Fall Detection in Homes of Older Adults Using the Microsoft Kinect

Erik E. Stone, *Member, IEEE*, and Marjorie Skubic, *Member, IEEE*

***Abstract*—A method for detecting falls in the homes of older adults using the Microsoft Kinect and a two-stage fall detection system is presented. The first stage of the detection system characterizes a person's vertical state in individual depth image frames, and then segments on ground events from the vertical state time series obtained by tracking the person over time. The second stage uses an ensemble of decision trees to compute a confidence that a fall preceded on a ground event. Evaluation was conducted in the actual homes of older adults, using a combined nine years of continuous data collected in 13 apartments. The dataset includes 454 falls, 445 falls performed by trained stunt actors and nine naturally occurring resident falls. The extensive data collection allows for characterization of system performance under real-world conditions to a degree that has not been shown in other studies. Cross validation results are included for standing, sitting, and lying down positions, near (within 4 m) versus far fall locations, and occluded versus not occluded fallers. The method is compared against five state-of-the-art fall detection algorithms and significantly better results are achieved.**

***Index Terms*—Fall detection, kinect, older adults.**

I. Introduction

FALLS are a major issue among older adults. It is estimated that one out of every three older adults (those age 65 and over) falls each year [1]. Of those who fall, many suffer serious injuries, such as hip fractures and head traumas, which reduce their mobility and independence, and lead to an increased risk of early death [1]. The direct medical cost of falls among older adults in the U.S. in the year 2000 was more than $19 billion [2]. This cost does not account for the decreased quality of life and other long term effects experienced by many after suffering a fall. Studies have also found an increased risk of physical and physiological complications associated with prolonged periods of lying on the floor following a fall, due to an inability to get up [3]. Older adults living alone are at great risk of delayed assistance following a fall. A low-cost, unobtrusive system capable of automatically detecting falls in the homes of older adults could help significantly reduce the incidence of delayed assistance after a fall.

Many methods have been investigated or commercially developed for reporting or detecting falls among older adults. These include wearable devices that allow an individual to manually push a button in the event of fall, and wearable devices that automatically detect a fall using sensors such as accelerometers [4]–[7]. However, wearable devices must be continuously worn, require batteries to be recharged, and may simply be forgotten by the user. In the case of devices requiring action on the part of the wearer, loss of consciousness following a fall would prevent use. Studies have also indicated older adults' preference for nonwearable sensors [8].

A number of researchers have looked at the use of environmentally mounted sensors for fall detection, such as floor vibration sensors [9], [10], passive infrared sensors [11], acoustic sensors [10], [12], and video-based sensors, including traditional cameras [13]–[18] and depth imaging sensors [19]–[23]. Studies have found that privacy concerns of older adults to vision-based monitoring systems may be addressed by the use of appropriate privacy preserving processing techniques, such as silhouettes [24].

This paper presents an unobtrusive method for detecting falls in the homes of older adults using an environmentally mounted depth imaging sensor, namely the Microsoft Kinect. A two-stage system is used to detect falls. Initially, 3-D foreground objects are segmented from each depth image frame using a dynamic background subtraction algorithm. The first stage of the system characterizes the vertical state of a 3-D object for an individual frame, and then segments on ground events from the vertical state time series. The second stage utilizes an ensemble of decision trees and a set of features extracted from an on ground event to generate a confidence that a fall preceded it.

For the purposes of training and evaluation, data were collected (as part of an IRB approved study) in 13 apartments with a total of 16 residents over the course of 1 year. The dataset, consisting of approximately 3 339 days of continuous data, contains 454 falls, including 445 falls performed by trained stunt actors, and nine naturally occurring resident falls. The extensive data collection allows for characterization of system performance under real-world conditions to a degree that has not been shown in other published studies.

The remainder of this paper is organized as follows. First, a discussion of similar work is presented. Second, a brief overview of the Kinect-based system installed in the homes of older adults is given. Third, our methodology for detecting falls using the depth data from the Kinect is described. Fourth, results are presented and compared against those of an earlier system reported

Manuscript received November 8, 2013; revised February 20, 2014; accepted March 10, 2014. Date of publication March 17, 2014; date of current version December 30, 2014. This work was supported in part by the U.S. National Science Foundation under Grant CNS-0931607 and by the Agency for Healthcare Research and Quality under Grant R01-HS018477.

The authors are with the Center for Eldercare and Rehabilitation Technology, Department of Electrical and Computer Engineering, University of Missouri, Columbia, MO 65211, USA (e-mail: stoneee@missouri.edu; skubicm@missouri.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JBHI.2014.2312180

in the literature. Finally, there is a discussion of key points, limitations, and future work.

II. Related Work

Much work has been done investigating the use of standard imaging sensors for fall detection. Approaches have ranged from single cameras mounted on the wall [13], [14], to cameras mounted on the ceiling [15], [16], to multiple cameras placed around a room to allow 3-D reconstruction of foreground objects [17], [18]. Single camera systems typically rely on image space features extracted from silhouettes, such as bounding box ratios. Multicamera systems instead extract features such as velocity from 3-D objects constructed from back projecting multiple silhouettes. However, traditional camera-based systems suffer from a number of limitations.

First, foreground extraction generally relies on background modeling in color image space, which is difficult in real-world conditions due to issues such as lighting changes and shadows [25]. Second, operation in low light or no light conditions is only possible if an active source of infrared (IR) light is installed along with the cameras. However, color information is no longer available under only IR illumination, posing another issue for background modeling [26]. Third, in the case of single camera systems, it is difficult to extract features that accurately capture the 3-D movement of objects necessary to robustly characterize falls [14], [23]. Fourth, in the case of multicamera systems, installation and calibration of the cameras in the same reference frame becomes a major concern, along with cost as the system complexity rises. Due to these issues, among others, researchers have looked at using the recently released Microsoft Kinect depth imaging sensor to detect falls. The Kinect uses a pattern of actively emitted IR light to create a depth image, where the value of a pixel is dependent on the distance to what it is viewing.

In [19], the authors present a method for fall detection using a Kinect and a wearable device. The OpenNI framework [27] is used to obtain the center of gravity of an individual from the Kinect. The CoG is then used as input for fall detection, along with acceleration and angular velocity from the wearable device. All falls performed by three volunteers were correctly detected by the system, and no false alarms were triggered when using both the Kinect and the wearable device.

A Kinect placed 30 cm above the floor is used to detect falls in [20]. A manual presegmentation is used to initially identify areas where a fall could occur, and spatial characteristics of segmented objects are then used to identify when a person is lying on the floor. Out of 55 falls collected in a laboratory setting, 93% were classified correctly.

A method for the detection of walking falls using the Kinect is presented in [21]. The width, height, and depth of a 3-D bounding box of a person, obtained using the OpenNI framework, along with the first derivatives, are used for fall detection. The algorithm was evaluated on a set of 48 falls and 112 nonfalls collected in a laboratory setting. All falls were detected with no false alarms.

In [22], the authors present a method for fall detection using the centroid height, and, in the event of occlusion, body velocity of objects obtained from the Kinect depth data using a background subtraction algorithm. The method was evaluated on approximately 15 min of data containing 25 falls and 54 nonfalls collected in a laboratory setting. Only one fall was not detected and no false alarms were reported.

Finally, researchers describe a method for fall detection utilizing the skeletal model from the Kinect SDK in [23]. The major orientation of an individual's body, along with the height of their spine, is computed and used as input for fall detection. The method was evaluated on a set of 40 falls and 32 nonfalls collected in a laboratory setting. Before the removal of tracking errors, 37 falls were detected with five false alarms. A similar approach, reported in [28], first uses fuzzy logic to generate on ground, in between, and upright state confidences from the body orientation and spine height features. The confidences are then thresholded for fall detection. This method was evaluated on a set of 40 falls and 32 nonfalls collected in a laboratory setting. Accuracy of 98.6% was reported with one false alarm.

Although the results are encouraging, the major limitation of these previous studies is a lack of evaluation in real-world settings. Given that these systems were evaluated on an hour or less of total data, collected in a laboratory setting, it is impossible to know how they will perform in the intended setting of an older adult's home. Generally, systems evaluated on controlled laboratory data perform poorly in complex, real-world environments, as the wide range of difficulties such environments pose make it nearly impossible to collect a realistic set of data in a laboratory setting. These issues are further illustrated in Section VI, where the method proposed here is compared against those in [17], [21]–[23], and [28], on the extensive real-world dataset collected in this paper.

III. System Overview

A Kinect sensor and a computer were deployed in the homes of elderly residents at an independent living facility as part of an IRB approved human subjects study. Fig. 1 shows the Kinect sensor installed in one apartment. The Kinect is placed on a small shelf a few inches below the ceiling (height 2.75 m), above the front door. The computer is placed in a cabinet above the refrigerator. This arrangement has proven to be unobtrusive to the residents, with most indicating that they do not notice the equipment after a short period of time.

The Microsoft Kinect SDK and the skeletal tracking it provides are not used. Instead, the raw values from the Kinect depth stream (obtained using the open source libfreenect library [29]) are processed directly. The main reason for not using the Kinect SDK is the limited range of the skeletal tracking, approximately 1.5 to 4 m from the Kinect. This range is insufficient to capture falls in many areas of the apartments. Given that residents may be in any position or orientation prior to a fall, it is also likely that the Kinect SDK may fail to acquire a skeletal model in some cases, or be unreliable during the fall motion [21].

A dynamic background subtraction algorithm is used to identify foreground pixels from the depth imagery of the Kinect. The algorithm is based on the well-known mixture of distributions approach in [30]; however, each distribution is a simple range.

Fig. 1. *Top:* Kinect sensor and computer (inside cabinet) as installed in apartments. *Middle:* Example depth images and extracted foreground from an apartment. *Bottom:* Three-dimensional object formed using extracted foreground.

Each pixel is modeled using three background distributions, and three foreground distributions, and the background distributions are initialized using a set of training frames (10 s). Given a new pixel value, any distribution to which the new value matches has its range updated and weight increased; nonmatched distributions have their weight decayed (and become inactive when their weight reaches zero). If the new value does not match any active distribution, the foreground distribution with the least weight (or inactive) is reinitialized based on this value. After updating, any foreground distribution whose weight has reached a predefined threshold replaces the least weight (or inactive) background distribution. The parameter settings are such that a stationary object placed in the scene will be updated into the background after approximately 5 min, and background distributions will become inactive if not matched for 20 min. Due to the depth imagery using an actively emitted pattern of infrared light, many of the problems, such as lighting and shadows, associated with background modeling in color imagery are avoided (although other issues arise, such as overpowering sunlight).

After postprocessing (filtering, dilation, hole filling, and erosion), foreground pixels are projected into 3-D with respect to the Kinect, and then translated into a world-based coordinate system using an automated estimate of the floor plane that is updated in real time. Next, segmentation of foreground objects is performed by projecting each point onto discretized (1 in^2) X/Y (floor) and X/Z (vertical) planes and using single-linkage clustering to group points on each plane. Points that are grouped together on both planes are combined to form a 3-D object. A basic tracking algorithm is used to track objects, of at least a minimum size, across frames. At each frame, the location and trajectory of the currently tracked objects is used to assign them (greedily) to objects segmented from the current frame. The distance between the predicted centroid location of a currently tracked object and an object in the current frame must be below a threshold (60 cm) for an assignment to be made. Objects in the current frame which do not match to a tracked object are added as tracked objects themselves. Tracked objects are removed after 20 s without being matched.

IV. Fall Detection Methodology

A two-stage system is used for fall detection. The first stage of the system characterizes the vertical state of a segmented 3-D object for individual frames, and then identifies *on ground events* through temporal segmentation of the vertical state time series of tracked 3-D objects. The second stage of the system utilizes an ensemble of decision trees and features extracted from an *on ground event* to compute a confidence that a fall preceded it.

A. First Stage—Vertical State Characterization

Three features are used to robustly characterize the vertical state of a 3-D object for an individual frame: the maximum height of the object $Z_{\max}$, the height of the object's centroid Z_{cent}, and the number of elements of the discretized (1 inch squares) X/Y (floor) plane to which only points from the object with a height below 38 cm (15 in, three fourths typical knee height) project Z_{pg}. The following equation is used to obtain the measure of vertical state V_s:

$$V_s = \left(\frac{Z_{\max}}{k_{\max}} + \frac{Z_{\text{cent}}}{k_{\text{cent}}} - \frac{Z_{pg}}{k_{pg}} \right) \tag{1}$$

where $k_{\max}$ is set to 170 cm (estimated average height of all adults in the USA [31]), k_{cent} is set to 85 cm (half of $k_{\max}$), and k_{pg} is learned from training data as the average value of Z_{pg} following all falls in the training set. During evaluation, this value ranged from 350 to 391 over 13 cross validation folds.

Thus, values in the ranges of 1.6 to 2.2, and 0.9 to 1.4, could be expected when a person is standing, or sitting, respectively; while a near zero, or less, could generally be expected when a person is on the ground following a fall.

B. First Stage—On Ground Event Segmentation

Given vertical state estimates for a tracked 3-D object over time, *on ground events* are identified through temporal segmentation of the vertical state time series. Accurate segmentation of the fall motion is critical for robust feature extraction.

Depending on the type of fall, the environment, and the individual, among other variables, the duration of the fall motion will vary. Such variation can create problems if a fixed window size is used for feature extraction.

After filtering, the vertical state time series using median and average filters (with a window size covering half a second) to yield the smoothed signal $V_{\text{avg}}(i)$, $i = 1, \ldots, N$, the following procedure is used to segment *on ground events*:

INPUT: $V_{avg}(i)$, $i = 1,\ldots,N$
OUTPUT: E, list of *on ground events* (e.g., t_{fall}, t_{init}, t_{start}, t_{end})
PARAMETERS: T_{trig}, f_{rate}

```
i = 1
WHILE i < N
    IF V_avg(i) < T_trig

        t_start = t_init = i
        WHILE t_start < N AND V_avg(t_start+1) < V_avg(t_start)-ε
            t_start = t_start + 1
        END
        t_fall = t_start
        WHILE t_fall > max(1, t_start – 4*f_rate) AND V_avg(t_fall-1) > V_avg(t_fall)
            t_fall = t_fall – 1
        END
        t_end = t_start
        WHILE t_end < N AND V_avg(t_end) < T_trig
            t_end = t_end + 1
        END

        Add on ground event (t_fall, t_init, t_start, t_end) to list E

        i = t_end + 1
    ELSE
        i = i + 1
    END
END
```

where f_{rate} is the frame rate, and T_{trig} is a threshold learned from training data as the value required to trigger *on ground event* extraction (if possible) for all falls, plus 5%. During evaluation, this value ranged from 0.876 to 0.903 over 13 cross validation folds. To allow real-time response to fall events, if the end of an *on ground event* t_{end} is not identified within 4 s of the start t_{start} fall confidence is computed at that time, denoted t_{fend}, before t_{end} is identified. Thus, the time elapsed from t_{fall} to t_{fend} is at most 8 s. The point t_{init}, which denotes when *on ground event* extraction was triggered, is included for illustrative purposes.

Fig. 2 shows example sets of signals, with segmentation overlaid, from one stunt actor standing fall, one stunt actor sitting fall, and one naturally occurring resident fall. Finally, to address the issue of objects blending into the floor in the depth imagery from the Kinect (and thus no longer being tracked), if the time elapsed from t_{start} to the last time an object was tracked is less than 4 s, the last observation is artificially repeated to meet this condition. This allows falls to be detected at larger distances from the Kinect, where the faller may not be distinguishable from the floor in the depth imagery following the fall.

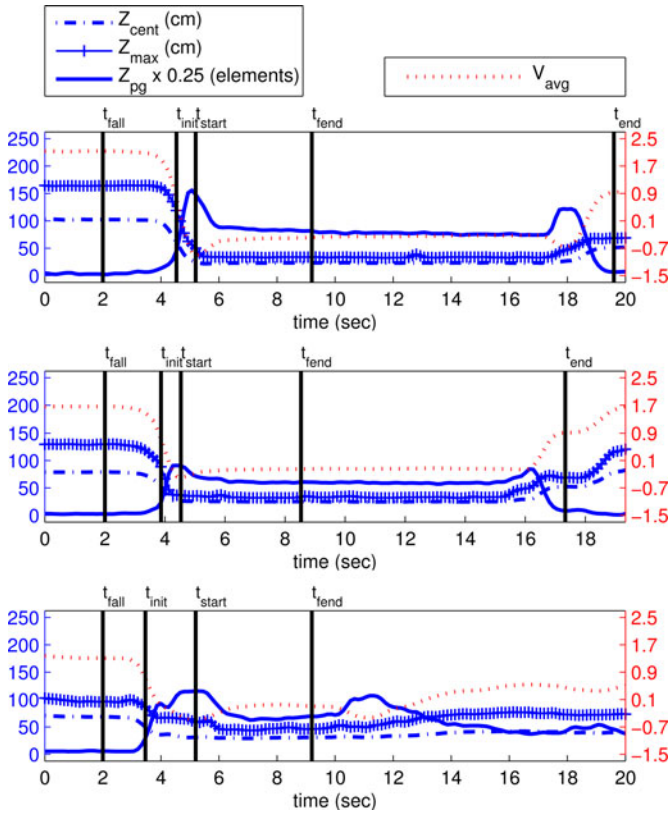

Fig. 2. Signals and event segmentation from stunt actor standing (top) and sitting (middle) falls, as well as a naturally occurring resident fall (bottom). Before the fall, the resident bent down to pick something up, and then fell backward.

C. Second Stage—On Ground Event Features

Five features are extracted for each *on ground event* for the purpose of computing a confidence that a fall preceded it. A description of these features follows.

1) Minimum Vertical Velocity (MVV): Minimum vertical velocity is computed as the minimum of V_s' (the derivative of the vertical state time series) from t_{fall} to t_{start}. This feature characterizes the downward vertical motion of falling to the ground.

2) Maximum Vertical Acceleration (MVA): Maximum vertical acceleration is computed as the maximum of V_s'' (the second derivative of the vertical state time series) from the frame of minimum velocity to t_{start}. This feature characterizes the action of abruptly hitting the floor, stopping downward vertical motion.

3) Mean V_{avg}: Mean V_{avg} is computed as the mean of V_{avg} from t_{start} to t_{fend}. A higher value is indicative of bad foreground segmentation, or nonfall activities such as bending over or kneeling.

4) Occlusion Adjusted Change in Z_{pg} (Δ_{pg}): When objects in the environment move, whatever the object was previously occluding may be identified as foreground depending on a variety of factors that influence the background model.Δ_{pg} measures the change in Z_{pg} from t_{fall} to t_{start}, accounting for the possible impact of occlusion. The presence of occlusion is assessed for each pixel, at each frame, using the change in measured depth.

If the rate of change between the current and previous frame exceeds 420 cm/s (selected as a value significantly faster than a person would ever be moving in their home), it is assumed that whatever the pixel is currently viewing was occluded by a different, closer object in the prior frame.

Given this assessment of occlusion, the number of foreground pixels belonging to a 3-D object that have a height (after projection to world 3-D) below 38 cm and were previously occluded p_{occlude} can be counted at each frame, along with the total number of pixels belonging to the object that have a height below 38 cm p_{ground}. The percentage of p_{ground} pixels appearing due to occlusion from t_{fall} to t_{start} is approximated as

$$r_{\text{occlude}} = \min\left(\frac{\sum_{i=t_{\text{fall}}}^{t_{\text{start}}} P_{\text{occlude}}(i)}{\max_i p_{\text{ground}}(i) - \min_i p_{\text{ground}}(i)}, 1\right). \tag{2}$$

Finally, Δ_{pg} is computed as

$$\Delta_{pg} = \left(\max_i Z_{pg}(i) - \min_i Z_{pg}(i)\right)(1 - r_{\text{occlude}}). \tag{3}$$

A near zero value of Δ_{pg} implies there was no significant change in Z_{pg} that was not a result of occlusion. A higher value is indicative of an object moving from a more off the ground position to a more on the ground position.

5) Minimum Frame-to-Frame Vertical Velocity (MFFVV): Minimum frame-to-frame vertical velocity is an estimate of the MVV from t_{fall} to t_{start} computed by mapping foreground pixels in a depth image frame, to foreground pixels in the previous depth image frame, independent of the tracking algorithm. At each frame, a pointwise matching is found (after projection to 3-D) between all the foreground pixels in the current frame and all the foreground pixels in the prior frame such that the overall change in 3-D position is minimized. A simple heuristic and random assignment is used to initialize the matching. The matching is then iteratively refined using a neighborhood guided search, where neighborhood structure is obtained from image space. A many-to-one mapping of current to previous foreground pixels is allowed as needed. After refinement, the vertical movement of a foreground object between the previous and current frame is estimated as the median of the vertical movement of all the pixels belonging to the object. MFFVV is computed as the minimum of these frame-to-frame vertical velocity estimates for an object from t_{fall} to t_{start}.

In the absence of tracking errors, MFFVV will likely not be as accurate as MVV derived from the vertical state time series. However, MVV will reflect any vertical motion inferred by the tracking algorithm, whether that motion is correct, or the result of a tracking error. MFFVV, meanwhile, is computed from vertical velocity estimates made at each frame independent of the tracking algorithm. Thus, MFFVV will generally be more robust when the scene contains multiple foreground objects (or foreground segmentation errors) that are in close proximity.

D. Second Stage—Ensemble for Fall Confidence

A fall confidence is computed for each *on ground event* using the five extracted features and an ensemble of decision trees. Each tree is a binary tree in which the decisions are based on a single, randomly selected, predictor (dimension). The tree is built recursively, in a top-down fashion, with the optimization criterion for selecting split cut points being the mean-squared error (MSE) of predictions versus the training data. Leaf nodes are recursively split until a node contains fewer than 10 observations, a split would create a child node with fewer than five observations, or the MSE for the node's predictions drops below a threshold ε. After the tree is built, the predicted value of each leaf node is the average target value of the training observations assigned to the node.

Each decision tree is trained using a sample of the training data. A large imbalance, roughly 1:400, of positive (fall) to negative (nonfall) samples exists in the collected data (described in Section V). Thus, each sample of the training data contains all of the minority class examples (falls) and a randomly selected, with replacement, undersampling of the majority class examples (nonfalls). Additive Gaussian noise (zero mean with standard deviation equal to 5% of that present in the training data positive samples), or jitter, is added to each training sample to further improve generalization. The output of the trees is combined by averaging.

E. Ground Truth Matching

Matching of *on ground events* (with temporal segmentation $t_{\text{fall}}, t_{\text{start}}, t_{\text{fend}}, t_{\text{end}}$) to ground truth, a set of times, $G = \{g_1, \ldots, g_n\}$, for each known fall, is assessed by the condition

$$t_{\text{fall}} - \varepsilon \leq g_i \leq t_{\text{end}} + \varepsilon \tag{4}$$

where ε is 2 s. If true, for exactly one ground truth entry, then the *on ground event* is said to match known fall i. The time of each known fall was identified, based on review of the depth imagery, as the time at which the faller was on the ground and vertical movement due to the fall had ceased.

V. Evaluation

Kinect systems were deployed in 13 apartments as part of an ongoing IRB approved study. The apartments are located in an independent living facility for older adults. Ages of the 16 residents ranged from 67 to 97. Nine were female and seven were male. In total, 3 339 days of continuous data collected from the apartments, containing 454 falls of varying types, were used for analysis. In addition to the residents' activity, that of all pets and visitors, including cleaning and clinical staff, is present in the data.

To keep the space required to store the data to a manageable size, the depth imagery was stored at 7.5 frames per second, with a resolution of 320 $\times$ 240 using a lossless video codec. Following 20 s without motion, empty frames (encoded using no data beyond the header information) were written to the video stream until motion was again detected. Thus, when there was no activity in view of the Kinect (such as when residents were sleeping, out of the apartment, or seated relatively still) virtually no disk space was used. Motion was detected by thresholding the average pixel difference between consecutive frames (excluding pixels for which a depth measurement in both frames was not available). When processing the data, empty frames

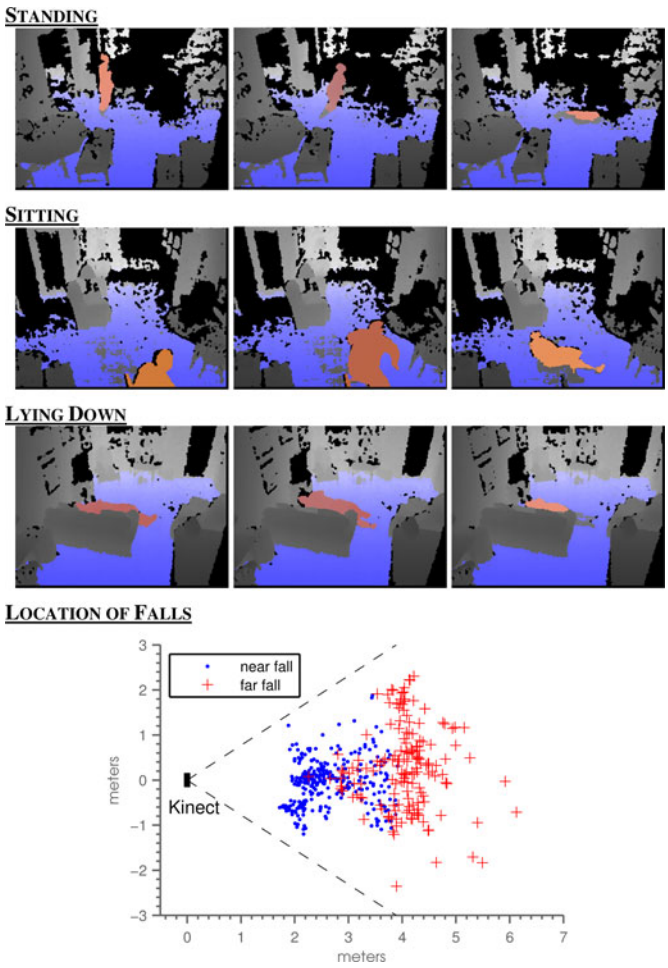

Fig. 3. *Top:* Sample depth imagery from stunt actor falls performed in apartments of older adults. Pixels identified as ground are shaded blue, while pixels identified as foreground are shaded red. Black pixels indicate no depth value was returned by the sensor. *Bottom:* Centroid location of faller (with respect to Kinect) following the fall motion for the ground truth falls. Blue dots are falls classified as near, while red pluses are falls classified as far. The classification is based on the maximum distance of the faller over the entire fall motion, not just the position of the faller following the fall motion.

were replaced by the last nonempty frame. In total, 80 939 h of video, requiring 3.44 TB of disk space, was recorded and used for evaluation.

Each month approximately four stunt actor falls [32] were performed in each apartment resulting in a total of 445 stunt actor falls. The stunt actor falls are categorized into three major types: standing, sitting, and lying down. Within each major type, additional variation is present. Standing falls comprise 14 subtypes, ranging from sudden loss of balance falls to tripping falls. Sitting falls comprise five subtypes, and lying down falls comprise two subtypes. In addition to falls, the stunt actors also performed motions designed to trigger false alarms while in the apartments, such as slowly lying down on the floor, stretching on the floor, and bending over to pick up objects. Example depth imagery of stunt actor standing, sitting, and lying down falls, with ground plane estimate (blue) and extracted foreground (red) overlaid, is shown in Fig. 3, along with a plot showing the location of the collected falls with respect to the Kinect.

Nine naturally occurring resident falls, from four residents, are also included in the captured data. These naturally occurring falls were identified using two methods. First, incident reports, filled out by clinical staff members after responding to a fall, or after having a resident report a fall, were manually cross-checked with the Kinect depth imagery to see if the falls occurred in view of the Kinect. Seven of the nine naturally occurring falls were identified by this method. Second, after initial testing, on ground events detected as false alarms when operating at a false alarm rate of four false alarms per month were manually examined. Two of the nine naturally occurring falls was identified as a result of this examination. Despite best efforts to track resident falls in the apartments, it is possible that other naturally occurring falls exist in the data. Of the nine naturally occurring falls, seven were standing falls, while two were sitting falls.

Each fall was assigned one of four contexts, based on the distance of the faller from the Kinect sensor and whether significant occlusion was present during the fall. A fall was considered to be far from the sensor if the faller was beyond 4.0 m (the current range of skeletal tracking in the Microsoft SDK) from the Kinect at any time during the fall motion (from t_{fall} to t_{fend}). Otherwise, the fall was considered near to the sensor. Significant occlusion was considered present if the faller could not be tracked for the duration of the fall; generally due to either beginning the fall out of view of the Kinect, or ending the fall out of view of the Kinect. This could be the result of being outside the field of view, or being occluded behind structures or furniture in the environment. Partial occlusion of the faller, which did not prevent tracking for more than a few frames, and fallers blending into the floor or walls, were not considered significant occlusion.

Of the seven naturally occurring standing falls, four were classified as far from the Kinect, while three were classified as near. None were significantly occluded. Of the two naturally occurring sitting falls, one was classified as near, while one was classified as far. Both suffered from significant occlusion. In the first case, the resident fell from her chair behind a table. In the second, the resident tipped over in his recliner behind a kitchen counter.

Both resubstitution and cross validation were used to evaluate the performance of the system. In the case of resubstitution, an ensemble of 200 decision trees was both trained and tested on data from all 13 apartments. In the case of cross validation, the data from one apartment were removed, an ensemble of 200 decision trees was trained on the data from the remaining apartments, and, finally, the resulting ensemble was evaluated on the data from the left out apartment. This process was repeated for all 13 apartments.

Fig. 4 shows cross validation and resubstitution results for each of the three major fall types under each of the four contexts. Each line represents the mean of 20 trials and error bars extend from the minimum to the maximum result. The range for the false alarm rate (x-axis) is set to a maximum of four false alarms per month, approximately one per week. This range was selected because higher false alarms rates would most likely not be tolerated by users of the system, and/or lead to users

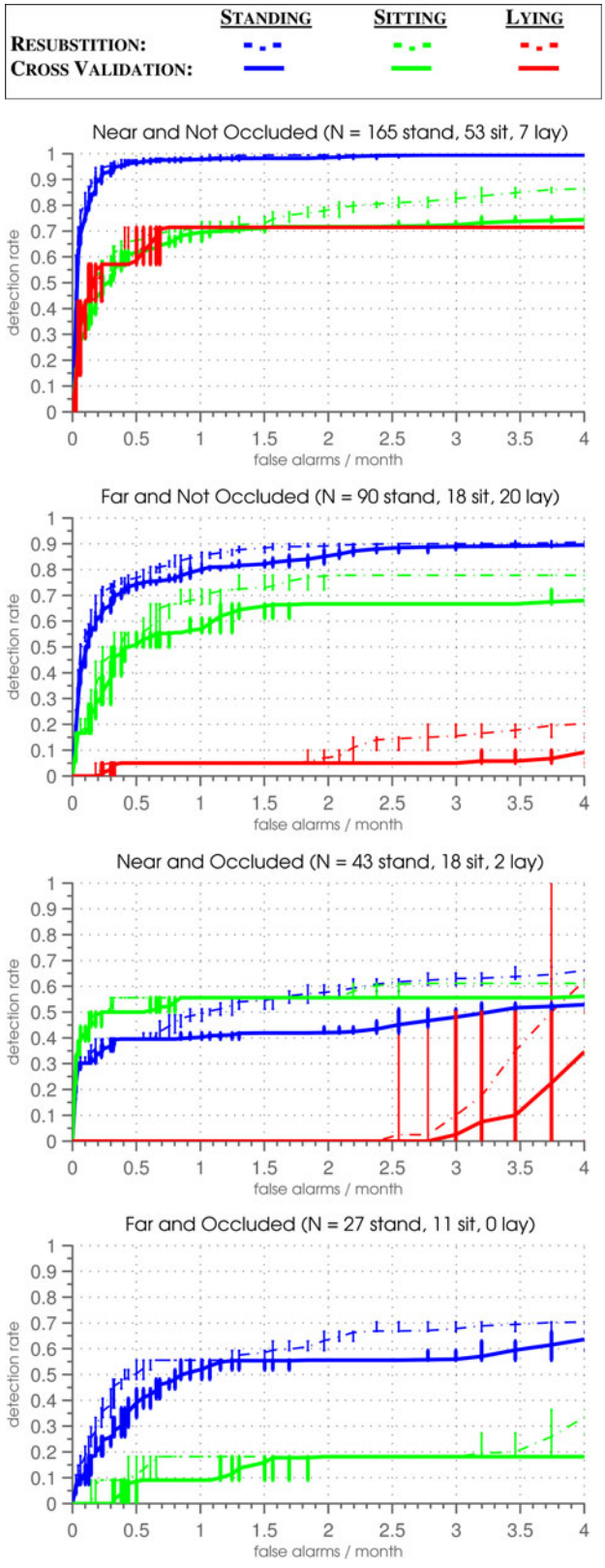

Fig. 4. Performance curves showing resubstitution and cross validation results on standing, sitting, and lying down falls under each of four contexts. Lines represent the mean of 20 trials, with vertical bars extending from the minimum to the maximum result.

ignoring the alarms. In total, 175 667 on ground events were segmented by the first stage of the system. After matching the events to ground truth, 24 out of the 454 ground truth falls failed to generate an on ground event. Eleven of those falls suffered significant occlusion. Of the remaining 13, all were located far from the Kinect, with eight being lying down falls. The system can achieve (cross validation) 98%, 70%, and 71% detection of standing, sitting, and lying down falls, respectively, while incurring one false alarm per month, when the falls are near to the sensor and not significantly occluded. The results at one false alarm per month are essentially unchanged under resubstitution.

As shown in Fig. 4, significant occlusion of the faller greatly reduces fall detection performance. Performance is also reduced, to a lesser extent, when falls are far (greater than 4.0 m) from the sensor, with detection rates at one false alarm per month (cross validation) dropping to 79%, 58%, and 5%, for standing, sitting, and lying down falls, respectively.

The three naturally occurring standing falls near to the Kinect would have been detected with a maximum false alarm rate of 0.15 per month, over all 20 cross validation trials. Three of the four naturally occurring standing falls far from the Kinect would have been detected with a maximum false alarm rate of 4.6 per month, over all 20 cross validation trials. The remaining naturally occurring standing fall, which occurred in front of a window 5.5 m from the Kinect, went completely undetected by the system, as sunlight prevented depth measurements from being returned for a large majority of the pixels viewing the resident. The two naturally occurring sitting falls, which both suffered from significant occlusion, did trigger extraction of on ground events. However, they would have only been detected as falls if the system were operating at a false alarm rate in excess of 100 per month.

To assess the relative strength of the on ground event features for fall detection, an ensemble of 40 decision trees was trained (under cross validation) on each individual feature. The top of Fig. 5 shows performance curves for these ensembles on the near and not occluded standing falls. MFFVV, mean V_{avg}, and Δ_{pg} are relatively stronger features, while MVV and MVA, computed from the vertical state time series, are relatively weaker. The performance achieved with the individual features is significantly worse than that achieved using them in combination.

Finally, the parameter T_{trig} (which ranged from 0.876 to 0.903 during cross validation) was varied from –0.3 to 1.7 to assess the impact on performance. Results on the near and not occluded standing falls are shown at the bottom of Fig. 5. Although the number of segmented on ground events varies significantly (from 2 882 to 476 430), fall detection performance is very similar for all values above –0.3. When T_{trig} is set this low, it causes a number of the falls to not be segmented.

VI. Comparison With State of the Art

A comparison with five state-of-the-art fall detection algorithms recently reported in the literature [17], [21]–[23], [28] was performed using the 165 near and not occluded standing falls described in Section V. For the algorithms in [21]–[23],

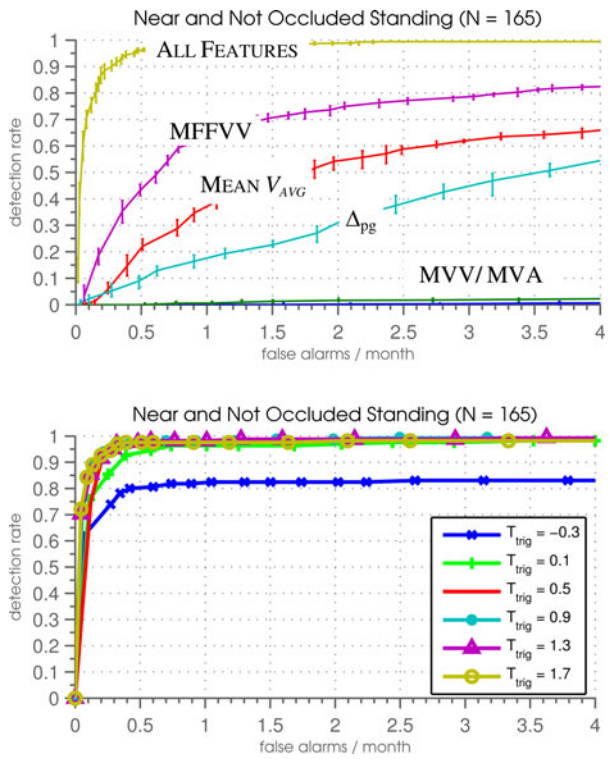

Fig. 5. *Top: On ground event* individual feature comparison. Performance curves show cross validation results on near and not occluded standing falls. Curves are the mean of ten trials, with bars extending from the minimum to maximum result. *Bottom:* Impact of variation in parameter T_{trig} on performance on near and not occluded standing falls. T_{trig} was varied by over 100% from the values learned during cross validation (0.876–0.903).

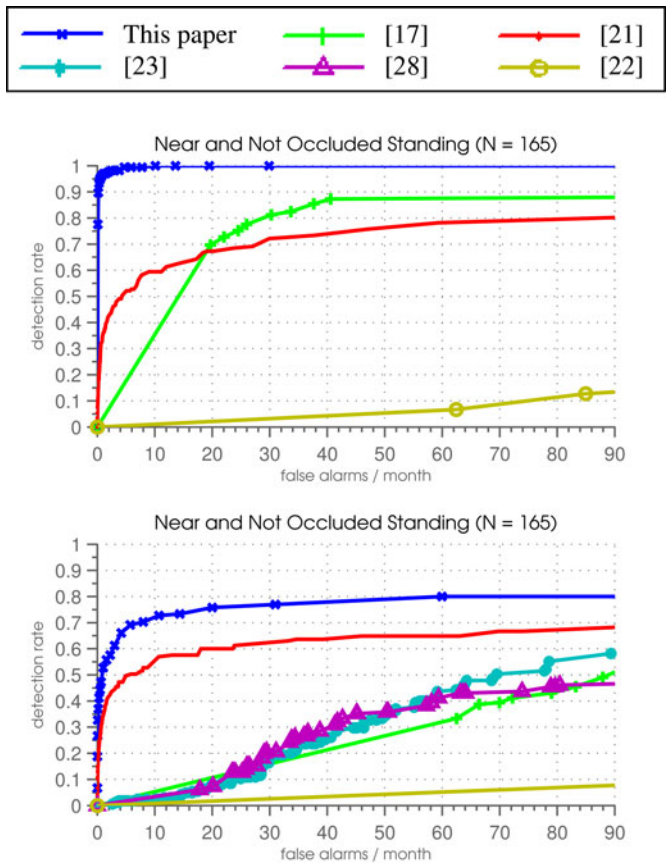

Fig. 6. Comparison with five state-of-the-art algorithms on near and not occluded standing falls (see Section V). Legend entries refer to the reference number of the publication in which the algorithm was reported. *Top:* Results using the segmentation and tracking system described in this paper (see Section III). *Bottom:* Results using OpenNI system for segmentation and tracking. Algorithms [23] and [31] could only be evaluated using the OpenNI system.

[28], all the positive samples were used to set the range for a dense grid-based sampling of the parameter space. Parameter combinations yielding the highest detection rate for a given false alarm rate were retained to obtain maximal performance curves. Additionally, a 2% hysteresis was added to all thresholds (in the absence of any other instructions) to prevent rapid successive detections when signals were near the threshold values, limiting false alarms. Cross validation, as described in Section V, was used to adjust the parameters of the system proposed in this work. Results are shown in Fig. 6.

The algorithms, except for those in [23] and [28] which require skeletal joint data, were evaluated using both the output of the segmentation and tracking method described in this paper (see Section III), and the output of the open source OpenNI framework [27] using the PrimeSense NITE Natural Interaction Middleware [33] (both version 1.5). OpenNI, in combination with the proprietary NITE middleware, is one of two major software packages (along with the Kinect SDK) that provide user segmentation, tracking, and skeletal joint locations from Kinect depth data. The studies in [21] and [28] used this software for segmentation and tracking. Although the depth data used for the analysis were not captured with OpenNI, OpenNI provides a mechanism for processing the data. However, no mechanism currently exists to allow processing of the data with the Kinect SDK.

The NITE middleware, like the Kinect SDK, is designed for game play and gesture recognition purposes. Recommendations suggest having the sensor positioned 1 to 2 m off the ground, with users standing upright in front of the sensor at a distance of 1.5 to 4.0 m, in an open area. It is specifically stated that being close to walls, objects, or other people are difficult situations that can produce inaccuracies [33]. Thus, it was not designed for continuous monitoring in a typical older adult's home, where residents and visitors are frequently near, in contact with, or partially occluded by furniture and other objects. Furthermore, it was not designed for use with a depth camera placed near the ceiling. However, it has successfully been used for this purpose in laboratory settings in other studies [28].

Recent versions of the NITE middleware (1.5 and above) include automatic skeletal tracking initialization. They do not require users to enter a calibration pose as was the case with earlier versions. Initial testing with the depth data in this paper found the automatic calibration procedure rarely completed successfully, causing skeletal joint data to be unavailable a vast majority of the time. To address this issue, a saved skeletal model (of a 167 cm individual) was used to initialize skeletal tracking for all users. Although this is not ideal, it allowed skeletal joint data to be available a large majority of the time.

Despite the 165 falls not suffering from significant occlusion as defined in Section V, the OpenNI system loses track of the faller in at least one frame during 46 of the falls, and loses track of the faller completely while the faller is approaching and on the ground during 30 of the falls. This is likely due to the more sophisticated scene segmentation performed by the NITE middleware. Ideally, this allows moved objects, such as chairs and tables, to immediately be adapted into the background while maintaining user tracking. However, when users enter poses which the system is not familiar with, or are in close proximity to other objects in the scene, they may be adapted into the background.

The OpenNI system also suffers from other tracking issues when analyzing the data. Chairs, tables, bookcases, parts of walls, and pets are often identified as users after a person moves through or interacts with the scene. In addition, for a number of the falls when tracking is only lost for a few frames, the faller is identified as a new user when tracking resumes. This causes the fall motion to be split among identified users. Lastly, skeletal joint data are only available when the faller is on the ground for 113 of the 165 falls. Thus, the OpenNI system effectively imposes a ceiling of between 68% and 82% on the detection rate depending on what features are used by an algorithm, and how it detects falls.

The algorithm in [22] uses centroid height to detect when a person is on the ground, and, if occlusion (loss of silhouette) is detected, body velocity prior to the occlusion. Initial testing found the body velocity component of the algorithm, using the threshold indicated in the paper (0.63 m/s), triggered a huge number of false alarms when people walked out of the scene; on the order of 20 per day. As such, the body velocity component was disabled for this comparison. Using centroid height alone, the algorithm still triggers a large number of false alarms at low detection rates. False alarms are caused by a variety of everyday occurrences, including pets moving on the floor, items dropped or moved on the floor (such as blankets, pillows, bags, foot rest, etc.), and residents and visitors lying or playing on the floor.

The algorithm in [23] uses spine height and a measure of body orientation derived from skeletal joint data to detect when a person is on the ground. This algorithm performs significantly better than the algorithm in [22], but still suffers from a large number of false alarms due to many of the same issues. It is also limited by the fact that the required skeletal joint data are only available when the faller is on the ground for 113 of the 165 falls. It could not be evaluated using the segmentation and tracking system described in this paper, given the need for skeletal joint data.

The algorithm in [28] is similar to the algorithm in [23]. In this version, spine height and body orientation are used as inputs to a fuzzy logic system. The fuzzy logic system then generates confidences of whether a person is in an upright, in between, or on the ground state. Finally, on the ground and upright confidences are thresholded to detect falls. The performance of this algorithm is very similar to that of the algorithm in [23], in which spine height and body orientation are thresholded directly. Here again, as the algorithm requires skeletal joint data, it could not be evaluated using the segmentation and tracking system described in this paper.

The algorithm in [21] uses the rate of change of the height, width, and depth of the 3-D bounding box of a person to detect falls. It requires the height of the bounding box to be shrinking faster than a threshold t_{vH} while a combined measure of width and depth is expanding faster than a threshold t_{vWD}, for N consecutive frames to initiate fall detection. Detection is then completed if the bounding box height velocity stays below another threshold for 2 s, indicating inactivity. This algorithm performs significantly better than the previous methods as it attempts to detect the fall motion. Thus, many of the false alarms detected by the previous algorithms are eliminated. However, this method still suffers from many false alarms itself.

One example is a resident walking up to a counter that is between the resident and the Kinect and grabbing an item on the counter. Given the positioning of the Kinect, the resident is initially not occluded as they start walking toward the counter. However, as the resident approaches the counter, the resident's legs become occluded, causing the bounding box height to shrink. Simultaneously, the resident extends their arms and reaches out to grab the item on the counter, causing the combined width and depth measure to expand. The resident then remains at the counter, keeping the bounding box height steady, which causes the fall detection to complete. Furthermore, due to partial occlusion and segmentation issues, roughly 15% of the falls do not obey the underlying assumption of the algorithm; namely, that the combined width and depth measure will expand noticeably during the fall. The 3-D bounding box is also sensitive to limb placement. If a person's arms are extended at the beginning of a fall, and are drawn in during the fall motion, this can contribute to the combined width and depth measure not exhibiting the expected behavior.

The algorithm in [17] is a two-stage fuzzy logic-based approach incorporating features such as centroid height, centroid velocity, and Eigen-based height, along with domain knowledge from nurses. It first identifies intervals when a person is thought to be on the ground, and then uses the acceleration of the centroid during a 10 s window prior to the on ground interval to compute a confidence that an impact occurred. These features are then used to compute a fall confidence. With the tracking system described in this paper, the algorithm has comparable performance to that in [21], and yields better results than the algorithm in [22]. Performance of this algorithm is significantly reduced using the OpenNI system. The main reason for this is discussed later.

The major limitation of the algorithm in [17] is the impact confidence measure used to characterize the fall motion. This measure, which is the ratio of peak acceleration from each half of the 10 s window preceding the interval, assesses whether there is a relative spike in acceleration prior to the on ground interval. The presence of a relative spike in acceleration is insufficient to distinguish many nonfall motions from fall motions. It is also susceptible to missed detections if an abrupt motion, such as quickly standing up, takes place prior to the fall. The need for acceleration information prior to the fall is also problematic if a

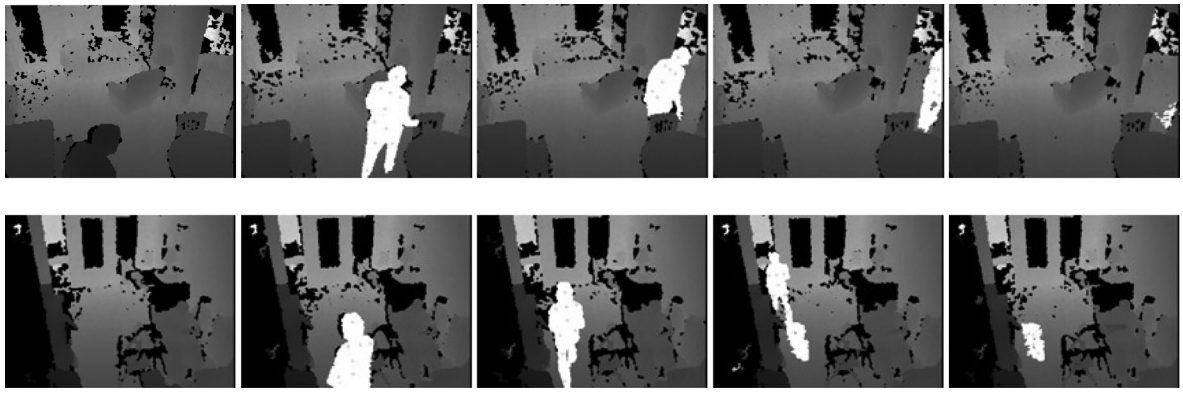

Fig. 7. Issues with OpenNI system, likely resulting from poor floor plane estimation. *Top:* Section of floor is identified as the user (with shoulder, neck, torso, and hip joints) after person exits the scene. *Bottom:* Same behavior in a different apartment. (White pixels indicate foreground segmented by the OpenNI system. Green dots indicate skeletal joint locations with a confidence greater than 0.5.)

fall occurs immediately, or very soon after, a person enters the scene.

The method proposed here does considerably better than the other algorithms using the segmentation and tracking system described in Section III. It also performs better than the other algorithms using the OpenNI system, although performance is worse than expected (even after accounting for the previously mentioned issues with the OpenNI system) at false alarm rates below 10 per month. This is largely due to poor floor plane estimation. The OpenNI system frequently generates floor plane estimates that are below or above the floor level by up to 12 cm in some areas (or the entirety) of an apartment. This leads to significant variability in the Z_{pg} signal (within and between apartments), and impacts the fall segmentation as well as 4 of the 5 features used to compute fall confidence.

Inaccurate floor plane estimates are also likely the cause of issues such as those shown in Fig. 7. In these cases, a section of floor is identified as the user when a user exits the scene. These cases, along with others caused by the inaccurate floor plane estimation, lead to increased false alarms for the method proposed here, and those in [17], [22], [23], and [28]. However, as the 3-D bounding box typically does not expand in width and depth when the height shrinks in these cases, and height from the floor plane is not itself a feature, they have little impact on the performance of the algorithm in [21].

VII. Discussion

A method for fall detection in the homes of older adults using the Microsoft Kinect was presented. An extensive data collection consisting of 9 years of continuous data from actual apartments of older adults containing 445 falls performed by trained stunt actors, along with nine naturally occurring resident falls, allowed characterization of performance in real-world conditions under four different contexts. The method was compared against five state-of-the-art algorithms previously reported in the literature and significantly better results were achieved.

The major issues faced when detecting falls at larger distances from the Kinect include decreased resolution of the faller in the depth image, which makes foreground segmentation more difficult, decreased precision of the pixel depth estimates, and an increased likelihood of natural light overpowering the actively emitted infrared pattern of the Kinect causing pixel depth estimates not to be returned. A future version of the Kinect, with an increased depth image resolution and accuracy, would likely improve the detection of falls far from the sensor. However, the issue of sunlight overpowering the actively emitted infrared pattern would still be a major problem at increased distances.

Reduced performance on nonoccluded sitting and lying down falls, as compared to nonoccluded standing falls, is due to a number of factors. With regards to sitting falls, the major issue is the presence of the chair used by the faller. Often, part of or the entire chair will be identified as foreground as a result the fall, and will become part of the segmented 3-D object representing the person. Although the use of the Zpg signal helps make the system robust to segmentation issues, this will, of course, impact calculation of the features used for fall detection. Additionally, when the chair is positioned between the faller and the Kinect, the chair may occlude part of the faller (not enough to be considered significant occlusion), which, again, will impact calculation of the features used for fall detection.

With regards to lying down falls, the major issue is the significantly reduced fall motion. All lying down falls were performed from couches, with the faller's height being approximately 75 cm at the beginning of the fall. Consequently, the fall motion covers less than half the vertical distance of a standing fall, and the peak downward velocity is significantly reduced. This reduced motion, along with the faller possibly blending into the couch, makes it more difficult to discriminate lying down falls from nonfalls, especially at large distances from the Kinect.

The majority of false alarms detected by the system at low false alarm rates (less than 4 per month) represent significant challenges. Examples include a maid dropping a large bag of trash on the floor, visiting children dropping to the floor while playing, certain pets jumping down from furniture, and visitors (such as family and maintenance staff) sitting or lying down on the floor quickly. Such actions do not generally trigger a false

alarm, but if they occur in just the right manner they can appear as a fall to the system. A more sophisticated tracking algorithm could potentially deal with some of these false alarms. However, ultimately, there is a need to determine whether a foreground object actually is or is not a person, and in some cases, is or is not an elderly person. Such a determination is quite challenging given that individuals could be in any location, with any posture, and never be facing the Kinect prior to falling.

Although processing of the data for this analysis was performed offline, real-time operation is possible using low cost hardware. Systems installed in independent living facilities are currently running the fall detection algorithm in real time (10 frames per second) using computers equipped with a dual core Intel Atom CPU (D2700). The computers are small in size, similar to a standard paperback book, which allows for a small footprint when installed in a home.

The major limitation of the described method is the need for the fall to be in view of the sensor. Although a Kinect sensor in each room of an apartment would likely detect most falls, given the field of view of the device and the possibility of occlusion, multiple sensors per room may be required to cover all areas.

Future work includes further improvement of the fall detection methodology, along with personalization of the fall detection threshold used in a home based on the fall risk of the resident(s). Such fall risk information can be computed automatically, in real time, from walking sequences analyzed in the home using the same Kinect system [34], [35].

REFERENCES

[1] (2013). Center for Disease Control and Prevention (CDC), "Falls among older adults: An overview," [Online]. Available: http://www.cdc.gov/homeandrecreationalsafety/Falls/adultfalls.html

[2] J. A. Stevens, P. S. Corso, E. A. Finkelstein, and T. R. Miller, "The costs of fatal and non-fatal falls among older adults," *Injury Prevention*, vol. 12, no. 5, pp. 290–295, 2006.

[3] 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.

[4] N. Noury, A. Fleury, P. Rumeau, A. K. Bourke, G. O. Laighin, V. Rialle, and J. E. Lundy, "Fall detection-principles and methods," in *Proc. IEEE 29th Ann. Int. Conf. Eng. Med. Biol. Soc.*, 2007, pp. 1663–1666.

[5] A. K. Bourke, P. W. van de Ven, A. E. Chaya, G. M. OLaighin, and J. Nelson, "Testing of a long-term fall detection system incorporated into a custom vest for the elderly," in *Proc. IEEE 30th Ann. Int. Eng. Med. Biol. Soc. Conf.*, 2008, pp. 2844–2847.

[6] 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.

[7] M. Prado-Velasco, M. G. del Rio-Cidoncha, and R. Ortiz-Marin, "The inescapable smart impact detection system (ISIS): An ubiquitous and personalized fall detector based on a distributed 'divide and conquer strategy'," in *Proc. IEEE 30th Annu. Int. Eng. Med. Biol. Soc. Conf.*, 2008, pp. 3332–3335.

[8] G. Demiris, M. J. Rantz, M. A. Aud, K. D. Marek, H. W. Tyrer, M. Skubic, and A. A. Hussam, "Older Adults' attitudes towards and perceptions of smart home technologies: A pilot study," *Inform. Health Social Care*, vol. 29, no. 2, pp. 87–94, 2004.

[9] M. Alwan, P. J. Rajendran, S. Kell, D. Mack, S. Dalal, M. Wolfe, and R. Felder, "A smart and passive floor-vibration based fall detector for elderly," in *Proc. IEEE Int. Conf. Inf. Commun. Technol.*, 2006, pp. 1003–1007.

[10] Y. Zigel, D. Litvak, and I. Gannot, "A method for automatic fall detection of elderly people using floor vibrations and sound—Proof of concept on human mimicking doll falls," *IEEE Trans. Biomed. Eng.*, vol. 56, no. 12, pp. 2858–2867, Dec. 2009.

[11] A. Sixsmith, N. Johnson, and R. Whatmore, "Pyrolitic IR sensor arrays for fall detection in the older population," *J. Phys. IV France*, vol. 128, pp. 153–160, 2005.

[12] Y. Li, Z. L. Zeng, M. Popescu, and K. C. Ho, "Acoustic fall detection using a circular microphone array," in *Proc. IEEE 32nd Annu. Int. Eng. Med. Biol. Soc. Conf.*, 2010, pp. 2242–2245.

[13] C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau, "Robust video surveillance for fall detection based on human shape deformation," *IEEE Trans. Circ. Syst. Video Technol.*, vol. 21, no. 5, pp. 611–622, May 2011.

[14] D. Anderson, J. M. Keller, M. Skubic, X. Chen, and Z. He, "Recognizing falls from silhouettes," in *Proc. IEEE 28th Annu. Int. Eng. Med. Biol. Soc. Conf.*, 2006, pp. 6388–6391.

[15] S. G. Miaou, P. H. Sung, and C. Y. Huang, "A customized human fall detection system using omni-camera images and personal information," in *Proc. Transdisciplinary Conf. Distrib. Diagnosis Home Healthcare*, 2006, pp. 39–42.

[16] T. Lee and A. Mihailidis, "An intelligent emergency response system: Preliminary development and testing of automated fall detection," *J. Telemed. Telecare*, vol. 11, no. 4, pp. 194–198, 2005.

[17] D. Anderson, R. H. Luke, J. Keller, M. Skubic, M. Rantz, and M. Aud, "Linguistic summarization of activities from video for fall detection using voxel person and fuzzy logic," *Comput. Vision Image Understanding*, vol. 113, no. 1, pp. 80–89, 2009.

[18] E. Auvinet, F. Multon, A. Sait-Arnaud, J. Rousseau, and J. Meunier, "Fall detection with multiple cameras: An occlusion-resistant method based on 3-D silhouette vertical distribution," *IEEE Trans. Inf. Tech. Biomed.*, vol. 15, no. 2, pp. 290–300, Mar. 2011.

[19] M. Kepski, B. Kwolek, and I. Austvoll, "Fuzzy inference-based reliable fall detection using Kinect and accelerometer," in *Proc. 11th Int. Conf. Artif. Intell. Soft Comput.*, pp. 266–273, 2012.

[20] C. Marzahl, P. Penndorf, I. Bruder, and M. Staemmler, "Unobtrusive fall detection using 3D images of a gaming console: Concept and first results," in *Ambient Assisted Living*, 2012, pp. 135–146.

[21] G. Mastorakis and D. Makris, "Fall detection system using Kinect's infrared sensor," *J. Real-Time Image Process.*, Mar. 2012. DOI: 10.1007/s11554-012-0246-9.

[22] C. Rougier, E. Anvient, J. Rousseau, M. Mignotte, and J. Meunier, "Fall detection from depth map video sequences," in *Proc. Int. Conf. Smart Homes Health Telematics*, 2011, pp. 121–128.

[23] R. Planinc and M. Kampel, "Introducing the use of depth data for fall detection," *Personal Ubiquitous Comput.*, vol. 17, pp. 1063–1072, 2012.

[24] G. Demiris, O. D. Parker, J. Giger, M. Skubic, and M. Rantz, "Older adults' privacy considerations for vision based recognition methods of eldercare applications," *Technol. Health Care*, vol. 17, pp. 41–48, 2009.

[25] M. Piccardi, "Background subtraction techniques: A review," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2004, pp. 3099–3104.

[26] Z. Zhou, E. Stone, M. Skubic, J. M. Keller, and Z. He, "Nighttime in-home activity monitoring for elder-care," in *Proc. IEEE Int. Conf. Eng. Med. Biol. Soc.*, 2011, pp. 5299–5302.

[27] (2013). OpenNI. [Online]. Available: www.openni.org

[28] R. Planinc and M. Kampel, "Robust fall detection by combining 3D data and fuzzy logic," in *Proc. ACCV Workshop Color Depth Fusion Comput. Vision*, 2012, pp. 121–132.

[29] (2014). OpenKinect. [Online]. Available: www.openkinect.org

[30] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," presented at the IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recog., Fort Collins, CO, USA, 1999.

[31] C. D. Fryar, Q. Gu, and C. L. Ogden, "Anthropometric reference data for children and adults: United States, 2007–2010," *Nat. Center Health Stat., Vital Health Stat*, vol. 11, no. 252, pp. 1–40, 2012.

[32] M. Rantz, M. A. Aud, G. Alexander, B. J. Wakefield, M. Skubic, R. H. Luke, D. Anderson, and J. M. Keller, "Falls, technology, and stunt actors: New approaches to fall detection and fall risk assessment," *J. Nursing Care Quality*, vol. 23, no. 3, pp. 195–201, 2008.

[33] "PrimeSense^{TM} NITE Algorithms 1.5," PrimeSense, Inc., Tel Aviv, Israel, 2011.

[34] E. Stone and M. Skubic, "Unobtrusive, continuous, in-home gait measurement using the Microsoft Kinect," *IEEE Trans. Biomed. Eng.*, vol. 60, no. 10, pp. 2925–2932, Oct. 2013.

[35] E. Stone and M. Skubic, "Mapping kinect-based in-home gait speed to TUG Time: A methodology to facilitate clinical interpretation," in *Proc. ICST Seventh Int. Conf. Pervasive Comput. Technol. Healthcare*, 2013, pp. 57–64.

Erik E. Stone (S'11–M'14) received the B.S. degree in electrical and computer engineering, the M.S. degree in computer engineering, and the Ph.D. degree in electrical and computer engineering from the University of Missouri, Columbia, MO, USA, in 2006, 2009, and 2013, respectively.

He is currently a Postdoctoral Fellow in the Center for Eldercare and Rehabilitation Technology at the University of Missouri, Columbia, MO, USA. His research interests include computer vision, machine learning, and pattern recognition.

Marjorie Skubic (S'90–M'91) received the Ph.D. degree in computer science from Texas A&M University, College Station, TX, USA, in 1997, where she specialized in distributed telerobotics and robot programming by demonstration.

She is currently a Professor in the Department of Electrical and Computer Engineering, University of Missouri, Columbia, with a joint appointment in Computer Science. In addition to her academic experience, she has spent 14 years working in industry on real-time applications such as data acquisition and automation. Her current research interests include sensory perception, computational intelligence, spatial referencing interfaces, human-robot interaction, and sensor networks for eldercare. In 2006, he established the Center for Eldercare and Rehabilitation Technology, University of Missouri and serves as the Center Director for this interdisciplinary team. The focus of the center's work includes monitoring systems for tracking the physical and cognitive health of elderly residents in their homes, logging sensor data in an accessible database, extracting activity and gait patterns, identifying changes in patterns, and generating alerts for health changes.