A Viewpoint-Independent Statistical Method for Fall Detection

Zhong Zhang, Weihua Liu, Vangelis Metsis, and Vassilis Athitsos
University of Texas At Arlington, Texas, United States

Abstract

The goal of a fall detection system is to automatically detect cases where a human falls and may have been injured. We propose a statistical method based on Kinect depth cameras, that makes a decision based on information about how the human moved during the last few frames. Our method proposes novel features to be used for fall detection, and combines those features using a Bayesian framework. Our experiments explicitly evaluate the ability of our method to use training data collected from one viewpoint, in order to recognize falls from a different viewpoint. We obtain promising results, on a challenging dataset, that we have made public, and that contains, in addition to falls, several similar-looking events such as sitting down, picking up objects from under the bed, or tying shoelaces.

1. Introduction

In this paper, we propose a statistical method to detect cases where a human falls and may have been injured. Detection is based on depth video captured using Kinect cameras. Compared to existing vision-based fall detection methods, what differentiates our method is that it combines viewpoint invariance, simple system setup, and statistical decision making (as opposed to using hardcoded thresholds).

Several approaches have been proposed for fall detection, some recent reviews include [5, 13]. Several existing methods use non-vision sensors, such as the accelerometer [6, 7, 9], oftentimes combined with other devices such as gyroscopes [9] and microphone [7]. However, these methods require subjects to actively cooperate by wearing the sensors, which can be problematic and possibly uncomfortable (e.g., wearing sensors while sleeping, to detect falls during a night trip to the restroom). Our method is less intrusive, as all information is collected from cameras.

Several vision-based methods have been proposed for fall detection. Multi-camera calibrated systems [1, 2, 3] extract and use 3D features for fall detection. However, those methods require time-consuming external camera calibration. When a single camera is moved, the system needs to be re-calibrated. Our method can use a single camera, or multiple, uncalibrated cameras, to cover more of the living space.

Some vision-based methods use 2D appearance-based features to detect falls [10, 12, 14, 15]. Such methods can be used with a single camera, but they are viewpoint-dependent. Moving a camera to a different viewpoint (especially a different height from the floor) would require collecting new training data for that specific viewpoint. In our method, minimal effort is required to adjust the system to a new viewpoint.

Depth cameras provide 3D information without requiring calibration of multiple cameras. Depth cameras for fall detection are used in [4, 8, 11], and those three methods are the most related to our method. However, those three methods only use two features, namely distance from the floor and acceleration, and make a decision by applying hardcoded thresholds to individual features. In our method, we propose three additional features that improve accuracy, and decisions are made probabilistically, incorporating information from all features before making a hard decision.

An additional contribution in this paper is our experimental protocol, whereby all training data are collected from a specific viewpoint, and all the test data are collected from another viewpoint, several meters away from the training viewpoint. We believe that this new evaluation protocol is a useful approach for measuring the robustness of the system to displacements of the camera. Furthermore, we have made our dataset available online, for use by other researchers.

2 Image and World Coordinates

In our experimental setup, two Kinect depth cameras are set up at two corners of a simulated apartment. The reason for using two Kinects is simply to cover more of the living space, as the range of a Kinect is about $4m$. Figure 1 shows our simulated home environment.

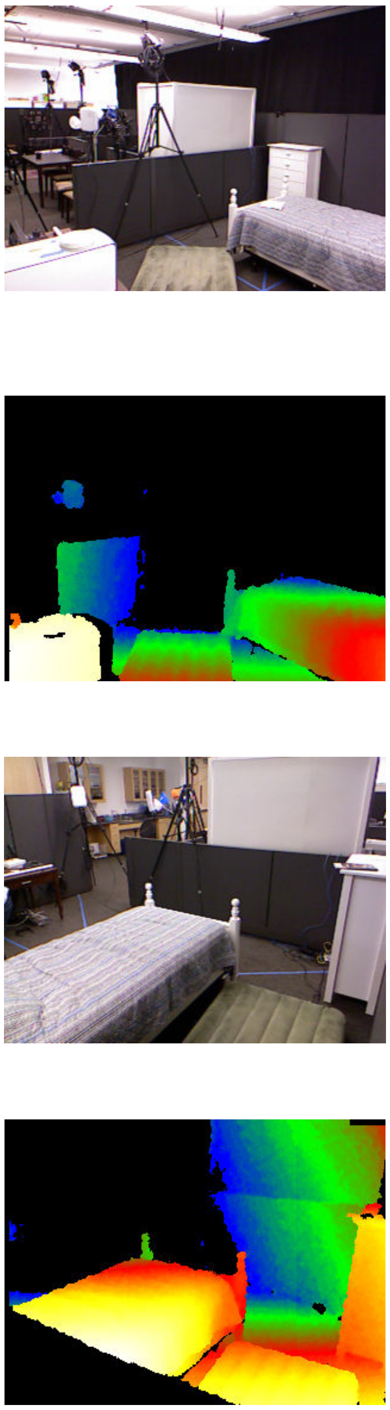
Figure 1: Our simulated apartment, seen from the two viewpoints that we used to collect videos. For each viewpoint we show a color image and a depth image. Depth images are color-coded so that: white indicates small depth values, and yellow, orange, red, green, blue indicate progressively larger depth values. Black indicates invalid depth.

A total of 12 sequences from six subjects and two views were acquired with a resolution of 320 x 240 pixels (31614 frames, about 3160 seconds of video).

Before processing the video collected from a camera, we need to compute three internal calibration parameters of the camera: x_0, y_0, and m. Parameters x_0 and y_0 are respectively the horizontal and vertical pixel coordinates of the principal point (also known as image center). Parameter m is a scaling factor mapping lengths measured in pixel units to lengths measured in another unit, such as millimeters. We note that precomputing x_0, y_0, m only needs to be once per camera, and those three parameters do not change when the camera moves.

Let $\mathbf{x_i} = [x_i, y_i, z_i]$ denote a pixel on the depth image, with location $[x_i, y_i]$ and depth value z_i. Using the three parameters, m, x_0 and y_0 discussed above, we can compute camera-centered 3D coordinates $\mathbf{x_k} = [x_k, y_k, z_k]$ for $\mathbf{x_i}$, as:

$$x_k = (x_i - x_0) * m * z_i \tag{1}$$
$$y_k = (y_i - y_0) * m * z_i \tag{2}$$
$$z_k = z_i \tag{3}$$

The above equations map pixels to locations in a camera-centered 3D coordinate system, whose origin is the camera pinhole and whose axes are defined based on the Kinect's 3D orientation: the x and y axes are the same as for the image, and the z axis is perpendicular to the image plane. Those Kinect-centered coordinates can easily be mapped to conventional 3D world coordinates, where the y axis corresponds to height. To do this mapping, we only need the user to click on two points of a vertical line on the image. If the camera moves (which we do not expect to happen on a daily basis), then the user simply needs to click again on two points, to reinitialize the system.

After we compute the height for every pixel in the image, the floor level can be easily computed, using the lowest value of heights observed in the image. To reduce influence by outliers, we use the 2.5-th percentile of heights as the floor level. Figure 2 shows an example of floor level detection.

3 Person Detection and Feature Extraction

We identify where the person is by performing background subtraction. The background depth map $B(x, y)$ is reinitialized every time the system observes a few frames without any motion, and thus can easily adapt to changes in the apartment setup. Let $D_j(x, y)$ denote the

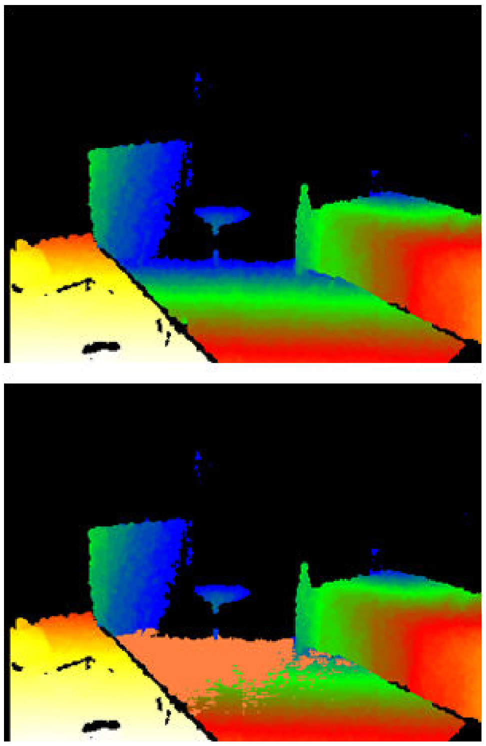

Figure 2: Result of floor level detection. Top: a background depth image. Bottom: the same image as on the top, with an orange color superimposed on pixels with the lowest 5% values in height. Note: as we only show the pixels with the lowest 5% values in height, and the floor is expected to contain more than 5% of the pixels in the image, we do not expect to see the entire floor detected. The goal of floor detection is NOT to detect the entire floor, but to reliably detect a part of the floor, without the detection including non-floor parts.

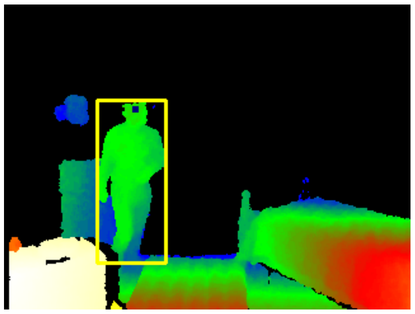

Figure 3: Result of person detection using background subtraction.

depth of pixel (x, y) at the j-th frame. We define a binary mask $M_j(x, y)$ that identifies regions with motion, by simply checking whether $D_j(x, y)$ and $B(x, y)$ are different by more than a threshold T, where the value of T is chosen using training data.

Once binary image M_j has been computed, we identify the largest connected component of M_j. That component is used by our system as the location of the person. Figure 3 shows an example output of this person detection method, which in general works near flawlessly in our dataset.

From the pixels identified by binary mask M_j, we identify the top 5% pixels with largest values in the y-coordinate, and we use the median of those values as an estimate of the height of the person's head. Since we have already computed the level of the floor, we express the height of the head in terms of distance from the floor.

Our fall detection system runs in real-time. When observing the j-th frame, the system evaluates whether a fall has just concluded at that frame. To do that, the system evaluates a range of frames $R_j = \{j - d_{\max}, \ldots, j - d_{\min}\}$ as possible start frames for the fall. Values $d_{\min}$ and $d_{\max}$ are determined using training data.

For every $i \in R_j$, features are extracted from the video sequence between frame i and frame j, and our classifier determines whether those features correspond to a fall event. Let the sequence of head positions in every frame be denoted as $[h_i, ..., h_j]$. The five features that we extract are the following:

Duration. The duration of a fall in frames, denoted as f_1, is defined simply as: $f_1 = j - i + 1$.

Total head drop. This is the total change of head height during the fall, denoted as f_2, and defined as: $f_2 = h_i - h_j$.

Maximum speed. This is the largest drop in head height, from one frame to the next, denoted as f_3, and defined as: $f_3 = \max_{m \in \{i+1,...,j\}}(h_{m-1} - h_m)$.

Smallest head height. Denoted as f_4, this is defined simply as $f_4 = \min(h_i, ..., h_j)$.

Fraction of frames where head drops. Denoted as f_5, this is simply the percentage of frames, between $i+1$ and j, where the head has a smaller height than in the previous frame.

4 Classifier Training and Application

The training process is both user-independent and viewpoint-independent. Our training videos have been recorded from two Kinects, placed at two different viewpoints, and recording at different times (i.e., there is no video captured simultaneously from both Kinects). Six subjects appear in the videos collected from each

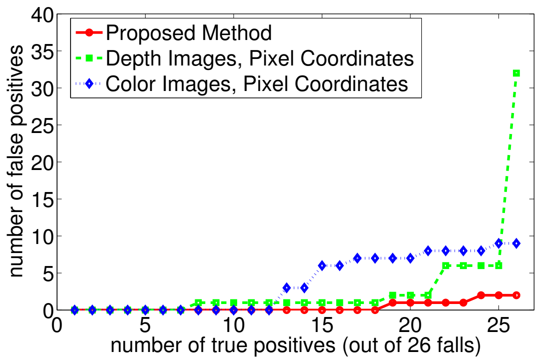

Figure 4: Results of the proposed method, as well as results using two variations: a variation where height is measured in pixel coordinates (as opposed to vertical distance from floor level), and a variation where color images instead of depth images are used.

viewpoint. To train the classifier that will be used for a specific subject and viewpoint, we use as training data only videos of other subjects from the other viewpoint. Thus, to classify a subject's actions as seen from a specific viewpoint, the system does not use training data neither from the same subject nor from the same viewpoint.

As described in Section 3, any frame j is a candidate end frame for a fall, with candidate start frames ranging from $j - d_{\min}$ to $j - d_{\max}$. The correct start and end frames of real falls are manually annotated, and define the positive training data. All candidate start/end frame combinations that do not overlap more than 30% with a real fall are used as negative training data.

We assume that the five features of a candidate event are conditionally independent, given the class of the event (fall or non-fall). For the positive examples, we model each feature distribution as a Gaussian, as we have very few positive samples (10 to 14 training samples for each test video, since we exclude data from the same viewpoint or same subject as in the test video). For the negative examples, we model each feature distribution nonparametrically as a histogram, since we have tens of thousands of negative examples, a number that is quite adequate for estimating a nonparametric one-dimensional distribution.

At runtime, given current frame j, and a possible start frame i, we compute the five features $[f_1, f_2, f_3, f_4, f_5]$. Using the positive and negative distributions that we have computed at training, we compute the probability of a fall event using a straightforward application of Bayes rule. We compare that probability with a threshold θ to make a final decision as to whether the event is fall or non-fall. Setting different values of θ we get different rates of true positives and false positives, as shown in the experiments.

5 Experiments

There are 10,400 frames and 12 real falls in videos from the first viewpoint, and 21,214 frames and 14 real falls in videos from the second viewpoint. Our subjects also performed a total of 61 actions that tend to produce features similar to those of a fall event, namely: 23 examples of picking up something from the floor, 12 cases of sitting on the floor, 10 examples of tying shoelaces, 9 examples of lying down on the bed, 5 examples of opening/closing a drawer at floor level, 1 example of jumping on the bed, and 1 example of lying on the floor. Figures 5 and 6 show respectively an example of a person lying down on the floor, on an air mattress, and a person jumping on the bed. Our entire dataset and annotations are publicly available at:

`http://vlm1.uta.edu/~zhangzhong/fall_detection/`

As mentioned in Section 4, our experiments are both user-independent (to recognize the actions of the user, no training data from that same user is used) and viewpoint-independent (to recognize actions observed from a specific viewpoint, no training data from the same viewpoint is used). Also, because the fall action is a continuous process, if two detected fall events overlap by more than 30%, the system rejects the detection with the lower score.

Figure 4 shows the results. For comparison, we also include the results of two variations: In the first variation, the height of the head is simply the y pixel coordinate, as opposed to actual vertical distance from floor level. The second variation is the method we described in [14], and uses color images, as opposed to depth images, while height is again measured using the y pixel coordinate. We note that moving from color to depth images significantly improves accuracy (because background subtraction is much more accurate), and measuring the actual vertical distance from the head to the floor also improves accuracy. The proposed method identifies all 26 fall events with only 2 false alarms.

The two false alarms are shown on Figures 5 and 6. In Figure 5, the person lies down on the floor. In Figure 6, the person jumps to the bed, and the lowest head vertical coordinate is almost the same as in a real fall. In future work, we hope to further improve accuracy, using additional training data (especially positive examples), and identifying additional features.

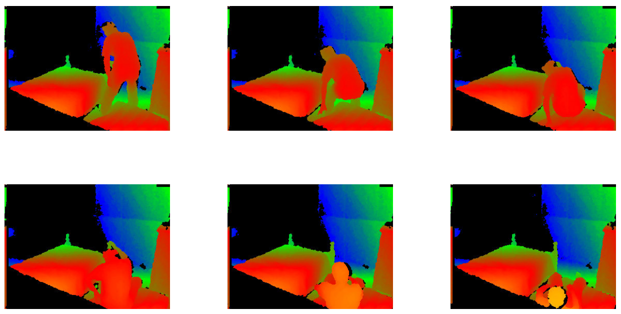

Figure 5: Example frames from a false alarm, showing a person lying on the floor.

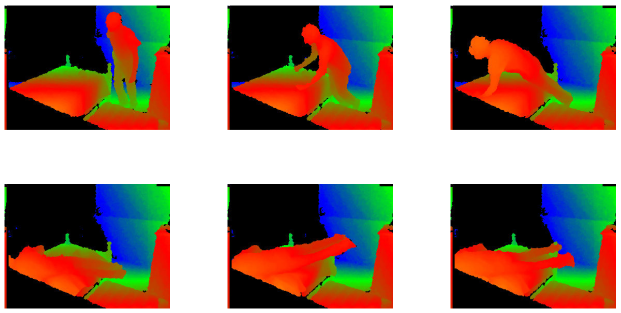

Figure 6: Example frames from a false alarm, showing a person jumping on the bed.

Acknowledgments

This work was partially supported by NSF grants IIS-0812601, IIS-1055062, CNS-1059235, ECCS-1128296, CNS-0923494, and CNS-1035913.

References

[1] D. Anderson, R. H. Luke, J. M. Keller, M. Skubic, M. Rantz, and M. Aud. Linguistic summarization of video for fall detection using voxel person and fuzzy logic. *Computer Vision and Image Understanding*, 113:80–89, January 2009.

[2] E. Auvinet, F. Multon, A. St-Arnaud, J. Rousseau, and J. Meunier. Fall detection with multiple cameras: An occlusion-resistant method based on 3-d silhouette vertical distribution. *Information Technology in Biomedicine*, 15:290–300, 2011.

[3] R. Cucchiara, A. Prati, and R. Vezzani. A multi-camera vision system for fall detection and alarm generation. *Expert Systems*, 24:334–345, 2007.

[4] G. Diraco, A. Leone, and P. Siciliano. An active vision system for fall detection and posture recognition in elderly healthcare. In *Design, Automation & Test in Europe Conference & Exhibition*, March 2010.

[5] M. Kangas, A. Konttila, P. Lindgren, I. Winblad, and T. Jamsa. Comparison of low-complexity fall detection algorithms for body attached accelerometers. *Gait & Posture*, 28:285–291, 2008.

[6] C.-F. Lai, S.-Y. Chang, H.-C. Chao, and Y.-M. Huang. Detection of cognitive injured body region using multiple triaxial accelerometers for elderly falling. *IEEE Sensors Journal*, 11:763–770, 2011.

[7] A. Leone, G. Diraco, C. Distance, P. Siciliano, M. Malfatti, L. Gonzo, M. Grassi, A. Lombardi, G. Rescio, P. Malcovati, V. Libal, J. Huang, and G. Potamianos. A multi-sensor approach for people fall detection in home environment. In *IEEE Workshop on European Conference Computer Vision for Multi-camera and Multi-modal Sensor Fusioin Algorithms and Applications*, 2008.

[8] A. Leone, G. Diraco, and P. Siciliano. Detecting falls with 3d range camera in ambient assisted living applications: A preliminary study. *Medical Engineering & Physics*, 33:770–781, 2011.

[9] Q. Li, J. A. Stankovic, M. A. Hanson, T. Barth, J. Lach, and G. Zhou. Accurate, fast fall detection using gyroscopes and accelerometer-derived posture information. In *Werable and Implantable Body Sensor Networks*, 2009.

[10] F. Nater, H. Grabner, T. Jaeggli, and L. V. Gool. Tracker trees for unusual event detection. In *ICCV Workshop on Visual Surveillance*, 2009.

[11] C. Rougier, E. Auvinet, J. Rousseau, M. Mignotte, and J. Meunier. Fall detection from depth map video sequences. In *International Conference on Smart Homes and Health Telematics*, 2011.

[12] C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau. Robust video surveillance for fall detection based on human shape deformation. *IEEE Transactions on Circuits and Systems for Video Technology*, 21:611–622, 2011.

[13] T. Shany, S. J. Redmond, M. R. Narayanan, and N. H. Lovell. Sensors-based werable systems for monitoring of human movement and falls. *IEEE Sensors Journal*, pp:1–13, 2011.

[14] Z. Zhang, E. Becker, R. Arora, and V. Athitsos. Experiments with computer vision methods for fall detection. In *Conference on Pervasive Technologies Related to Assistive Environments (PETRA)*, 2010.

[15] A. Zweng, S. Zambanini, and M. Kampel. Introducing a statistical behavior model into camera-based fall detection. In *Proceedings of the 6th international conference on Advances in visual computing - Volume Part I*, 2010.