Fall Detection Based on Depth-Data in Practice

Christopher Pramerdorfer[1(✉)], Rainer Planinc[1], Mark Van Loock[2], David Fankhauser[1], Martin Kampel[1], and Michael Brandstötter[1]

[1] CogVis, Vienna, Austria
pramerdorfer@cogvis.at
[2] Toyota Motor Europe, Brussels, Belgium

Abstract. Falls are a leading cause of accidental deaths among the elderly population. The aim of fall detection is to ensure quick help for fall victims by automatically informing caretakers. We present a fall detection method based on depth-data that is able to detect falls reliably while having a low false alarm rate – not only under experimental conditions but also in practice. We emphasize person detection and tracking and utilize features that are invariant with respect to the sensor position, robust to partial occlusions, and computationally efficient. Our method operates in real-time on inexpensive hardware and enables fall detection systems that are unobtrusive, economic, and plug and play. We evaluate our method on an extensive dataset and demonstrate its capability under practical conditions in a long-term evaluation.

Keywords: Fall detection · Depth-data · Evaluation · Practice

1 Introduction

Falls are a major public health problem. Between 30 % and 60 % of US citizens of age 65 or older suffer from falls each year, and 10 % to 20 % of these falls result in serious injury, hospitalization, or death [1]. In fact, falls are the leading cause of accidental death in this age group [2]. Immediate help and treatment of fall-induced injuries is vital for minimizing morbidity and mortality rates [3,4]. However, statistically every other fall victim is unable to get back up without help [2]. Falls are thus particularly dangerous to older persons that live alone.

To this end, there has been active research in the field of *fall detection*, with the aim of detecting fall incidents and informing caretakers such as family members or ambulance personnel automatically [4]. One approach to fall detection is to employ optical sensors and computer vision. For example, several methods based on video cameras have been proposed [4]. However, camera images convey limited information in terms of scene geometry, reducing the robustness of such methods. Moreover, cameras do not work in darkness, hence unobtrusive fall detection during nighttime is not possible. Furthermore, camera-based fall detection raises privacy concerns [5]. Active depth sensors do not have these limitations. They work in darkness, measure distances from which scene geometry can be recovered, and are robust to illumination changes and shadows. A well-known example is the Microsoft Kinect.

G. Hua and H. Jégou (Eds.): ECCV 2016 Workshops, Part II, LNCS 9914, pp. 195–208, 2016.
DOI: 10.1007/978-3-319-48881-3_14

Several fall detection methods using the Kinect sensor have been proposed, most of which utilize background subtraction [6] for person detection and height-based features for fall detection. For instance, [5,7,8] analyze the centroid height of persons, whereas [9] examine head heights. In [10] falls are detected based on the spine height and orientation. Many methods (e.g. [5,11,12]) also incorporate velocities, arguing that the vertical velocity of persons increases significantly during falls. Dubey et al. [13] follow a different approach, extracting HU features [14] from motion history images and using a SVM for classification. All these methods reportedly perform well on simulated falls, but none were evaluated extensively under real-world conditions.

These methods have limitations in practical use, in which falls must be detected regardless of fall speed and ending pose. Under these circumstances, velocity is not a reliable feature for fall detection. Falls may end in a sitting position, but it seems that only [10] take such falls into account. Furthermore, falls might be partially occluded or invisible due to the limited field of view, a challenge that is not addressed by many of these methods. Several methods (e.g. [5,8,9]) do not perform tracking nd thus do not support multiple moving objects.

Some of the discussed methods also lack in terms of practicability. Those presented in [7,13] seem to be too complex for real-time operation on inexpensive hardware such as ARM-based single-board computers, which we consider important to allow for a broad acceptance. The method presented in [9] is able to do so but requires a wearable device, which is intrusive. The features used in [9,13] are not invariant with respect to the sensor position, which restricts the sensor placement and thus complicates the system setup.

In this paper, we present a fall detection method that aims to address these shortcomings. Our method can detect slow and partially occluded falls as well as falls ending in a sitting position. We utilize the available distance information to recover the geometry of objects in an invariant and efficient way, which enables discriminative features for person and fall detection. Automatic calibration enables plug and play operation and flexibility in terms of sensor placement, facilitating the installation. In order to minimize the hardware costs for end users, our method was designed to be efficient enough to run on inexpensive hardware such as the Raspberry Pi 2 or the Odroid C1+. Our method supports various inexpensive off-the-shelf depth sensors including those based on PrimeSense technology (e.g. Kinect 1, Asus Xtion, Orbbec) and Kinect 2.

We evaluate the performance of our fall detection method using a comprehensive test dataset as well as a large publicly available dataset. Furthermore, we present results of a long-term study carried out under real-world conditions. The results show that our method is able to detect even challenging falls reliably while having a low false alarm rate in practice (around one false alarm per week).

This paper is organized as follows. Our fall detection method involves the steps (i) automatic calibration and scene analysis, (ii) motion detection, (iii) person detection and tracking, and (iv) fall detection, which are presented in Sects. 2, 3, 5, and 6, respectively. Experimental results are presented in Sect. 7, and conclusions are drawn in Sect. 8.

2 Automatic Calibration and Scene Analysis

Our fall detection method is flexible in terms of sensor placement in order to facilitate the system setup and support plug-and-play; the only requirement is that a part of the floor must be visible to the sensor. In order to achieve plug-and-play functionality, we perform automatic sensor calibration at system startup in order to estimate the sensor position and orientation. On this basis, we locate areas in which reliable fall detection is possible, and detect scene objects.

2.1 Automatic Calibration

The purpose of automatic calibration is to recover the sensor extrinsics (position and orientation). This is accomplished using ground floor detection. Our algorithm for this purpose is based on filtering using normal vectors followed by iterative RANSAC [15] plane fitting. We found this method to be more reliable than alternatives [16] in case of significant floor occlusions.

Floor detection is accomplished by first converting a depth map $\mathcal{D}$ from the sensor to an organized point cloud $\mathcal{C}$ in camera coordinates (organized means that the structure is preserved, so that $\mathcal{D}(u,v)$ and $\mathcal{C}(u,v)$ correspond). This allows us to estimate normals $\mathcal{N}$ efficiently,

$$\begin{aligned} \mathcal{N}(u,v) &= -\mathbf{n}(u,v)/||\mathbf{n}(u,v)||_2 \\ \mathbf{n}(u,v) &= (\mathcal{C}(u+1,v) - \mathcal{C}(u,v)) \times (\mathcal{C}(u,v+1) - \mathcal{C}(u,v)). \end{aligned} \tag{1}$$

We then discard all points whose normal vectors do not coincide with the sensor tilt, which is assumed to be between 0° and 60° downwards. This assumption is general enough to allow for flexible sensor positioning and enables us to discard points that cannot possibly represent the floor. This increases the robustness and efficiency of the subsequent floor detection step.

In order to detect the floor plane reliably in presence of furniture and clutter, we iteratively find the best-fit plane for the remaining points using RANSAC and remove all inliers. This procedure continues until the number of inliers decreases below a threshold. This generally leads to several detected planes. As the sensor is located at the origin of the camera coordinate system, the plane that corresponds to the floor is that with the largest distance to the origin.

The equation of this plane, (a,b,c,d) with $\|(a,b,c)\|_2 = 1$, is then used to recover the extrinsics; the sensor height is d and the orientation corresponds to the rotation that maps $(0,1,0)$ (the normal vector of the floor plane in world coordinates, assuming that the positive Y axis points upwards) to (a,b,c).

2.2 Scene Analysis

Once the extrinsics are estimated, we convert a depth map to a point cloud in world coordinates and detect scene objects via height-based point classification; a point is classified as part of an object if its height is between 30 cm and 90 cm,

which includes beds, couches, and other resting accommodations. Object detection is carried out periodically in order to support scene changes at runtime.

Furthermore, we analyze the scene in order to determine which regions qualify for reliable fall detection. This is the case if both fallen and upright persons in a considered region would be in the field of view of the sensor.

The information obtained during scene analysis is utilized for person state prediction (Sect. 6.1), and for visual feedback on the sensor setup (Fig. 1).

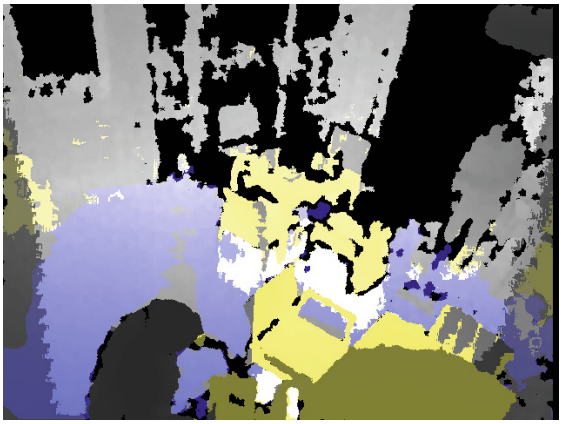


Fig. 1. Scene visualization after automatic calibration and scene analysis. The detected floor is shown in blue, scene objects are yellow, and regions in which reliable fall detection is impossible are shaded (e.g. left side of the image). (Color figure online)

3 Motion Detection

Falls are characterized by motion. For efficiency, we therefore consider only areas in which motion occurs. This is achieved by means of background subtraction, i.e. motion is detected by comparing image frames to a background model that represents the static parts of the scene [17]. Depth maps are well-suited for this purpose as they encode distances and are thus robust to clothing color and illumination changes [18]. On the other hand, this implies that background subtraction must be sensitive in order to reliably capture fallen persons, which are close to the background. To this end, we propose a distance-dependent noise model that ensures that fallen persons are detected reliably.

The noise model described in this section is optimized for PrimeSense sensors, but adaption to other sensors such as Kinect 2 is straightforward.

3.1 Noise Model and Pixel Classification

In [19] it was found that the random measurement errors of the Kinect increase quadratically with distance d, with an RMSE of $e_d \approx 1.425 \cdot 10^{-6} d^2$. We utilize this information to derive a distance-dependent noise model for background

subtraction by approximating the noise of the sensor at object distance d as a normal distribution with $\mu = 0$ and $\sigma = e_d$. On this basis, we cast pixel classification as a novelty detection problem; a pixel with value v is classified as foreground if (i) v differs by more than $3e_m$ from the corresponding background model value m, and (ii) if $v - m$ is negative. The second condition encodes the fact that foreground objects must always appear in front of the background. This effectively suppresses ghosts (foreground areas due to background changes [17]).

In order to account for the fact that our noise model is an approximation, as well as increased sensor noise at object borders, we perform morphological erosion with a small structuring element after all pixels are classified.

Our distance-dependent noise model and pixel classification methods ensure a high sensitivity and thus that moving objects are captured reliably throughout the measuring range, while at the same time suppressing noise.

3.2 Background Model

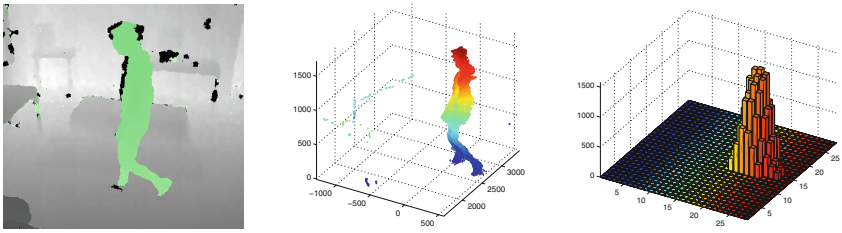
As our noise model is unimodal and depends only on the object distance, we employ a background model in the form of a single matrix that represents the central tendency of the observed measurements in every pixel. This model is simple to process and maintain, which is important considering the limited computational resources available. As the noise model is already known, an initial training phase is not required. In fact, our method only requires a single frame for initialization; the observed pixel values of the first frame constitute the initial values of the background model. Model pixels that are zero (which denotes an unsuccessful measurement) are inpainted.

Under practical conditions, the background might change over time, which must be accounted for by updating the background model. To this end, we periodically compare every pixel value $v \neq 0$ of the current frame with the corresponding background model value m; m is increased if $m < p$ and decreased if $m > p$. This causes m to converge towards the temporal median [20].

Such gradual update strategies entail a compromise in terms of the update rate; high rates cause unmoving foreground objects to disappear quickly (which can hinder fall verification) whereas low rates cause persistent errors in case of background changes. To overcome this problem, we employ a second, complementary means for updating the model; we periodically test for whether $v - m > 3e_m$ and, if so, set $m := v$. This method effectively compensates background changes and allows us to keep the gradual update rate low.

4 Conversion to Plan-View Space

A key characteristic of our fall detection method is that all analysis is done in *plan-view space* [21]. This space, which resembles a synthetic top-view of the scene under orthographic projection, is well-suited for fall detection for two reasons. First, it represents the scene geometry in a concise way, allowing real-time operation on low-end hardware. Second, it is invariant with respect to the



(a) depth map (b) point cloud (c) height map

Fig. 2. Transformation of foreground pixels (green) to plan-view space. (Color figure online)

sensor position and orientation, enabling invariant features for person and fall detection as well as facilitating the system setup.

Foreground pixels are mapped to plan-view space by first reprojecting them to world coordinates. The resulting points are then downsampled along the X and Z axes (the positive Y axis points upwards) and discretized to obtain plan-view coordinates. In this process, several points may be mapped to the same plan-view coordinates, and different means for consolidating these points to a single scalar result in different scene representations. We employ two kinds of these representations, occupancy maps $\mathcal{O}$ and height maps $\mathcal{H}$ [22]. The former encode the number of points mapped to each plan-view coordinate, whereas the latter store the largest observed Y coordinates [21,22]. Figure 2 illustrates the process of height map generation. After conversion, both maps are smoothed using a Gaussian filter to compensate for rounding effects.

Occupancy and height maps are complementary with regard to person detection. The former are robust to sensor noise but not to occlusions, while the latter are robust to partial occlusions but susceptible to noise. To this end, we set all occupancy and height map pixels whose occupancy is below a threshold to 0, which is a reliable method for noise removal [22,23].

5 Person Detection and Tracking

Our fall detection method includes an effective person detection and tracking stage. This enables reliable analysis even if there are multiple moving objects (e.g. persons or pets) in the scene. This stage entails two steps, the detection of persons in the current frame, and tracking of persons over time.

5.1 Person Detection

We perform person detection on a per-region basis in plan-view space. For this purpose, we find all connected components in the binary height map $\mathcal{H} > 0$, which are then classified as (non-)persons. For classification we use a vector $\mathbf{f}$ of four features that encode object geometry: (i) occupied area (number of

plan-view pixels), (ii) object height (0.95th quantile of height map values), (iii) object density (0.95th quantile of occupancy map values), and (iv) object shape (side ratio of the bounding box). These features have a clear interpretation, are efficient to compute, and robust with respect to person position and orientation.

Classification is performed by a random forest [24] that was trained on about $20,000$ frames depicting persons and other foreground objects such as chairs, rollators, and pets. The frames were extracted from a subset of sequences in our test dataset (Sect. 7.1), and foreground objects were manually segmented to obtain ground-truth data. The frames depict persons performing various activities, including walking, using a wheelchair or rollator, sitting, and lying, in order for the classifier to recognize persons regardless of pose. The frames depict persons at varying levels of occlusions in order to obtain a classifier that is robust in this regard. Random forest hyperparameters were cross-validated.

Random forest classifiers are well-suited for our task because they generalize well [24], are efficient, and predict class-conditional probabilities $\Pr(P|\mathbf{f})$.

5.2 Tracking

The goal of the tracking step is to associate person regions (those for which $\Pr(P = 1|\mathbf{f}) > 0.5$) between frames. For this purpose, we represent each person region R_j by a feature vector $\mathbf{x}_j = (\mathbf{c}_j; \mathbf{f}_j)$, which is utilized for computing association costs. $\mathbf{c}_j = (x_j, z_j)$ is the location (center of mass) of R_j,

$$\mathbf{c}_j = \frac{1}{\sum \mathcal{O}(R_j)} \sum_{\mathbf{p} \in R_j} \mathcal{O}(\mathbf{p})\mathbf{p}. \tag{2}$$

The goal is thus to associate n person regions in the current frame with m regions in the previous frame in a way that minimizes a global association cost. Our per-sample association cost w_{ij} incorporates both proximity and feature similarity of the associated regions R_i and R_j, and is defined as

$$w_{ij} = \begin{cases} \alpha_1 \left\| \mathbf{c}_i - \mathbf{c}_j \right\|_2 + \alpha_2 \left\| \mathbf{f}_i - \mathbf{f}_j \right\|_2 & \text{if } \left\| \mathbf{c}_i - \mathbf{c}_j \right\|_2 < t_c \\ \infty & \text{otherwise.} \end{cases} \tag{3}$$

The factors α_1, α_2 weight the impact of proximity and feature similarity, while t_c accounts for the fact that the velocity of persons is limited.

In order to be able to solve the resulting optimization problem efficiently, we demand that $m = n$ and that each track must be assigned to a different region. This results in a linear assignment problem, which can be solved using the Hungarian algorithm [25]. We ensure that $m = n$ (which is not always the case because persons may enter or leave the view at any time) by introducing dummy regions [26]. Before association, the location of regions of the previous frame are predicted for the current frame using Kalman filters [27].

6 Fall Detection

Fall detection comprises the steps state prediction, event detection, and fall verification. The purpose of the first step is to estimate the state of every person that is being tracked. These state predictions are analyzed over time in order to detect events such as falls. This follows an optional verification step that aims to reduce the false alarm rate via long-term and scene analysis.

State and event detection are carried out in a probabilistic framework. This allows our method to report reliable event confidence scores, enabling caretakers to balance the trade-off between sensitivity and specificity. These scores are also used for automatic event filtering and routing. For instance, our method can be configured to send all fall events with a confidence greater than 0.5 via mail, and additionally send a text message if the confidence is greater than 0.9.

6.1 State Prediction

Persons can be in the following states: (i) Fallen (in a pose typical for fallen persons, such as lying or sitting), (ii) Active (any other pose), and (iii) Resting (being on top of a resting accommodation). State prediction is carried out independently in each frame. For this purpose, we define two binary random variables A and R, with $A = 1$ and $A = 0$ representing the Active and Fallen state, respectively, and R encoding whether the person is resting or not.

To predict $\Pr(A)$, we reuse the feature vector $\mathbf{f}$. $\mathbf{f}$ is discriminative for this purpose as it encodes geometrical properties that change significantly due to falls. For prediction we employ a binary random forest classifier that was trained on a subset of the frames used to train the person classifier (those depicting persons).

$\Pr(R = 1)$ is calculated as the fraction of the area occupied by a person that overlaps with scene objects. In order to ensure that falls beneath such objects (which is possible in case of tables) are correctly detected, person areas must be located above object areas in order to affect $\Pr(R = 1)$.

We are mainly interested in $\Pr(A = 0, R = 0)$, which corresponds to the condition of a person after a fall that should be detected. Assuming that A and R are independent, this joint distribution factorizes to $\Pr(A = 0)\Pr(R = 0)$. Let $\mathcal{P}_A$ and $\mathcal{P}_F$ denote $\Pr(A = 1, R = 0)$ and $\Pr(A = 0, R = 0)$, respectively.

6.2 Event Detection

Falls and other actions performed by humans are temporal in nature. We thus utilize the available tracking information and detect falls and related events via temporal analysis, based on recently observed person states.

Before performing event detection, we integrate state predictions from recent frames in order to increase their reliability. This is motivated by the observation that the state of a person is unlikely to change between frames. For instance, if a person is Active in frame $f-1$, they are likely to be Active in frame f as well. We considered different temporal models such as Markov chains for modeling this circumstance, but found that simply averaging state probabilities in a short time

window leads to comparable results while being more efficient. In mathematical terms, we thus compute e.g.

$$\mathcal{P}_A^{f+} = \frac{1}{\beta} \sum_{t=0}^{\beta-1} \mathcal{P}_A^{f-t}, \tag{4}$$

with with $\mathcal{P}_A^f$ being $\mathcal{P}_A$ in frame f, and $\beta \in \mathbb{N}$ defining the time window.

On this basis, we analyze the temporal evolution of $\mathcal{P}_A^+$ and $\mathcal{P}_F^+$ in order to detect two types of events, falls and recoveries. A fall event is triggered if the maximum over $\mathcal{P}_F^+$ in a time interval around the current frame f exceeds a specified threshold,

$$\mathcal{P}_F^* = \max(\mathcal{P}_F^{f-\gamma+}, \ldots, \mathcal{P}_F^{f+}, \ldots, \mathcal{P}_F^{f+\gamma+}) > t_{\mathcal{P}_F}. \tag{5}$$

A recovery event signals that a person managed to get back up again after a fall, i.e. if $\mathcal{P}_A^* > t_{\mathcal{P}_A}$ and if a fall event occurred recently for the same person.

We do not analyze the velocity of persons for fall detection, because doing so would prevent us from being able to reliably detect slow falls or falls originating from a position other than upright (e.g. persons rolling out of the bed). In fact, we detect falls solely based on the person pose; if a person lies or sits on the floor for several seconds, a fall event is triggered. This has the advantage that persons that fell outside the field of view can still trigger a fall event (and thus receive help) by crawling into the field of view.

6.3 Fall Verification

The purpose of fall verification is to reduce the false alarm rate. Fall verification consists of three tests that are carried out if a fall event is triggered. All tests can be disabled independently. Test 1 suppresses multiple fall alarms that occur in quick succession for the same person, unless corresponding recovery events occur in between. Test 2 suppresses a fall event if there is another Active person in the scene at the time the event occurs (i.e. if help is already available). Test 3 suppress a fall event if a recovery event is registered for the same person within one minute, i.e. if the person was able to recover on their own.

7 Results and Discussion

We evaluate our fall detection method on an extensive dataset of test sequences and assess its false alarm rate under real-world conditions in a long-term study.

7.1 Performance Under Experimental Conditions

In order to study the fall detection performance, we compiled a dataset of 579 test sequences. These sequences depict persons that simulate different types of falls (following the protocol shown in Table 1) and various activities of daily

living (e.g. using a wheelchair or rollator, sitting, cleaning). The sequences were recorded in an attempt to capture as much variability that occurs in practice as possible (e.g. different persons, rooms, and locations; partially visible or occluded actions; interaction with scene objects such as chairs). The dataset includes the CVL dataset [10], which contains 80 fall sequences and 64 sequences with activities of daily living. In total, there are 146 fall incidents in our dataset.

Table 1. Protocol for simulating fall incidents. Our dataset covers all combinations.

Property	Values
Activity prior to fall	Walking, standing, sitting
Fall against	Nothing, wall, unmovable object, movable object
Fall direction	Forward, backward
Partially occluded	No, yes
Walking aid	No, yes

For evaluation, we apply our fall detection method on every test sequence and compare the number of reported fall events to the ground-truth information. On this basis, we compute the precision and recall of our method at varying fall event confidence thresholds $t_{\mathcal{P}_F}$ (between 0.5 and 0.95). The precision is the fraction of triggered fall events that correspond to an actual fall, while the recall is the fraction of falls in the dataset that were detected by our method. We use the same configuration for all sequences and disable fall verification test 3 (Sect. 6.3) as most test sequences are shorter than one minute.

Figure 3 shows the resulting recall vs. precision graph. At $t_{\mathcal{P}_F} = 0.5$, seven of the 146 falls were undetected for the following reasons. In three cases there was significant sensor noise that caused motion detection to fail. This noise occurs only initially and is thus unlikely to affect the fall detection performance in practice. The remaining four falls were particularly challenging because they were from a sitting position, only partially visible, and partially occluded.

18 false alarms were registered at $t_{\mathcal{P}_F} = 0.5$, all of which were due to objects that were left in the scene. In the cases that lead to these false alarms, these objects were considered as persons due to person classification and tracking errors. These errors happened when persons moved large objects such as rollators and chairs while being partially outside the field of view and/or occluded.

Figure 3 illustrates the benefit of the event confidence scores reported by our method: they allow users such as caretakers to balance the trade-off between sensitivity and specificity according to their needs by setting $t_{\mathcal{P}_F}$ accordingly. We note that our method reliably assigns high confidence scores (greater than 0.8) to all but partially invisible or occluded falls. Figure 4 illustrates two challenging test sequences, one that is handled correctly and one that causes a false alarm.

In order to provide a comparison with other fall detection methods, we additionally evaluated our method on the public CVL dataset [10]. To our knowledge,

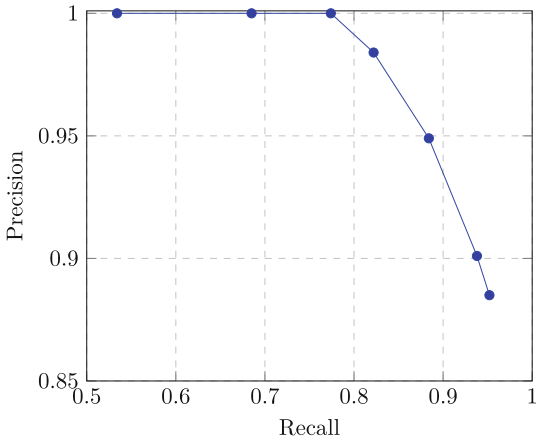
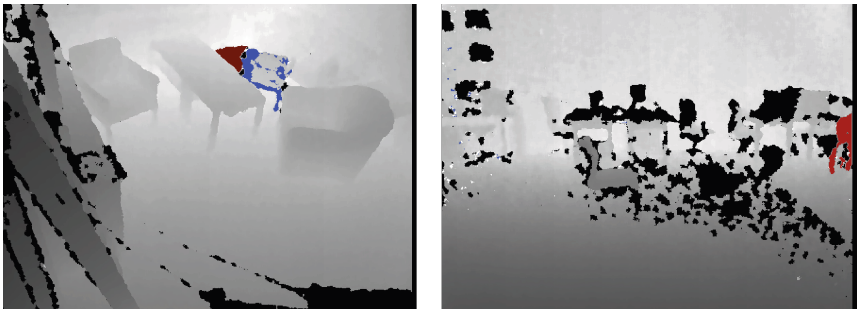


Fig. 3. Recall vs. precision of our fall detection method on the test dataset. The graph was obtained by varying $t_{\mathcal{P}_F} \in \{0.5, 0.6, 0.7, 0.8, 0.85, 0.9, 0.95\}$.



(a) limited field of view (b) wheelchair

Fig. 4. Challenging test sequences. Left: A fall that is both partially occluded and in a region with limited field of view. Despite these challenging conditions, the fall is detected successfully with a confidence of 58 %. Right: A wheelchair remains at the border of the view, triggering a false alarm with a low confidence. (a) limited field of view (b) wheelchair.

this is the largest public fall dataset available. Our method triggered no false alarms and detected 78 out of 80 falls that occur in this dataset. Both false negatives were caused by the strong sensor noise mentioned before. Method [10] detected all falls, but only half of the CVL dataset was used for evaluation.

7.2 Performance Under Real-World Conditions

At the time of writing, we are performing a long-term evaluation of our fall detection method in nursing and assisted living homes, i.e. under real-world conditions. In this section, we report the results obtained in a period of six months.

During this time, 53 of our fall detection systems were active for 5,246 full days in total (125,904 h). To our knowledge, this is the first long-term evaluation of a depth-data-based fall detection method.

For evaluation purposes, we labeled all fall events with a confidence greater than 0.6 in this timeframe as true positives or false positives (false alarms), using event visualizations generated by our method. We labeled 164 fall events as true positives as they were triggered by persons that were lying or sitting on the floor for longer than one minute. No falls were registered by the caretakers in the evaluation period. This means that these events were likely caused by deliberate actions, although it is possible that some were caused by actual falls after which the persons were able to recover on their own. As such, the results reported in this section do not directly correspond to the sensitivity of our method.

Figure 5 summarizes the results, which were obtained by dividing the number of true and false positives by the number of active days. During the considered six-month period, there were 0.14 false alarms with a confidence greater than 0.6 per day on average (i.e. per 24 h of system uptime). This amounts to a single false alarm per week at this confidence threshold. Most false alarms were again caused by objects (e.g. chairs or walking aids) that were left in the scene. Another major cause of false alarms was a large dog that would sometimes appear similarly to a fallen person in the depth data.

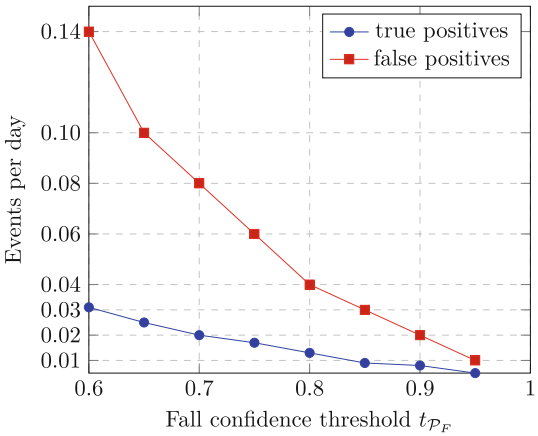


Fig. 5. Average number of true positives and false positives of our fall detection method per day under real-world conditions.

Our fall detection method assigned a low confidence score to a large fraction of these false alarms, so that increasing the threshold to 0.7 almost halves the false alarm rate. Doing so decreases the true positive rate by around one-third in relative terms, affecting mainly falls ending in a sitting position.

Even with conservative confidence thresholds, the false alarm rate is low enough in practice to impose little additional workload for caretakers, and allows a single caretaker to supervise many of our fall detection systems simultaneously.

7.3 Computational Efficiency

Our fall detection method was designed to be efficient enough to run on inexpensive low-end hardware. It achieves more than 30 fps on an Odroid C1+ single board computer, which has a price of only $32 at the time of writing.

In practice, we limit the sframerate to 15 so that two fall detection instances can run in parallel, enabling simultaneous fall detection in two rooms. This does not decrease the fall detection performance (all results reported in this section were obtained using this framerate).

8 Conclusions

We have presented a fall detection method that analyzes depth data in order to achieve a high sensitivity and specificity under real-world conditions, as verified in a comprehensive performance evaluation. Our method addresses limitations of existing depth-based fall detection methods in terms of sensitivity (inability to detect slow, sitting, or occluded falls) and practicability (high hardware costs, complicated or restrictive system setup), and provides reliable fall confidence scores as additional information for caretakers.

Future work will concentrate on further increasing the fall detection performance of our method. To this end, we are continuing to extend our test dataset in order to obtain more data for classifier training and evaluation. We also plan to improve our algorithms by modeling occlusions explicitly in order to improve person detection and state prediction performance.

References

1. Rubenstein, L.Z.: Falls in older people: epidemiology, risk factors and strategies for prevention. Age Ageing **35**, 1137–1141 (2006)
2. Porter, R.S.: The Merck Manual of Diagnosis and Therapy. Wiley, New York (2011)
3. Noury, N., Rumeau, P., Bourke, A., ÓLaighin, G., Lundy, J.: A proposal for the classification and evaluation of fall detectors. IRBM **29**(6), 340–349 (2008)
4. Mubashir, M., Shao, L., Seed, L.: A survey on fall detection: principles and approaches. Neurocomputing **100**, 1–9 (2012)
5. Rougier, C., Auvinet, E., Rousseau, J., Mignotte, M., Meunier, J.: Fall detection from depth map video sequences. In: Proceedings of International Conference Smart Homes and Health Telematics, pp. 121–128 (2011)
6. Brutzer, S., Hoferlin, B., Heidemann, G.: Evaluation of background subtraction techniques for video surveillance. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 1937–1944 (2011)
7. Auvinet, E., Meunier, J.: Head detection using Kinect camera and its application to fall detection. In: Proceedings of International Conference on Information Science, Signal Processing and Their Applications, pp. 164–169 (2012)

8. Dubois, A., Charpillet, F.: Automatic fall detection system with a RGB-D camera using a hidden Markov model. In: Proceedings of International Conference on Smart Homes and Health Telematics, pp. 259–266 (2013)
9. Kepski, M., Kwolek, B.: Fall detection using ceiling-mounted 3D depth camera, pp. 640–647 (2014)
10. Planinc, R., Kampel, M.: Robust fall detection by combining 3D data and fuzzy logic. In: Park, J.-I., Kim, J. (eds.) ACCV Workshops 2012, Part II. LNCS, vol. 7729, pp. 121–132. Springer, Heidelberg (2013)
11. Kumar, D.P., Yun, Y., Gu, I.Y.H.: Fall detection in RGB-D videos by combining shape and motion features. In: Proceedings of International Conference on Acoustics, Speech and Signal Processing, pp. 1337–1341 (2016)
12. Yun, Y., Gu, I.Y.H.: Human fall detection in videos via boosting and fusing statistical features of appearance, shape and motion dynamics on Riemannian manifolds with applications to assisted living. Comput. Vis. Image Underst. **148**, 111–122 (2016)
13. Dubey, R., Ni, B., Moulin, P.: A depth camera based fall recognition system for the elderly. In: Campilho, A., Kamel, M. (eds.) ICIAR 2012, Part II. LNCS, vol. 7325, pp. 106–113. Springer, Heidelberg (2012)
14. Hu, M.K.: Visual pattern recognition by moment invariants. IRE Trans. Inf. Theory **8**(2), 179–187 (1962)
15. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM **24**(6), 381–395 (1981)
16. Labayrade, R., Aubert, D., Tarel, J.P.: Real time obstacle detection in stereovision on non flat road geometry through V-disparity representation. In: IEEE Intelligent Vehicle Symposium, vol. 2, pp. 646–651. IEEE (2002)
17. Toyama, K., Krumm, J., Brumitt, B., Meyers, B.: Wallflower: principles and practice of background maintenance. In: Proceedings of IEEE International Conference on Computer Vision, pp. 255–261 (1999)
18. Pramerdorfer, C.: Evaluation of kinect sensors for fall detection. In: Proceedings of IASTED Conference on Signal Processing, Pattern Recognition and Applications (2013)
19. Khoshelham, K., Elberink, S.: Accuracy and resolution of kinect depth data for indoor mapping applications. Sensors **12**(2), 1437–1454 (2012)
20. McFarlane, N., Schofield, C.: Segmentation and tracking of piglets in images. Mach. Vis. Appl. **8**(3), 187–193 (1995)
21. Beymer, D.: Person counting using stereo. In: Proceedings of Workshop on Human Motion, pp. 127–133 (2000)
22. Harville, M.: Fast, integrated person tracking and activity recognition with plan-view templates from a single stereo camera. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 398–405 (2004)
23. Muñoz Salinas, R.: A Bayesian plan-view map based approach for multiple-person detection and tracking. Pattern Recogn. **41**(12), 3665–3676 (2008)
24. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
25. Kuhn, H.W.: The Hungarian method for the assignment problem. Naval Res. Logistics Q. **2**(1), 83–97 (1955)
26. Papadimitriou, C., Steiglitz, K.: Combinatorial Optimization: Algorithm and Complexity. Prentice Hall, Upper Saddle River (1982)
27. Kalman, R.E.: A new approach to linear filtering and prediction problems. Trans. ASME J. Basic Eng. **82**(D), 35–45 (1960)