be implemented on graphics processing units (GPUs) [39] providing real-time performance.

1) Training Procedure: Pixel-wise representations obtained previously using the DCF feature detector are used in learning a supervised RDF. Each tree in the forest is trained independently and with random parameter settings to ensure a high level of decorrelation between trees and hence improve forest prediction [36], [38], [40]. A decision tree is a collection of nodes that are organized hierarchically. Nodes are divided into split and leaf nodes. Each split node has an associated weak learner model acting as a simple binary classifier. During training, parameters of the split functions as well as leaf predictors are optimized according to a defined training objective function. For split node j, we select the record of parameter values that maximizes the information gain over incoming set of labeled pixel feature vectors as

$$I_j = H(S_j) - \sum_{i \in \{R,L\}} \frac{|S_j^i|}{|S_j|} * H(S_j) \quad (2)$$

where i indexes the right and left child nodes and $H(S_j)$ is the entropy of the incoming training set S_j. The entries of the optimized record are an offset pair and a threshold value.

Tree training stops when a defined stopping criterion is met. It could be a minimum information gain, maximum tree depth, or reaching a minimum number of training samples. In this work, we use maximum tree depth of 20 levels as stopping criteria. Finally, by the end of the training procedure, each leaf node of a tree t in a forest of size N stores a posterior distribution over a pixel label $p_t(c)$.

2) Batch RDF Training: Having the DCF detector applied to 2000 pixels using 2000 randomly sampled offsets creates approximately 4 MB of feature vectors per image. Therefore, as the training samples increase, the computational requirements become a challenge. We used the horizontal resource scaling approach to tackle this problem. Being a candidate problem for key-value pair formulation, suitable for batch processing, and constituted from totally independent records of feature vectors opened the door to use the MapReduce component of the Apache Hadoop distributed computing ecosystem [41] for training. The MapReduce framework decomposes complex computational problems into a set of simple MapReduce jobs, each of these jobs consists of a map, sort, and reduce phase. The load of processing these jobs is distributed and balanced among cluster members by means of a master node. A Hadoop cluster totaling 18 cores, 80 GB RAM, and 6 TB storage is used for the purpose of training. Large data files are stored on the Hadoop distributed file system (HDFS) divided into chunks of equal size to a fixed HDFS block size. A map task is created for processing each data block. Therefore, we maximize the block size to 4 GB to minimize the number of needed tasks and reduce the synchronization overhead. For more information about the theoretical and technical aspects of deploying and managing a Hadoop cluster as well as the MapReduce paradigm, resources [42] and [43] are highly recommended. The training procedure is decomposed into three MapReduce jobs such as in [26].

Using the MapReduce component of Hadoop is advantageous. First, Hadoop balances the computational load over a commodity of hardware. Second, horizontal scalability via adding extra nodes enables large-scale training. Most importantly, training the RDF is time consuming. Therefore, the fault tolerance capabilities for Hadoop are tremendously beneficial. Furthermore, the availability of powerful cloud solutions such as elastic MapReduce and IBM cloud softlayer makes MapReduce and Hadoop an adequate and flexible solution for large-scale RDF training.

3) Pose Recognition: The result of the RDF learning process is a set of (N) binary decision trees. Each leaf node stores a normalized histogram of labels reached that node during training. For evaluation, foreground segmentation is a critical preprocessing step that directly affects the performance of the posture recognition module as it relies on the evaluation of foreground pixels. In the proposed system, the background is modeled using the mode of a set of empty frames, no presence of subjects, as a calibration step. Then, the modeled background is subtracted from input depth frames. Further preprocessing operations (filtering, erosion, hole filling, and blob analysis) are applied to improve the quality of the foreground frames. Finally, the RDF is applied on foreground pixels generating pixel-wise votes for current postures.

An input test sample v is pushed simultaneously into all trees starting from the first split node until reaching its leaf. At each split node of a tree, a binary test is applied comparing the DCF outcome of the input sample against an optimized threshold. As soon as reaching a leaf node, the tree t outputs the conditional probability distribution $P_t(c|v)$ where c is a discrete label discriminating the human posture. Then, the produced probabilities are combined using the average ensemble model [38] to obtain the forest probabilistic estimate $P(c|v)$ for the pixel as

$$P(c|v) = \frac{1}{N_t}\sum_{t=1}^{N_t} P_t(c|v) \tag{3}$$

where v is the feature vector extracted for a foreground pixel in an input depth frame. The recognized posture is determined

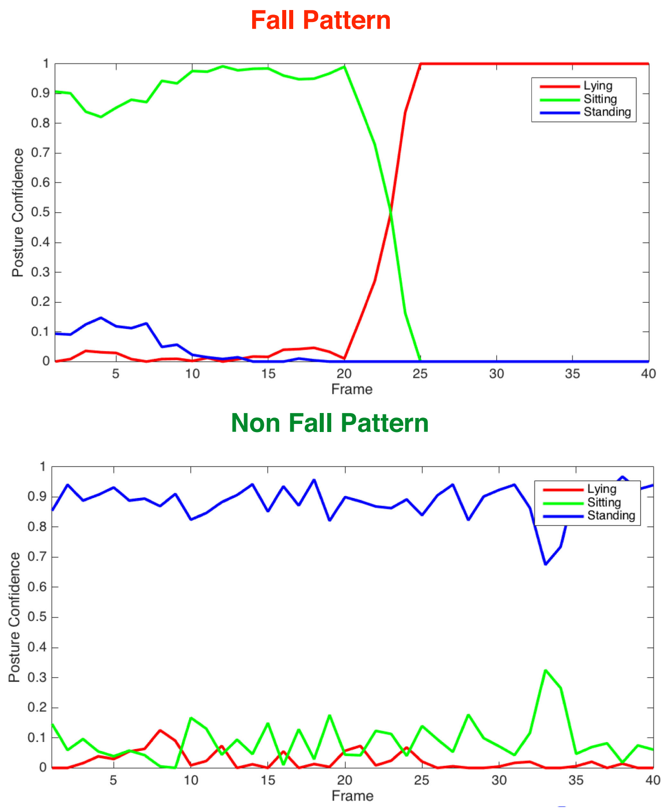

Fig. 7. Example of fall and nonfall patterns for the posture recognition module applied on sequences from the URFD dataset.

by the majority of pixel votes. In other words, we are reducing the classical pose estimation pipeline into a classification problem on a finite set of postures characterized by the application domain. The next section describes the use of the posture recognition outcome in detecting the occurrence of falls.

One strict requirement for our system is to ensure real-time performance. Therefore, we perform pixels evaluation on the GPU [39].

IV. Methods: SVM-Based Fall Detection

The fall is commonly defined as inadvertently coming to rest on the ground or other lower level excluding intentional change in position to rest on furniture, wall, or other objects [1]. This definition suggests that the fall is an event characterized by the speed of reaching the ground. This is the reason that joints tracking based fall detection methods require ground plane initialization.

In our approach, however, the fall is characterized as an abrupt change in lying posture confidence. The posture recognition module computes the lying posture confidence pattern for d consecutive frames. This pattern is used as an input feature vector for the SVM to decide the occurrence of a fall. The dimensionality d of the feature vector should be large enough to cover the start, rapid movement, and the end of the fall motion. Fig. 7 shows an example of fall and nonfall confidence patterns.

Detecting fall patterns is a binary classification problem. One of the most widely used classification algorithms is the SVM as it guarantees maximum-margin separation between classes [44]. In turn, this property yields good generalization performance with relatively little training data [38].

A. SVM Training

We used the UR fall detection dataset (URFD) [28] for training and evaluating the performance of the SVM classifier. Other fall detection datasets such as TSTv1 [45] and TSTv2 [46] are also available. However, the TSTv1 dataset was recorded using ceiling-mounted RGB-D sensors. The TSTv2 dataset was captured using a Kinect v2 sensor, which is not suitable for deployment on embedded devices due to its power and cooling requirements [8]. The latest comprehensive comparison of RGB-D datasets lists only URFD and TST datasets under the fall detection category [47].

The URFD dataset consists of 30 fall activities of two kinds: fall from standing and from sitting on a chair, 30 typical daily life activities (ADL) such as walking, sitting down, squatting, and picking-up an object, and 10 sequences with fall-like activities such as lying on a wooden sofa and lying on the floor. These activities were performed by five persons of different anthropometric measures. The sequences are accompanied with accelerometric data and were recorded using two Microsoft Kinect cameras, one of them is attached to the ceiling. The proposed solution does not rely on accelerometric data. The ceiling-mounted camera based fall detection is not supported in the current implementation. However, thanks to the data generation pipeline, we can easily synthesize a training dataset simulating a depth camera positioned at the ceiling and further extend the solution.

For the purpose of SVM training, the analysis of the URFD dataset suggested that the fall motion takes about 40 frames including an initial period, rapid movement, and resting [28]. Sequences of the dataset are segmented using a sliding window of 40 with stride of 1 to include variability to the start and ending periods. The resulting patterns are manually labeled and a subset of 500 fall patterns and 500 nonfall and fall-like patterns were randomly sampled and shuffled for SVM training, see Fig. 7. The trained SVM classifier is validated using tenfold cross validation achieving an average misclassification error of 0.01%.

B. Fall Confirmation

The trained SVM model acts as an online fall detector where a buffer of size $d = 40$ is used to hold the lying posture confidence over time. The buffer is periodically passed to the SVM for evaluation. Once the SVM detects a fall pattern, it is deactivated and a recover motion analysis algorithm is performed to authenticate the occurrence of a fall and avoid false alarms. The algorithm allows a recovery window of 20 frames, after which if the person remains in a lying posture, then the fall alarm is triggered. Otherwise, the SVM model is activated and no alarm is triggered.

V. Experiments and Results

We have trained a pixel-wise RDF to analyze the current human posture without relying on skeleton data. On top of that, an SVM classifier is used to analyze the change in lying posture confidence over time and indicate the occurrence of a fall. In this section, we evaluate the performance of the proposed fall detection system. First, the test datasets and evaluation criteria are described. Second, posture recognition experiments on examining the effect of various parameters such as forest size, number of tree levels, and size of the training dataset on the generalization capabilities are detailed. Third, the performance of the overall system is evaluated on the challenging URFD dataset and compared to other approaches. Finally, qualitative results are presented.

A. Test Datasets

We evaluate the performance of the proposed system modules using three test datasets: synthetic, real, and the aforementioned URFD dataset, see Section IV-A. The synthetic dataset contains 12 K images uniformly distributed among the three postures and rendered using camera angles from 0 to 360 with a step of 45^o, whereas the real test dataset contains 6 K frames recorded using the Kinect sensor in a noisy environment under low lighting conditions. The images were manually selected from different activities and labeled each with the respective posture. The posture recognition module is evaluated using the three test sets. None of the test images are included in training this module. The SVM module is trained and evaluated on the URFD dataset [28].

B. Evaluation Criteria

We calculate the average posture recognition accuracy over the synthetic and the real depth images and the average pixel-wise voting accuracy over the synthetic set. In addition to that, the fall sensors performance assessment criteria defined in [48] are used. Noury *et al.* [48] performed analysis on series of tests and concluded with proposing

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4}$$

measuring the capacity to recognize the considered occurrence properly, for instance, a fall motion or a lying posture, and

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{5}$$

determining the capacity to correctly detect nonfalls as two evaluation metrics, where

1) true positive (TP): correctly identifies the occurrence of a lying posture or a fall motion;
2) false positive (FP): false alarm indicates that a posture is misclassified as lying or and ADL activity is identified as a fall;
3) true negative (TN): no fall alarm on ADL events and standing or sitting postures;
4) false negative (FN): incorrect identification of a fall.

C. Posture Recognition: Parameter Analysis

In this section, we study the effect of several hyper-parameters on the generalization capabilities of the RDF model responsible for identifying the current posture in a single depth image.

TABLE I
EFFECT OF TRAINING DATASET SIZE

Dataset	Synthetic test set				Real test set		
	Avg pixel-wise acc	Posture accuracy	Sensitivity	Specificity	Posture accuracy	Sensitivity	Specificity
150	0.88	0.98	0.97	1.00	0.89	0.89	0.96
300	0.90	0.99	0.98	0.99	0.87	0.88	0.96
600	0.95	1.00	1.00	1.00	0.90	0.91	0.94
1200	0.94	0.99	0.99	1.00	0.90	0.91	0.95
1800	0.95	1.00	1.00	1.00	0.90	0.91	0.94
2400	0.96	1.00	1.00	1.00	0.90	0.91	0.95
3000	0.96	1.00	1.00	1.00	0.89	0.91	0.94

In order to determine the effective training dataset, a forest of size 4 trees of depth 20 levels is trained using balanced datasets of 150 to 3 K images. Each image contains 2000 training pixels creating a minimum size training set of 300 samples.

TABLE II
EFFECT OF FOREST SIZE

Forest size	Synthetic test set				Real test set		
	Avg pixel-wise acc	Posture accuracy	Sensitivity	Specificity	Posture accuracy	Sensitivity	Specificity
1	0.92	1.00	0.99	1.00	0.89	0.89	0.96
2	0.92	1.00	1.00	1.00	0.89	0.93	0.93
3	0.96	1.00	0.99	1.00	0.89	0.89	0.95
4	0.96	1.00	1.00	1.00	0.89	0.91	0.94
5	0.97	1.00	0.99	1.00	0.89	0.89	0.95

A forest of fixed tree depth of 20 levels is trained on 3 K samples 1 K per pose with varying number of trees. Increasing the forest size reduces the misclassification occurrences on the synthetic test set and shows stable posture recognition performance on the real test set.

TABLE III
EFFECT OF TREE DEPTH

Tree depth	Synthetic test set				Real test set		
	Avg pixel-wise acc	Posture accuracy	Sensitivity	Specificity	Posture accuracy	Sensitivity	Specificity
4	0.66	0.81	0.99	0.73	0.82	0.97	0.85
8	0.83	0.98	0.98	0.99	0.90	0.91	0.94
10	0.88	0.98	0.98	0.99	0.90	0.90	0.96
12	0.91	0.99	0.99	1.00	0.90	0.90	0.95
14	0.93	0.99	0.99	1.00	0.90	0.91	0.94
18	0.95	1.00	1.00	1.00	0.89	0.92	0.94
20	0.96	1.00	1.00	1.00	0.89	0.91	0.94

Results for a forest of size $T = 4$ trained using the same training dataset to increasing depth levels, up to 20. Although the results demonstrate a performance gain when using deeper trees, using trees with more than 20 levels causes performance degradation due to overfitting.

TABLE IV
RDF LYING POSTURE RECOGNITION ON THE URFD DATASET

Tree depth	Accuracy	Precision	Sensitivity	Specificity
4	0.90	0.71	0.99	0.86
8	0.96	0.87	0.99	0.95
10	0.97	0.89	0.99	0.96
12	0.97	0.89	0.99	0.96
14	0.96	0.87	0.99	0.95
18	0.97	0.89	0.99	0.96
20	0.97	0.90	0.99	0.96

The proposed holistic posture recognition method achieves full sensitivity on real unseen images of the URFD dataset. The precision is the ratio between true positives and the sum of both true and false positives.

TABLE V
EVALUATION AND COMPARISON OF SVM FALL DETECTION ON THE URFD DATASET

Method	Accuracy	Precision	Sensitivity	Specificity
SVM + Acc. [28]	0.94	0.88	1.00	0.90
Proposed	0.96	0.91	1.00	0.93

The proposed system achieves better results than the accelerometer-based system of Kwolek *et al.* [28].

The effect of forest size, tree depth, and the number of training images is investigated using both the synthetic and real test datasets. One parameter is changed at a time and unless otherwise specified, the default parameter settings are forest size of $T = 4$ trees trained to depth $D = 20$ on a balanced training set of 3 K synthetic images and using 2000 training pixels per image; hence, the default training dataset contains 6 M training samples. Each sample is represented using 2000 DCF features computed using randomly sampled offset pairs at a maximum pixel location offset of 300 pixel meters.

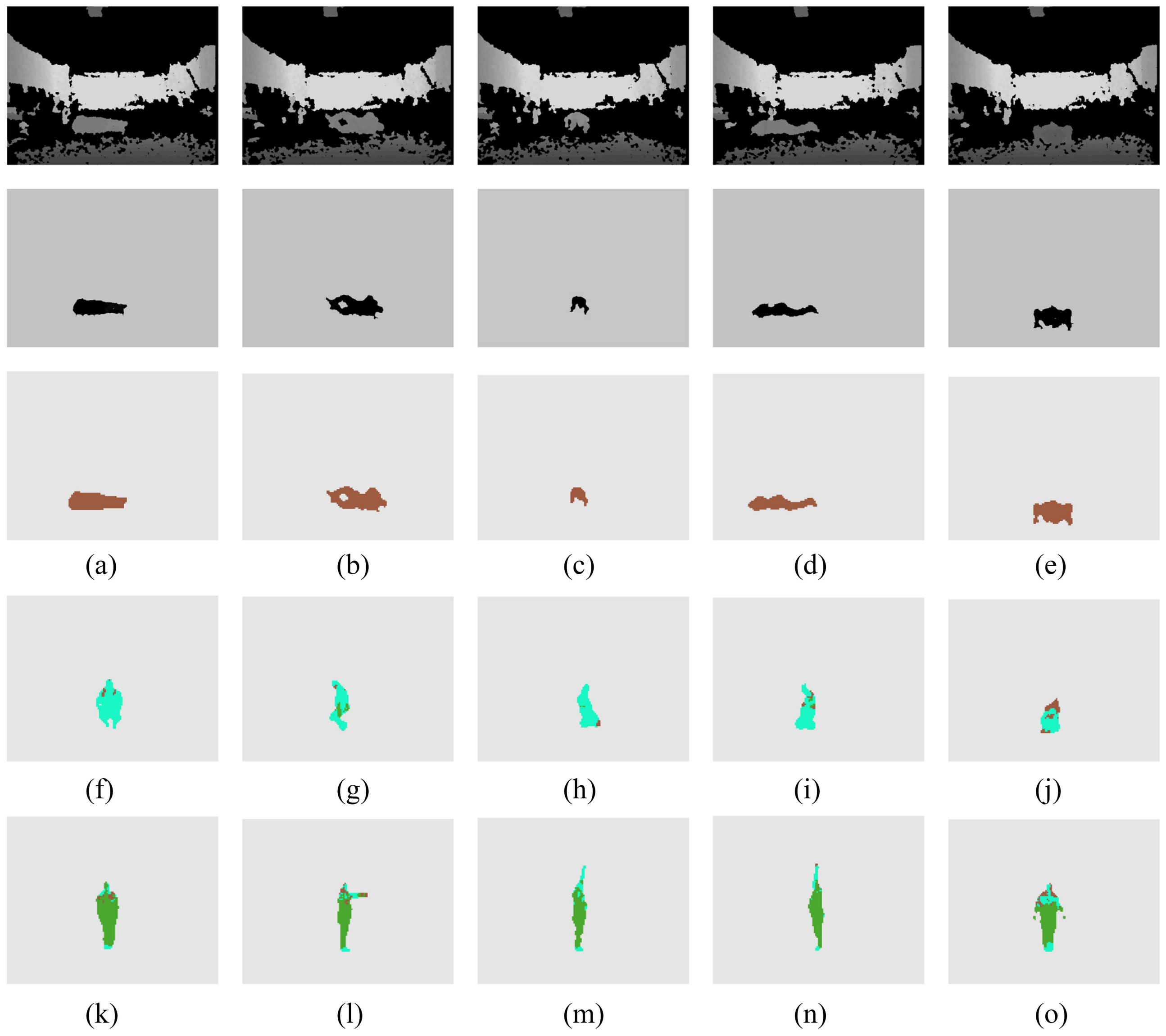

Fig. 8. Example of inferences on real test images. The first row shows samples of the depth images captured using the Kinect sensor. Foreground extraction via depth thresholding is applied and the resulting frames are shown in the second row. The next three rows display the recognition output of five samples from different angles for each posture. (a) Lying: 1:00, (b) lying: 1:00, (c) lying: 1:00, (d) lying: 1:00, (e) lying: 1:00, (f) sitting: 0:96, (g) sitting: 0:83, (h) sitting: 0:95, (i) sitting: 0:89, (j) stting: 0:65, (k) standing: 0:86, (l) standing: 0:72, (m) standing: 0:77, (n) standing: 0:83, and (o) standing: 0:63.

1) Effect of Training Dataset Size: Due to the challenges of the traditional data acquisition approach where people movements are captured using Kinect especially for fall scenarios, we extend the implementation of a data synthesis pipeline that works as a generative model for synthesizing high-quality depth frames accompanied with fully labeled ground truth frames. Generated images are customizable to suit any particular vision-based application, for instance, fall detection. However, it is not always about requiring massive data, generalization capabilities could be obtained using small and effective number of data points [44]. Therefore, several training experiments with varying number of training images are performed to identify the effective size for the training set. Table I shows the relationship between the amount of training data and the posture recognition performance. Training samples are equally distributed among all postures. Note the increase of the RDF model performance when growing the dataset up to 600 images (1.2 M training samples). The posture classification accuracy saturates at 600 images with slight changes between the decrease and the increase appearing on the different error metrics. This possibly happens due to reaching the model capacity of the trees and hence they stop learning.

2) Effect of Forest Size: Table II summarizes the effect of increasing number of trees T on the posture recognition module. A forest of deep trees $(D = 20)$, varying size $T = 1, 2, 3, 4, 5,$ is trained on 3 K depth images (1 K per pose). In this scenario, increasing the number of trees slightly improves the generalization performance on the synthetic test set and shows stable posture recognition behavior on the real test set. Also, having large forest size linearly increases the evaluation time and

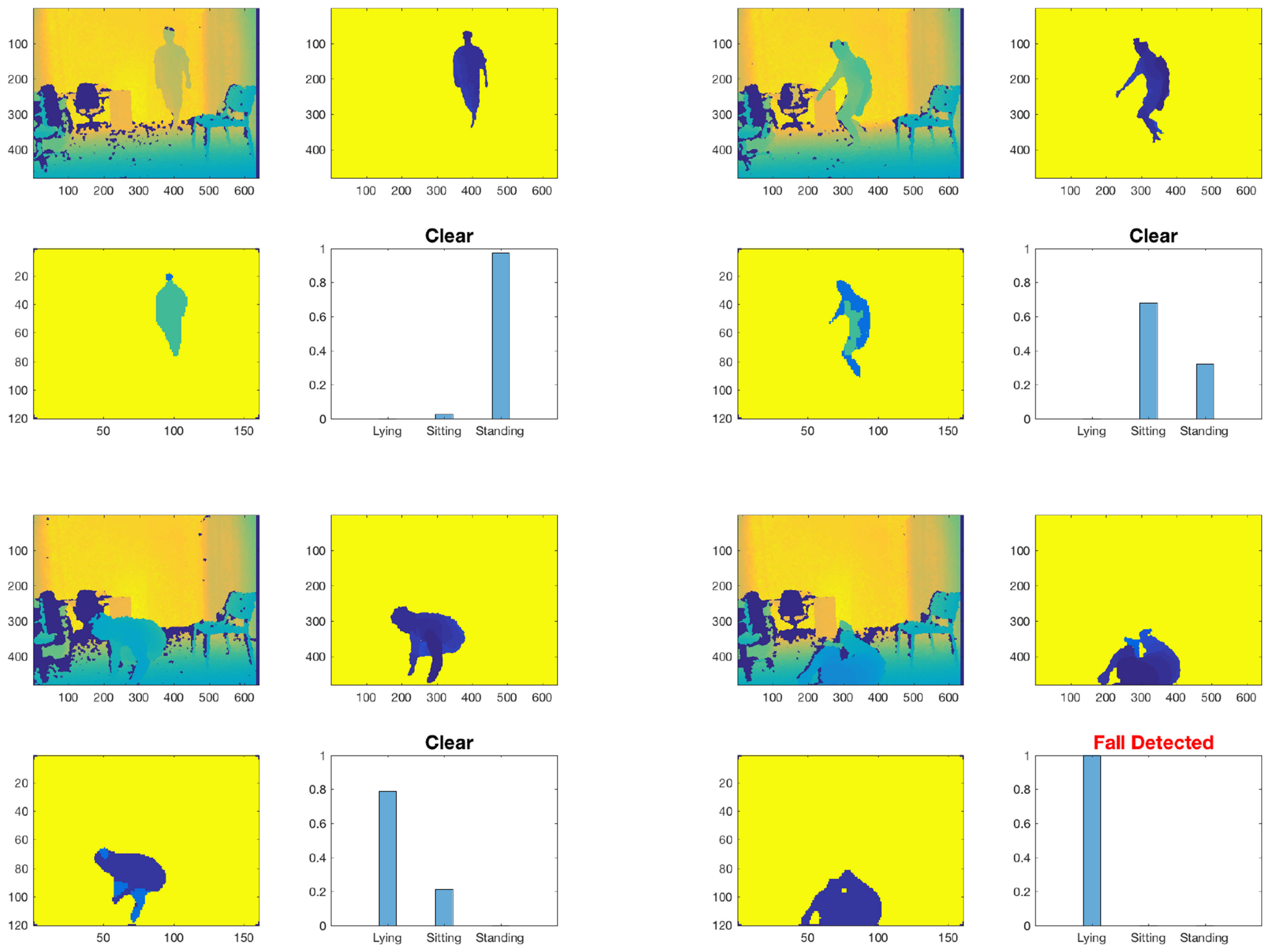

Fig. 9. Applying the proposed method on a fallen activity from the URFD dataset. For each frame, a foreground segmentation is performed as a preprocessing step. In this step, we perform background subtraction followed by (filtering, erosion, hole filling, and blob analysis) to improve the quality of the foreground frames. Then, the trained RDF posture recognition model is applied to label the foreground pixels. The current posture confidence is computed by counting pixel votes per posture. Meanwhile, the SVM module is monitoring the change of the lying posture confidence to detect the fallen actions.

complexity. Therefore, we have set the forest size to $T = 4$. This allows the proposed solution to be implemented easily within the memory capacity of most commercial GPUs.

3) Effect of Tree Depth: Table III reports evaluation results using different depth levels D of a forest of size $T = 4$ trained on 3 K samples (1 K per posture). We observe that increasing tree depth positively affects the overall generalization performance of the model. The results also demonstrate that tree depth has the most significant influence over the overall model performance as it directly affects the learning capacity of the model. However, at a certain limit, further increasing in depth $(D > 20)$ causes performance degradation due to overfitting.

D. Lying Posture Recognition: URFD Dataset

Table IV summarizes the performance of the RDF posture recognition model in identifying the lying postures in the URFD dataset. The ground truth frames are labeled with 1 for lying, 0 for transitional postures, and -1 as not lying. In [28], Kwolek and Kepski used a subset of 8459 images of which 2145 frames are lying and 6314 are sitting or standing postures. None of these images are included during forest training. Previous results in Section V-C3 indicated that tree depth is the most influential parameter on the generalization capabilities of the forest. Hence, we report the results using a forest of size $T = 3$ trees and different depth levels. The dataset features five subjects with different anthropometric measures from the training 3-D manikin. The reported results further demonstrate the generalization capabilities of the proposed posture recognition approach from synthetic training data to real test data and from a single subject to different subjects with various anthropometric measures.

E. Fall Detection Results[1]

The performance of the SVM classifier is evaluated using the URFD dataset. Kwolek and Kepski [28] constituted the dataset and kindly shared it for evaluation and comparison of fall detection methods. Their approach indicates an eventual fall through the thresholding of accelerometric signal and uses a lying posture classifier to authenticate the occurrence of the fall. Table V reports and compares the performance of the proposed method with the method described in [28]. The proposed system

[1][Online]. Available: http://ieeexplore.ieee.org

TABLE VI
EFFECT OF IMAGE RESOLUTION ON POSTURE RECOGNITION ACCURACY AND FRAME RATE

Resolution	Accuracy	Raspberry PI	CPU	NVidia GPU	INTEL GPU
640 × 480	1.00	0.48	1.75	41.72	30.84
320 × 240	1.00	2.01	5.25	56.70	42.28
160 × 120	0.99	7.46	8.62	67.25	57.82
80 × 60	0.98	21.0	10.0	88.67	84.33

We report the posture recognition accuracy and frame rate using different input resolutions and computing devices. All tests are conducted using a single tree of depth 20. At lower resolutions, the advantage of the Nvidia GPU is minimal due to the memory-to-GPU bottleneck.

achieves better results than the accelerometer-based system [28] in accuracy, precision, and specificity.

F. Qualitative Results

Fig. 8 shows the response of the holistic posture recognition method against a set of real frames, where the first row displays input depth frames captured using a single Kinect sensor for one subject with varying camera angles. The chosen subject has different anthropometric characteristics than the one used in generating the training data. The second row shows the resulting frames from the background subtraction preprocessing stage that are passed to the RDF model for pixel-wise classification. Four trees trained to depth 20 contribute in producing the last three rows of Fig. 8. The forest produces pixel-wise votes that are accumulated to compute the overall posture confidence. The proposed method does not rely on joint positions prediction, pose calibration, or underlying kinematic constraints. It depends on visible body pixels to identify the adopted posture even with the presence of self-occlusion and noisy environment, e.g., Fig. 8(c). However, the confidence decreases with the transitional postures, as shown in Fig. 8(j). Also, sitting on the ground has a low confidence due to the large similarity with the lying posture.

Fig. 9 demonstrates applying the proposed method on activities from the URFD dataset. It is divided into four subfigures for sampled frames from a fallen activity. As shown, the RDF model genaralizes from synthetic training images to unseen real images of a new subject with different anthropometric measures. The action recognition module analyzes the transitions between postures to identify the fallen actions.

G. Frame Rate

Table VI reports the effect of different image resolutions on the posture recognition accuracy and frame rate (FPS) of the proposed method. These results are evaluated on the synthetic test set using a single tree of depth 20. As reported, the posture recognition module maintains high classification accuracy with lower resolutions. The proposed method achieves up ten FPS on embedded devices with ARM architecture. At lower resolutions, the advantage of the Nvidia GPU is minimal due to the memory-to-GPU bottleneck. Note that the previously stated results are bounded with the frame rate provided by the Kinect sensor (30 FPS).

VI. CONCLUSION

In this paper, a vision-based fall detection system is presented. In the proposed method, input depth frames are passed through several processing modules. First, foreground segmentation is performed. Second, an RDF is used to identify the current articulated posture in the frame. The RDF is trained and evaluated using synthetic datasets to overcome the limitation of obtaining variate fall detection datasets. Third, an SVM classification model is used to analyze the change of the lying posture confidence overtime and indicate the occurrence of fall events. The proposed method meets the design goals identified in Section I and provides near unity sensitivity and specificity in the unintentional and inadvertent coming to rest on the ground situations, making it a fast and reliable solution for hospitals, homes, bathrooms, and laborious workplaces.

ACKNOWLEDGMENT

The motion files used in this research were obtained from mocap.cs.cmu.edu.

REFERENCES

[1] WHO, *WHO Global Report on Falls Prevention in Older Age*, 1st ed. Geneva, Switzerland: World Health Org., 2007.
[2] S. Deandrea, E. Lucenteforte, F. Bravi, R. Foschi, C. La Vecchia, and E. Negri, "Risk factors for falls in community-dwelling older people: A systematic review and meta-analysis," *Epidemiology*, vol. 21, pp. 658–668, 2010.
[3] L. Z. Rubenstein, "Falls in older people: Epidemiology, risk factors and strategies for prevention," *Age Ageing*, vol. 35, no. 2, pp. ii37–ii41, 2006.
[4] M. E. Tinetti, W. L. Liu, and E. B. Claus, "Predictors and prognosis of inability to get up after falls among elderly persons," *J. Amer. Med. Assoc.*, vol. 269, no. 1, pp. 65–70, 1993.
[5] J. Shotton *et al.*, "Efficient human pose estimation from single depth images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 12, pp. 2821–2840, Dec. 2013.
[6] D. Webster and O. Celik, "Systematic review of Kinect applications in elderly care and stroke rehabilitation," *J. NeuroEng. Rehabil.*, vol. 11, 2014, Art. no. 108.
[7] H. Haggag, M. Hossny, S. Haggag, S. Nahavandi, and D. Creighton, "Safety applications using Kinect technology," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, 2014, pp. 2164–2169.
[8] P. Fankhauser, M. Bloesch, D. Rodriguez, R. Kaestner, M. Hutter, and R. Siegwart, "Kinect v2 for mobile robot navigation: Evaluation and modeling," in *Proc. IEEE Int. Conf. Adv. Robot.*, 2015, pp. 388–394.
[9] M. Mubashir, L. Shao, and L. Seed, "A survey on fall detection: Principles and approaches," *Neurocomputing*, vol. 100, pp. 144–152, 2013.
[10] M. J. Mathie, A. C. F. Coster, N. H. Lovell, and B. G. Celler, "Accelerometry: Providing an integrated, practical method for long-term, ambulatory monitoring of human movement," *Physiol. Meas.*, vol. 25, no. 2, pp. R1–R20, 2004.
[11] G. Wu and S. Xue, "Portable preimpact fall detector with inertial sensors," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 16, no. 2, pp. 178–183, Apr. 2008.
[12] C.-F. Lai, S.-Y. Chang, H.-C. Chao, and Y.-M. Huang, "Detection of cognitive injured body region using multiple triaxial accelerometers for elderly falling," *IEEE Sens. J.*, vol. 11, no. 3, pp. 763–770, Mar. 2011.
[13] G. Demiris *et al.*, "Older adults' attitudes towards and perceptions of smart hometechnologies: A pilot study," *Med. Informat. Internet Med.*, vol. 29, no. 2, pp. 87–94, 2004.
[14] E. E. Stone and M. Skubic, "Fall detection in homes of older adults using the Microsoft Kinect," *IEEE J. Biomed. Health Informat.*, vol. 19, no. 1, pp. 290–301, Jan. 2015.

[15] Y. Li, K. C. Ho, and M. Popescu, "A microphone array system for automatic fall detection," *IEEE Trans. Biomed. Eng.*, vol. 59, no. 5, pp. 1291–1301, May 2012.
[16] Y. Li, K. C. Ho, and M. Popescu, "Efficient source separation algorithms for acoustic fall detection using a Microsoft Kinect," *IEEE Trans. Biomed. Eng.*, vol. 61, no. 3, pp. 745–755, Mar. 2014.
[17] X. Zhuang, J. Huang, G. Potamianos, and M. Hasegawa-Johnson, "Acoustic fall detection using Gaussian mixture models and GMM supervectors," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2009, pp. 69–72.
[18] M. Alwan *et al.*, "A smart and passive floor-vibration based fall detector for elderly," in *Proc. 2nd Int. Conf. Inf. Commun. Technol.*, vol. 1, 2006, pp. 3–7.
[19] H. Rimminen, J. Lindström, M. Linnavuo, and R. Sepponen, "Detection of falls among the elderly by a floor sensor using the electric near field," *IEEE Trans. Inf. Technol. Biomed.*, vol. 14, no. 6, pp. 1475–1476, Nov. 2010.
[20] Y. Zigel, D. Litvak, and I. Gannot, "A method for automatic fall detection of elderly people using floor vibrations and soundproof of concept on human mimicking doll falls," *IEEE Trans. Biomed. Eng.*, vol. 56, no. 12, pp. 2858–2867, Dec. 2009.
[21] D. Brulin, Y. Benezeth, and E. Courtial, "Posture recognition based on fuzzy logic for home monitoring of the elderly," *IEEE Trans. Inf. Technol. Biomed.*, vol. 16, no. 5, pp. 974–982, Sep. 2012.
[22] M. Yu, A. Rhuma, S. Naqvi, L. Wang, and J. Chambers, "Posture recognition based fall detection system for monitoring an elderly person in a smart home environment," *IEEE Trans. Inf. Technol. Biomed.*, vol. 16, no. 6, pp. 1274–1286, Nov. 2012.
[23] M. Yu, Y. Yu, A. Rhuma, S. M. R. Naqvi, L. Wang, and J. A. Chambers, "An online one class support vector machine-based person-specific fall detection system for monitoring an elderly individual in a room environment," *IEEE J. Biomed. Health Informat.*, vol. 17, no. 6, pp. 1002–1014, Nov. 2013.
[24] C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau, "Robust video surveillance for fall detection based on human shape deformation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 5, pp. 611–622, May 2011.
[25] C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau, "3D head tracking for fall detection using a single calibrated camera," *Image Vision Comput.*, vol. 31, pp. 246–254, 2013.
[26] K. Buys, C. Cagniart, A. Baksheev, T. De Laet, J. De Schutter, and C. Pantofaru, "An adaptable system for RGB-D based human body detection and pose estimation," *J. Vis. Commun. Image Represent.*, vol. 25, no. 1, pp. 39–52, 2014.
[27] Z. Bian, J. Hou, L. Chau, and N. Magnenat-Thalmann, "Fall detection based on body part tracking using a depth camera," *IEEE J. Biomed. Health Informat.*, vol. 19, no. 2, pp. 430–439, Mar. 2015.
[28] B. Kwolek and M. Kepski, "Improving fall detection by the use of depth sensor and accelerometer," *Neurocomputing*, vol. 168, pp. 637–645, 2015.
[29] P. Loncomilla, C. Tapia, O. Daud, and J. Ruiz-del Solar, "A novel methodology for assessing the fall risk using low-cost and off-the-shelf devices," *IEEE Trans. Human-Mach. Syst.*, vol. 44, no. 3, pp. 406–415, Jun. 2014.
[30] X. Ma, H. Wang, B. Xue, M. Zhou, B. Ji, and Y. Li, "Depth-based human fall detection via shape features and improved extreme learning machine," *IEEE J. Biomed. Health Informat.*, vol. 18, no. 6, pp. 1915–1922, Nov. 2014.
[31] G. Mastorakis and D. Makris, "Fall detection system using Kinects infrared sensor," *J. Real-Time Image Process.*, vol. 9, pp. 635–646, 2014.
[32] L. Yang, B. Yang, H. Dong, and A. E. Saddik, "3-D markerless tracking of human gait by geometric trilateration of multiple Kinects," *IEEE Syst. J.*, vol. 12, no. 2, pp. 1393–1403, Jun. 2018.
[33] H. Haggag, M. Hossny, D. Filippidis, D. Creighton, S. Nahavandi, and V. Puri, "Measuring depth accuracy in RGBD cameras," in *Proc. 7th IEEE Int. Conf. Signal Process. Commun. Syst.*, 2013, pp. 1–7.
[34] M. A. Fischler and R. A. Elschlager, "The representation and matching of pictorial structures," *IEEE Trans. Comput.*, vol. C-22, no. 1, pp. 67–92, Jan. 1973.
[35] "CMU Graphics Lab Motion Capture Database." [Online]. Available: http://mocap.cs.cmu.edu.
[36] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, pp. 5–32, 2001.
[37] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, pp. 81–106, 1986.
[38] A. Criminisi, "Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning," *Found. Trends Comput. Graph. Vision*, vol. 7, pp. 81–227, 2011.
[39] T. Sharp, "Implementing decision trees and forests on a GPU," in *Proc. Eur. Conf. Comput. Vision*, 2008, pp. 595–608.
[40] Y. Amit and D. Geman, "Shape quantization and recognition with randomized trees," *Neural Comput.*, vol. 9, pp. 1545–1588, 1997.
[41] "Apache Hadoop." [Online]. Available: https://hadoop.apache.org
[42] T. White, *Hadoop: The Definitive Guide*. Newton, MA, USA: O'Reilly Media, Inc., 2012.
[43] J. Lin and C. Dyer, *Data-Intensive Text Processing With MapReduce*, ser. Synthesis Lectures on Human Language Technologies, vol. 3. San Rafael, CA, USA: Morgan and Claypool, 2010, pp. 1–177.
[44] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. Cambridge, MA, USA: MIT Press, 2012.
[45] S. Gasparrini, E. Cippitelli, S. Spinsante, and E. Gambi, "A depth-based fall detection system using a Kinect sensor," *Sensors*, vol. 14, no. 2, pp. 2756–2755, 2014. [Online]. Available: http://www.mdpi.com/1424–8220/14/2/2756.
[46] S. Gasparrini *et al.*, "Proposal and experimental evaluation of fall detection solution based on wearable and depth data fusion," in *ICT Innovations 2015*, ser. Advances in Intelligent Systems and Computing, vol. 399, S. Loshkovska and S. Koceski, Eds. New York, NY, USA: Springer, 2016, pp. 99–108.
[47] M. Firman, "RGBD datasets: Past, present and future," in *Proc. IEEE Conf. Comput. Vision Pattern Recog. Workshops*, 2016, pp. 19–31.
[48] N. Noury *et al.*, "Fall detection—Principles and methods," in *Proc. 29th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc.*, 2007, pp. 1663–1666.

Ahmed Abobakr (GS'17) received the B.Sc. degree in computer science from Cairo University, Giza, Egypt. He is currently working toward the Ph.D. degree at the Institute for Intelligent Systems Research and Innovation, Deakin University, Geelong, VIC, Australia. His Ph.D. focuses on human posture analysis using deep representation learning and depth imaging technologies.

His research interests include computer vision, machine learning, deep learning, and deep generative models.

Mohammed Hossny (A'10–M'10) received the bachelor's degree in computer science from Cairo University, Giza, Egypt, the master's degree from Cairo University in collaboration with the IBM Center of Advance Studies, Cairo, Egypt, and the Ph.D. degree from the Institute for Intelligent Systems and Research Innovation, Deakin University, Geelong, VIC, Australia. He developed an algebraic framework for multimodal image fusion as part of his Ph.D. degree.

His research interest focuses on human performance analysis. He is approaching this from markerless motion capture and biomechanics perspectives by harnessing the power of deep convolutional neural networks.

Saeid Nahavandi (M'92–SM'07) received the Ph.D. degree from Durham University, Durham, U.K., in 1991.

He is an Alfred Deakin Professor, a Pro Vice-Chancellor (Defence Technologies), a Chair of Engineering, and the Director with the Institute for Intelligent Systems Research and Innovation, Deakin University, Geelong, VIC, Australia. He has authored or co-authored more than 600 papers in various international journals and conferences. His research interests include modeling of complex systems, robotics, and haptics.

Dr. Nahavandi is a Fellow of Engineers Australia and the Institution of Engineering and Technology. He is the Co-Editor-in-Chief for the IEEE SYSTEMS JOURNAL, an Associate Editor for the IEEE/ASME TRANSACTIONS ON MECHATRONICS, an Associate Editor for the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS, and an IEEE ACCESS Editorial Board member.