Evaluating Depth-Based Computer Vision Methods for Fall Detection Under Occlusions

Zhong Zhang, Christopher Conly, and Vassilis Athitsos

Computer Science and Engineering Department
University of Texas at Arlington
Arlington,Texas
USA

Abstract. Falls are one of the major risks for seniors living alone at home. Fall detection has been widely studied in the computer vision community, especially since the advent of affordable depth sensing technology like the Kinect. Most existing methods assume that the whole fall process is visible to the camera. This is not always the case, however, since the end of the fall can be completely occluded by a certain object, like a bed. For a system to be usable in real life, the occlusion problem must be addressed. To quantify the challenges and assess performance in this topic, we present an occluded fall detection benchmark dataset containing 60 occluded falls for which the end of the fall is completely occluded. We also evaluate four existing fall detection methods using a single depth camera [1–4] on this benchmark dataset.

1 Introduction

The growing population of seniors would benefit from systems that include fall detection functionality. To automatically detect falls in a home environment can help ensure the safety of elderly people living alone. In this work we focus on the task of vision-based occluded fall detection. An occluded fall occurs when an object, such as a bed, blocks the sensor's view of the end of the fall, and thus the entire fall is not visible. These occlusions frequently occur at a home environment since a room contains furniture and objects that could be placed between the subject and the camera. Kinect cameras are used to capture the occluded fall benchmark dataset, and the detection is thus based on depth videos.

Several sensor-based approaches have been proposed for fall detection, some recent reviews include [5, 6]. An accelerometer [7–9] is the most commonly used sensor, and it is often combined with other devices such as gyroscopes [9] and microphone [8]. These approaches do not suffer from the occlusion problem. 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, detecting falls during a night trip to the restroom).

Several vision-based methods have been proposed for fall detection. They can be broadly classified into two categories: 2D-based and 3D-based. [10–13] use 2D appearance-based features to detect falls. These methods use a single camera and

are viewpoint-dependent. Moving a camera to a different viewpoint (especially to a different height from the floor) would require collecting new training data for that specific viewpoint. The 3D features for fall detection can be extracted from a calibrated multi-camera system [14–16] or from depth cameras [1, 3, 4, 2]. Multi-camera calibrated systems require time-consuming external camera calibration while depth cameras based systems do not.

Most of these methods did not report any information about the robustness of the system to occlusions aside from [16] and [2]. Auvinet et al. [16] used multiple cameras to cover the whole space so that no occlusion occurred for at least one of these cameras. Thus, the total volume information in the scene can be reconstructed based on the multiple-cameras network. This solution, however, is expensive and difficult to set up (multiple calibrated and synchronized cameras are needed). Rougier et al. [2] classified each frame into two classes: the person is totally occluded or not. If it is an occluded case, the velocity of the person just before the occlusion occurs is used to determine whether a fall has occurred. In the non-occluded case, the height of the person's top relative to the floor plane is used to make a decision.

The focus of this paper is on the evaluation of existing depth-based computer vision approaches for fall detection in the case of occlusions. In particular, we make two contributions:

- A new, extensive and annotated dataset of depth videos, including fall events performed by several individuals, without occlusions and with occlusions. The dataset also includes several non-fall events that can lead to false alarms, such as tying shoelaces or lying on the floor. This dataset is freely available online, together with annotations marking the beginning and end of each fall event (`http://sites.google.com/site/occlusiondataset`).
- An evaluation of four recent methods for fall detection using depth cameras [1–4] on this new dataset, using a user-independent and viewpoint-independent experimental protocol (All training data are collected from a specific viewpoint, and all the test data are collected from another viewpoint). We compare the performance of these methods separately for unoccluded observations and for occluded observations.

2 Scene Setup

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 collect data from two different viewpoints so that the viewpoint-independent experimental protocol can be conducted. Figure 1 and Figure 2 show the scene setups for recording non-occluded and occluded fall dataset respectively. We can easily see that the two viewpoints in Figure 1 are different from the two viewpoints in Figure 2.

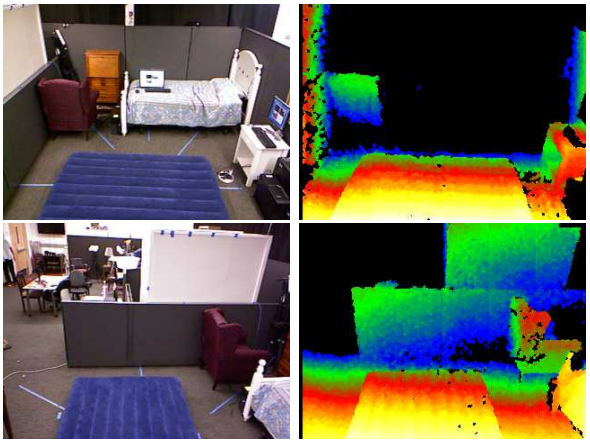

Fig. 1. Our simulated apartment, seen from the two viewpoints that is used to collect non-occluded fall videos. For each viewpoint we show a color image and a depth image. Depth images are color-coded so that: white indicates small depth values, and yellow, orange, red, green, blue indicate progressively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

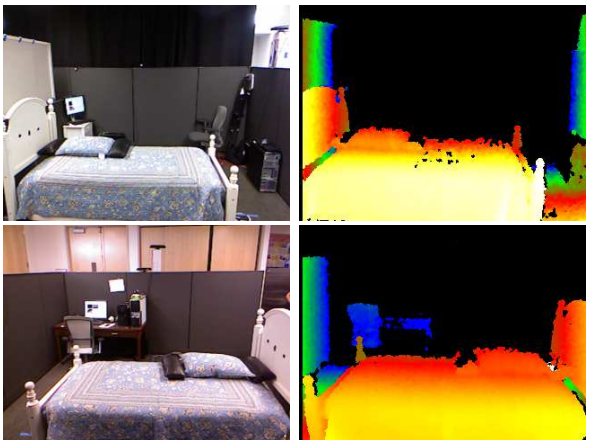

Fig. 2. Our simulated apartment, seen from the two viewpoints that is used to collect occluded fall videos. For each viewpoint we show a color image and a depth image. Depth images are color-coded so that: white indicates small depth values, and yellow, orange, red, green, blue indicate progressively larger depth values. Black indicates invalid depth.

3 Four Existing Fall Detection Methods

In this section, we briefly introduce four existing fall detection methods [1–4] that use a single depth camera. Since the distance of the person's top relative to the floor plane is the common feature used in all of these methods, we describe how to calculate this height first. To compute the distance between the top of the person and the floor plane, we first need to estimate the equation of the floor plane, and it can be used to compute the distance between a 3D point of the scene and the floor plane.

3.1 Floor Plane Estimation

The floor plan can be represented as the following equation:

$$Ax + By + Cz + D = 0 \tag{1}$$

We obtain parameters A, B, C, D directly using the Microsoft Kinect SDK, which has built-in floor estimation functionality. Furthermore, the Kinect SDK provides normalized values for parameters A, B, C, D so that the physical interpretation of D is the height of the camera from the floor. Figure 3 shows two examples of floor detection. The pixel is classified as floor if its height value is less than 2cm. It is worth noting that the floor plane detection method in [1] is not fully automatic and requires the user to click on two points in the depth image to specify the vertical orientation in world coordinates.

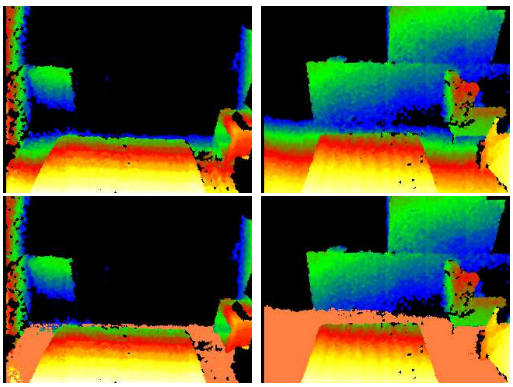

Fig. 3. Result of floor level detection. Top: a background depth image. Bottom: the same image as on the top, with an orange color superimposed on pixels with height value less than 2cm.

3.2 Person Detection and Height Calculation

In our environment, we can assume that the only moving object is the person. To detect moving objects, we use background subtraction on depth images. The background depth map B is obtained by using the average of the last few frames where no significant motion was observed.

Let $D_n(x, y)$ denote the value at pixel coordinates (x, y) for the n-th depth frame. By comparing $D_n(x, y)$ with the background depth map $B(x, y)$, the moving parts of the current frame can be easily extracted. In particular, using a system-specific threshold T ($T = 200$ in our implementation), a pixel is determined to belong to a moving object if its depth value differs from the corresponding background value by more than T:

$$M_n(x,y) = \begin{cases} 1 & if \quad |D_n(x,y) - B(x,y)| > T \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Once $M_n(x, y)$ has been computed for all pixels of frame n, connected component analysis is used to identify the largest connected component with values $M_n(x, y) = 1$. We call that component the *person region*.

To estimate the distance from the top of the person to the floor, we first identify the top region of the person as seen in the depth image. Once the person region has been detected in the image, the pixels of that region are sorted based on their row coordinate, and the value at the highest 5 percentile is selected as the "top row" of the region. We denote that "top row" as $T_i^y(n)$. By examining pixels in the person region that are located at the "top row", we identify the median column for those pixels and denote it as $T_i^x(n)$. We can then read the depth value at pixel $[T_i^x(n), T_i^y(n)]$ and denote it as $T^z(n)$.

Given $[T_i^x(n), T_i^y(n), T^z(n)]$, we compute the Kinect-centered world coordinates for that point using Equations 3, and from those coordinates, we compute the distance from the top of the person to the ground plane using Equation 4, where $[w, h]$ is the width and height of the depth image in pixels, and m_x (1.12032) and m_y (0.84024) are fixed parameters specified in the Kinect documentation provided by Microsoft.

$$\begin{aligned} T_k^x(n) &= (\frac{T_i^x(n) - 1}{w - 1}) * m_x * T^z(n) \\ T_k^y(n) &= (\frac{T_i^y(n) - 1}{h - 1}) * m_y * T^z(n) \\ T_k^z(n) &= T^z(n) \end{aligned} \tag{3}$$

$$y_w = \frac{|AT_k^x(n) + BT_k^y(n) + CT_k^z(n) + D|}{\sqrt{A^2 + B^2 + C^2}} \tag{4}$$

3.3 Fall Detection

The authors of [2–4] calculate the height as the distance of the centroid of the person's silhouette to the floor plane. The height of this centroid, however, might

not be accurate when the person is partially occluded (Figure 4). Instead, we calculate the distance of the person's top to the floor plane as described in section 3.2.

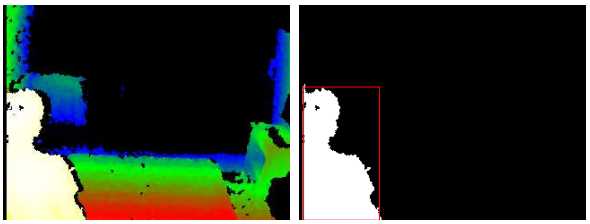

Fig. 4. The left side shows an example of partial occlusion, while the right side shows the person detection result with a red rectangle indicating the person's location in the depth image. It can easily be seen that the centroid of the silhouette is not the actual centroid of the person.

- In method [2], the human height relative to the ground is used to detect (non-occluded or partially occluded) falls. When the end of an action is totally occluded by furniture, an analysis of the 3D body velocity prior to occlusion allows detection of the fall.
- The method of [3] uses two rules to describe a fall: the distance of the top of the person to the floor plane being lower than a prefixed value; an unchangeable situation (negligible movements of the person) for at least 4 seconds.
- The method of [4] is also based on two rules: the distance of the top of the person to the floor plane, decreasing 0.40m within a window of about 900ms; the people's silhouette movement remaining negligible for a time window of about 4s.
- Zhang et al. [1] extract five features from a candidate event and use a pre-trained naive Bayesian classifier on these features to determine whether the candidate is a fall or not.

4 Experiments

4.1 Dataset Description

We created a non-occlusion dataset, **EDF**, and an occlusion dataset, **OCCU**, using Kinect cameras for XBOX 360 with the Microsoft Kinect for Windows SDK Beta at a frame rate of about 30fps. Each of the 5 subjects performed a non-occluded fall along eight directions in each viewpoint in the **EDF** dataset. Each of the 5 subjects performed six occluded falls in each viewpoint in the **OCCU** dataset. Figure 5 shows one fall example in every direction. In the

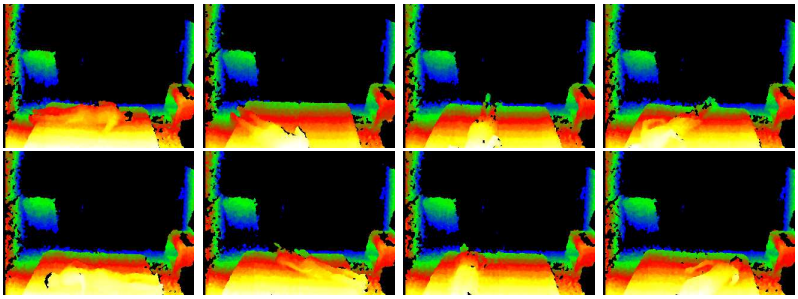

Fig. 5. A person falls down along eight different directions

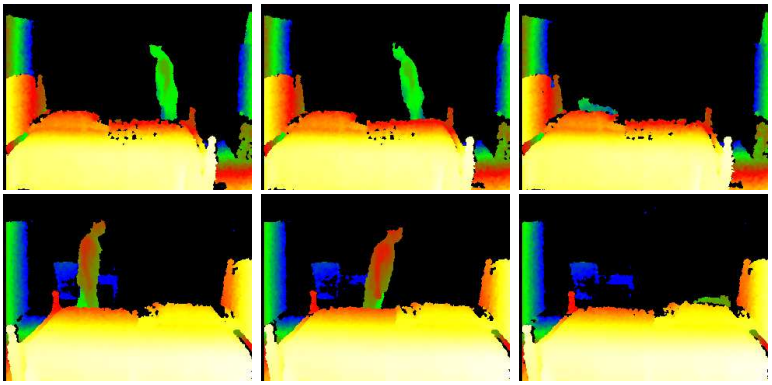

Fig. 6. Examples of the occluded fall. The top row shows an occluded fall in the first viewpoint while the bottom row shows an occluded fall in the second viewpoint.

OCCU dataset, the end of the fall is completely occluded by a bed. Figure 6 shows an example of an occluded fall in each viewpoint.

The **EDF** dataset is comprised of 25,881 frames and 40 real falls in videos from the first viewpoint, and 24,497 frames and 40 real falls in videos from the second viewpoint. The two viewpoints were recorded at the same time, and thus every event was recorded simultaneously from both viewpoints. Our subjects also performed a total of 30 actions that tend to produce features similar to those of a fall event, namely: 10 examples of picking up something from the floor, 10 cases of sitting on the floor and 10 examples of lying down on the floor.

The **OCCU** dataset includes 25,618 frames and 30 totally occluded falls in videos from the first viewpoint, and 23,703 frames and 30 totally occluded falls in videos from the second viewpoint performed by the same subjects. Each viewpoint was recorded at separate times from the other viewpoint, and thus we had no instances where the same events were recorded simultaneously from both viewpoints. Our subjects also performed a total of 80 actions that tended to produce features similar to those of a fall event, namely: 20 examples of picking up something from the floor (all of them are non-occluded), 20 examples of sitting

on the floor (all of them are non-occluded), 20 examples of tying shoelaces (all of them are non-occluded), and 21 examples of lying down on the floor(all of them are totally occluded at the end frame).

The **OCCU1** dataset is a subset of **OCCU**. To generate the **OCCU1** dataset, we removed all examples of lying down on the floor from **OCCU** and created **OCCU1** from the remaining examples. As the primary use of this application is intended for elderly people, and it is unlikely that an elderly person would all of a sudden lie down on the floor, we found it is necessary to conduct a separate evaluation on the dataset by excluding the "lying down on the floor" events.

4.2 Results

As mentioned in Section 1, our experiments are both user-independent (to recognize the actions of a 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 a fall is detected in the current frame, then no fall would be reported in the next 250 frames.

Here we evaluate four depth-based fall detection methods on the three datasets [1–4]. We have implemented these competitive methods ourselves. Since the thresholds provided by the authors may not be the best choice for our data, we searched and identified the best thresholds for them. The methods of [2, 1] need a training step while the methods of [3, 4] do not, since they are rule based. The experiments using the methods of [2, 1] strictly follow the user-independent and viewpoint-independent protocols.

The methods of [1, 3, 4] did not discuss how to handle occluded falls. When the person is totally occluded by the bed in our simulated environment, the height of the person's top can not be calculated. For these cases, we simply assign the height value of these frames to the last frame where the person is visible. Rougier et al. [2] classified every frame into two categories: whether the person is visible in the current frame or not. When the person is visible, the human top height relative to the ground is used to detect (not occluded or partially occluded) falls. When the end of an action is totally occluded by the bed, an analysis of the 3D body velocity prior to occlusion allows detection of the fall.

Results on EDF dataset Five subjects appear in the videos collected from each viewpoint in **EDF** dataset and there are two viewpoints in total. 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 from the same subject or from the same viewpoint.

Figure 7 shows the results on dataset **EDF**. The method proposed by Zhang et al. is significantly more accurate than those of the competitors. If we choose

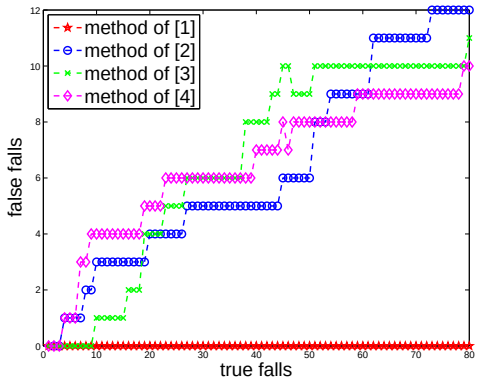

Fig. 7. Results of four evaluated methods [1–4] on **EDF** dataset.

thresholds for each method so as to detect all of the actual falls, the method of [1] produces zero false positive, compared to ten false positives by the best of the competitors [4]. A false alarm reported by [4] is shown on Figure 8. In Figure 8, the person lies down on the floor.

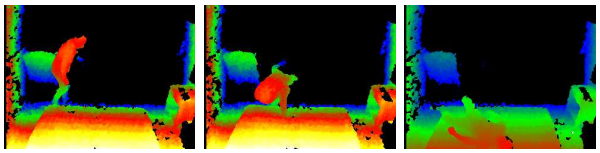

Fig. 8. Example frames from a false alarm, showing a person lying on the floor.

Results on OCCU dataset We use the **EDF** dataset to train the models for methods [2, 1] for tests on the **OCCU** dataset. We ensure user independence and viewpoint independence by using different subjects and viewpoints in **EDF** and **OCCU**.

Two parameters are used in the method of [2]: a threshold for the height of the person's top, T, and a threshold for the body velocity, V. If there is no occlusion, the height of the person's top relative to the ground is used to detect a potential fall; if an occlusion occurs, the body velocity just before the occlusion occurs is analyzed to try to determine whether a fall has occurred. Since there are no non-occlusion falls in the **OCCU** dataset, we set a low enough value for T to eliminate all non-occlusion false falls caused by such events as sitting down on the floor, tying the shoelaces, etc., and do not need to worry about missing the detection of any true fall (all of the true falls are totally occluded and thus would

be handled by velocity checking). In this way, we favor method [2] over other competitive methods, since they treat every candidate event, either occluded fall or non-occluded fall, equally.

The result is shown in the left side of Figure 9. The method of [2] is significantly better than other methods. But in detecting all the actual falls, it also reports 21 false positives. Most of these false positives are lying down on the floor. Figure 10 shows an example of this action. This action has similar features to a true occluded fall, and thus it is also hard to be correctly classified by all of other evaluated methods. The methods of [3, 4, 1] also fail on other actions, like picking up something from the floor and tying shoelaces (Figure 11).

Results on OCCU1 dataset The training protocol and parameters selection are the same as the experiment conducted on **OCCU** dataset (section 4.2). The result is shown in the right side of Figure 9. The method of [2] is still significantly better than other methods. To detect all the actual falls, only one false positive is reported. This verifies the fact that most of the false positives produced by the method of [2] on the dataset **OCCU** come from the event of lying down on the floor. By comparing the experimental results on **OCCU** and **OCCU1** dataset for other evaluated methods [3, 4, 1], we can clearly see that methods [3, 4, 1] also fail in most cases of lying down on the floor.

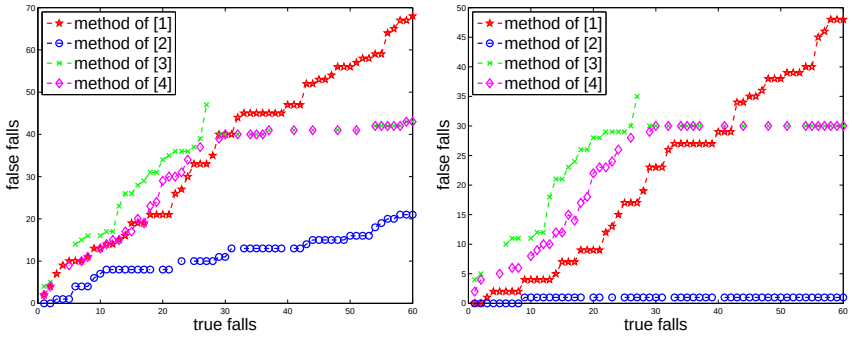

Fig. 9. Results of four evaluated methods [1–4] on **OCCU** dataset and **OCCU1** dataset respectively. The left side is about **OCCU** dataset while the right side is for **OCCU1** dataset

5 Discussion and Conclusion

For the non-occluded falls, the method proposed by Zhang et al. [1] achieves the best result, while for the occluded falls, the method proposed by Rougier et al. [2], which first classifies a candidate event into occluded case or non-occluded case

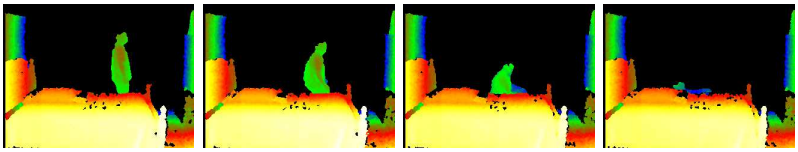

Fig. 10. Example frames from a false alarm, showing a person lying down on the floor.

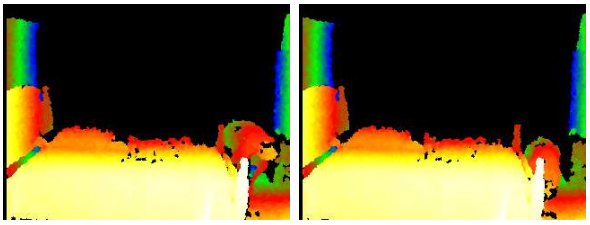

Fig. 11. The left side is an example of tying shoelaces and the right side is an example of picking up something from the floor.

and then checks it using the specific feature of its corresponding class, is better than other methods. We can conclude that it is necessary to handle occluded and non-occluded falls separately. Although the method of [2] is significantly better than the other three methods, we note that the velocity feature adopted by [2] is not adequate for distinguishing lying down from falling down.

For future work, we plan to explore more representative features, like acceleration, to make the method capable of distinguishing occluded fall from other similar occluded false events. At the same time, due to the low cost of the Kinect, it is practical to deploy several depth cameras, so as to cover most or all of the apartment area. In that scenario, the decisions from all the cameras are aggregated into a final decision for a candidate event. By doing so, we believe that the occluded cases can be handled more accurately.

Acknowledgments. This work was partially supported by National Science Foundation grants IIS-1055062, CNS-1059235, CNS-1035913, and CNS-1338118.

References

1. Zhang, Z., Liu, W., Metsis, V., Athitsos, V.: A viewpoint-independent statistical method for fall detection. In: Pattern Recognition (ICPR), 2012 21st International Conference on, IEEE (2012) 3626–3630
2. Rougier, C., Auvinet, E., Rousseau, J., Mignotte, M., Meunier, J.: Fall detection from depth map video sequences. In: International Conference on Smart Homes and Health Telematics. (2011)
3. Diraco, G., Leone, A., Siciliano, P.: An active vision system for fall detection and posture recognition in elderly healthcare. In: Design, Automation & Test in Europe Conference & Exhibition. (2010)

4. Leone, A., Diraco, G., Siciliano, P.: Detecting falls with 3d range camera in ambient assisted living applications: A preliminary study. Medical Engineering & Physics **33** (2011) 770–781
5. Kangas, M., Konttila, A., Lindgren, P., Winblad, I., Jamsa, T.: Comparison of low-complexity fall detection algorithms for body attached accelerometers. Gait & Posture **28** (2008) 285–291
6. Shany, T., Redmond, S.J., Narayanan, M.R., Lovell, N.H.: Sensors-based werable systems for monitoring of human movement and falls. IEEE Sensors Journal **pp** (2011) 1–13
7. Lai, C.F., Chang, S.Y., Chao, H.C., Huang, Y.M.: Detection of cognitive injured body region using multiple triaxial accelerometers for elderly falling. IEEE Sensors Journal **11** (2011) 763–770
8. Leone, A., Diraco, G., Distance, C., Siciliano, P., Malfatti, M., Gonzo, L., Grassi, M., Lombardi, A., Rescio, G., Malcovati, P., Libal, V., Huang, J., Potamianos, G.: 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)
9. Li, Q., Stankovic, J.A., Hanson, M.A., Barth, T., Lach, J., Zhou, G.: Accurate, fast fall detection using gyroscopes and accelerometer-derived posture information. In: Werable and Implantable Body Sensor Networks. (2009)
10. Nater, F., Grabner, H., Jaeggli, T., Gool, L.V.: Tracker trees for unusual event detection. In: ICCV Workshop on Visual Surveillance. (2009)
11. Rougier, C., Meunier, J., St-Arnaud, A., Rousseau, J.: Robust video surveillance for fall detection based on human shape deformation. IEEE Transactions on Circuits and Systems for Video Technology **21** (2011) 611–622
12. Zhang, Z., Becker, E., Arora, R., Athitsos, V.: Experiments with computer vision methods for fall detection. In: Conference on Pervasive Technologies Related to Assistive Environments (PETRA). (2010)
13. Zweng, A., Zambanini, S., Kampel, M.: 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)
14. Anderson, D., Luke, R.H., Keller, J.M., Skubic, M., Rantz, M., Aud, M.: Linguistic summarization of video for fall detection using voxel person and fuzzy logic. Computer Vision and Image Understanding **113** (2009) 80–89
15. Cucchiara, R., Prati, A., Vezzani, R.: A multi-camera vision system for fall detection and alarm generation. Expert Systems **24** (2007) 334–345
16. Auvinet, E., Multon, F., Saint-Arnaud, A., Rousseau, J., Meunier, J.: Fall detection with multiple cameras: An occlusion-resistant method based on 3-d silhouette vertical distribution. Information Technology in Biomedicine, IEEE Transactions on **15** (2011) 290–300
17. Baldonado, M., Chang, C.C., Gravano, L., Paepcke, A.: The stanford digital library metadata architecture. Int. J. Digit. Libr. **1** (1997) 108–121