A Video-based Algorithm for Elderly Fall Detection

Jared Willems[1], Glen Debard[2], Bart Vanrumste[2,3] and Toon Goedemé[1,4]

[1] De Nayer Instituut, Lessius university college, Sint-Katelijne-Waver, Belgium
[2] MOBILAB, Katholieke Hogeschool Kempen, Geel, Belgium
[3] SCD/ESAT, Katholieke Universiteit Leuven, Leuven, Belgium
[4] VISICS/PSI/ESAT, Katholike Universiteit Leuven, Leuven, Belgium

Abstract— **Falling is an important health issue, but how can we detect falls automatically? One possibility would be a camera-based system. In this paper, we present a grayscale video processing algorithm to detect falls in video data. Grayscale video data is chosen because it has several advantages which will be presented in this paper. We start with background subtraction after which we continue to shadow removal, ellipse fitting and fall detection based on fall angle and aspect ratio. When a fall is detected, we continue to fall confirmation using vertical projection histograms.**

Keywords— **elderly assisted living, fall detection, video processing, e-health.**

I. Introduction

Falling among elderly is an important current health issue. Approximately 28% to 35% of the adults aged 65 or more fall at least once a year. From this group, up to 50% falls two times or more [1]. Falling has severe consequences. It is one of the five most common causes of death amongst the elderly population and is an important cause of head injuries and fractures. Beside this physical aspect, falling has also a very high psychological impact, even if there aren't injuries. The fear of falling again, possibly laying on the ground for several hours or even days without help and the pain linked with that situation may make the elderly feel uncomfortable. These aspects in combination with the physical risks of falling (injuries, hypothermia or even death) make the need for a fall detection system obvious.

Fall detection can be handled by different ways. First of all, there is the well-known emergency button. The elderly only has to push the button in case of an emergency, such as a fall. This technique may be a first step to avoid an elderly laying on the ground for several hours, but doesn't cope with situations where the victim isn't able to push the alarm button for any reason. More advanced are the wearable sensor based devices which make use of accelerometers or gyroscope sensors to detect fall incidents. There is one important disadvantage to all these systems, the elderly should wear them. The elderly can forget to wear the device and may look at it as something uncomfortable. Therefore we need to offer another accurate and comfortable solution. In this paper, we will present a video processing algorithm that can detect falls. With this camera-based approach, the problems of wearable sensors are bypassed. In [3] an extensive literature study is presented on this matter.

The paper is structured as follows. Section II. will give a short description of a general camera-based fall detection system. The individual processing steps of the system are discussed in section III. where we will consecutively handle background subtraction, shadow detection and fall detection. After having explained our algorithm, we give a short description about the recording of our video dataset in section IV.. Section V. will show the results and a conclusion is formed in section VI..

II. System overview

In this section, we will give a general description of a camera-based fall detection system. We will focus only on the algorithm and not on the hardware used to capture the video data. A good overview about possible hardware configurations is presented by Debard et al. [4]. In this project, we prefer grayscale video input as this has some important advantages. First of all, a reduction in processing power and memory consumption is achieved considering grayscale video data. Another advantage is that when data has to be sent over a network, higher resolutions are possible than when using colour video sequences. A last important benefit are the costs related to camera's, a grayscale camera is cheaper than a colour camera.

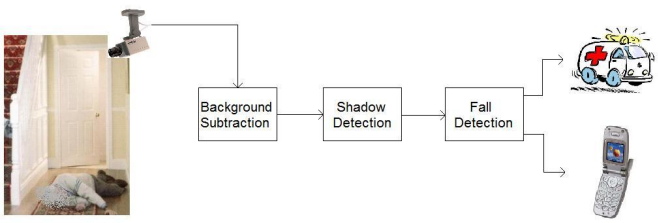

Fig. 1: System overview

A camera-based system contains some major parts, as represented in figure 1. First of all, we have the background subtraction step. In this stage, the person is detected.

A second stage in the algorithm is shadow detection. Here we will minimize shadows as shadows which are detected as moving object in the background subtraction step may influence further processing.

When we have determined our moving object, we model it with a best fit ellipse. In the third and last stage, we will use data of this ellipse and the foreground region (moving object) for fall detection.

All these steps are discussed in the following section.

III. Algorithm

A. Background subtraction

In this section we will handle background subtraction which is the first step in the algorithm as presented in figure 1. Background subtraction is used to identify the person in the image. The working principle is quite easy, a background model is estimated and then subtracted from the input frame. The resulting frame difference is the moving object and is classified as foreground. We opted for median filtering to estimate the background model as we found in the literature that this technique offers a good accuracy to calculation complexity trade-off. Although there are more complex and more accurate techniques such as Mixture of Gaussians, we believe that the method proposed in our solution give satisfying results.

Median filtering works as follows. We fill a buffer with the *N* last input frames. The estimated value of each pixel *B(x,y)* in the background model is the median for that pixel *(x,y)* calculated over all frames in the buffer [5].

In our experiments we use a circular buffer of 40 frames and add a new frame each 5 frames to reduce memory requirements and to increase processing speed.

Main advantage of this technique is the possibility of calculating a background estimate over a long period but a large buffer is needed. Disadvantages are the slower processing speed which increases if the buffer length increases and the memory consumption *(N x framesize)* attached to this method due to the buffer.

B. Shadow detection

After the background subtraction stage, the following step in our algorithm is shadow detection. Shadow detection is necessary because shadows move in conjunction with the object creating the shadow and thus are detected as moving object. It needs no further explanation that a moving object which isn't part of the person can corrupt the results. An intensive study on shadows and shadow detection in grayscale images is performed by Stauder et al. [6]. They observed that the edge between the umbra (shadow) and the part without shadow, called the penumbra, is a soft transition. Based on this feature, they propose a shadow detection technique. Although this technique may give good results, it is rather complex to compute. Another approach to the problem is presented by Jacques et al. [7]. This is the method we use in our system. The assumption is made that shadow pixels are darker (scaled) versions of corresponding pixels in the background estimate. To compute a first shadow pixel estimate, we use normalised cross correlation (NCC). For a pixel *(i,j)*, the NCC can be calculated as follows:

$$NCC(i,j) = \frac{ER(i,j)}{E_B(i,j)E_{T_{ij}}} \quad (1)$$

where

$$ER(i,j) = \sum_{n=-N}^{N}\sum_{m=-N}^{N} B(i+n,j+m)T_{ij}(n,m) \quad (2)$$

$$E_B(i,j) = \sqrt{\sum_{n=-N}^{N}\sum_{m=-N}^{N} B(i+n,j+m)^2} \quad (3)$$

$$E_{T_{ij}} = \sqrt{\sum_{n=-N}^{N}\sum_{m=-N}^{N} T_{ij}(n,m)^2} \quad (4)$$

Where I is the current frame, B is the background model, $T_{ij}(n,m)$ is a *(2N+1)*x*(2N+1)* neighbourhood around each pixel *I(i,j)*, $E_{T_{ij}}$ is the energy in this neighbourhood and $E_B(i,j)$ is the energy in a corresponding neighbourhood of the background image. To determine a first shadow pixel estimate, the NCC for a possible shadow pixel should be close to one, which means that there is a high correlation between the shadow pixel and the corresponding background pixel. Beside this correlation, the energy in the shadow pixel neighbourhood should be less than the energy of the corresponding background pixel neighbourhood.

$$NCC(i,j) \geq L_{ncc} \quad and \quad E_{T_{ij}} < E_B(i,j) \quad (5)$$

In these formulas, L_{ncc} is a fixed threshold close to one. Experiments showed that a threshold of 0.98 gave the best results for our system.

When only assessing the normalised cross correlation, a lot of the real moving object pixels are also classified as shadow pixels. We use the shadow refinement stage proposed by Jacques et al. to avoid this false shadow classification. The ratio *I(i,j)/B(i,j)* in a neighbourhood *R* around a possible shadow pixel should be approximately constant. To examine this, we calculate the standard deviation for the region and verify that the deviation is less than a fixed threshold, L_{std}. A second criterion is the ratio *I(i,j)/B(i,j)* which should be less than one and more than L_{low}. This criterium prevents dark objects to be falsely classified as shadows. Values proposed for L_{std} and L_{low} are respectively 0.05 and 0.5.

$$std_R\left(\frac{I(i,j)}{B(i,j)}\right) < L_{std} \quad and \quad L_{low} \leq \left(\frac{I(i,j)}{B(i,j)}\right) < 1 \quad (6)$$

The algorithm does perform quite well for shadow removal, but some moving object pixels are still recognised as shadow region. Despite this minor misclassification, further processing and fall detection is not affected.

C. Fall detection

After background subtraction and shadow removal which is discussed in the previous sections, we can proceed to the last and crucial stage of our camera-based fall detection system. In this section, we will present the fall detection algorithm.

We start with calculating a best fit ellipse around the foreground data. The aim of this ellipse is to model the person. To calculate the ellipse, we use the eigenvalues of the covariance matrix to compute the major and minor axis and the eigenvectors are used to calculate the angle of the ellipse. Important advantages of the ellipse tracking is that a few isolated false positive detected foreground points - which can still occur after preprocessing steps such as erosion, dilation and low pass filtering - do not influence the fall detection algorithm. A normal connected components analysis is not reliable enough to segment out the person as sometimes the body itself may be split up in more than one part. This can occur due to very small differences between background colour (grayscale value) and the skin colour or the colour of the clothing. The ellipse modelled around the person offers two parameters on which our fall detection system is based: fall angle and aspect ratio. When a fall is detected with one of these two detection methods, we continue to a fall confirmation stage, using vertical projection histograms to validate a fall. Only when the fall is detected and confirmed, an alarm is generated.

C1 Fall angle

Fall angle [2] is one of the most straightforward features to detect a fall in sideview. We can define fall angle as the angle between the person's main axis and horizontal axis (the ground). It was found in [2] that if the angle is less then 45 degrees or more than 135 degrees, we may presume that the person is falling. In our implementation, we use the angle between the ellipse's major axis and the horizontal axis. Notice that a fall angle does only give information for sideview video data. Although, with our major axis approach, experiments show that if a fall occurs in front view, the fall is still noticed as the ellipse will tilt for approximately 90 degrees. This fall decision is not based on the definition of fall angle but is a consequence of the ellipse fitting.

C2 Aspect ratio

Another frequently used technique for fall detection is the aspect ratio of a person. This is the height/width ratio of the bounding box around the object [2]. When this value is less than 1, we assume the person has fallen. In our algorithm, we calculate the aspect ratio only for foreground points within the boundaries (bounding box) of the ellipse. This to prevent spurious foreground points within the image to corrupt the result. This method provides more accurate results than taking a bounding box around the ellipse.

C3 Vertical projection histograms

When a fall is detected in the previous stage, we confirm it with vertical projection histograms (VPH). We utilise the algorithm described by Lin et al. [8] where they suggest to compare the maximum values of the two histograms. When a fall occurs, we calculate the VPH for about 1 second before, as a fall incident will take approximately 0.4 to 0.8 seconds, and the VPH for the current frame. A normalisation step is implemented because vertical projection histograms change with different distances to the camera. When the difference in maximum histogram values after normalisation is more than 25%, the fall is validated. VPH are calculated as follows:

$$H(x,y) = \begin{cases} 1 & if \quad (x,y) \quad is \quad an \quad object \quad pixel \\ 0 & otherwise \end{cases} \quad (7)$$

$$V(x) = \sum_{y} H(x,y) \quad (8)$$

In these formulas, the vertical projection histogram is presented by *V(x)*.

IV. Video acquisition

After implementing our algorithm, we need video data to test it. A search on the internet for usable video datasets with people who fall did not came up with satisfying results. Good datasets were non-existent on the internet. Only a very few movies were available and in most cases, the quality was very poor. Therefore we captured our own video dataset, containing more than 50 minutes of data with over 100 falls. Forwards and sidewards falls were simulated and filmed from different camera standpoints (eye-height and ceiling-mounted, both side view and front view). Videos were recorded using standard DV-camera's at a frame rate of 25 fps. After recording, the videos were downscaled to a resolution of 225 x 180 pixels with a frame rate of 10 fps. Typical videos contain one person, entering the room through a door, walking around and then falling. Different types of falls are simulated, such as a slow fall, stumbling and fainting. Not only falls are simulated but also laying down and sitting. All simulations are performed by four project members.

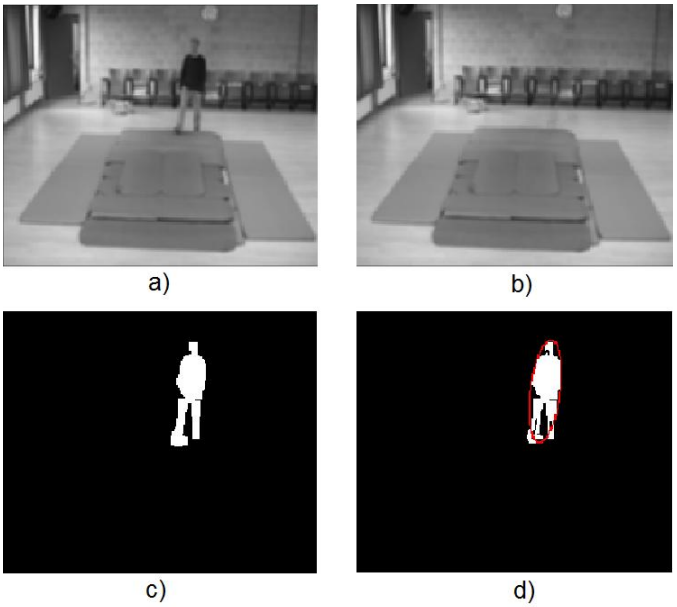

Fig. 3: a) input image, b) background estimate, c) background subtraction, d) shadow removal and ellipse fitting

V. Results

Now that we have summarised the algorithm and the recording of our test data, we can present the results of the system. The performance of the algorithm is shown in table 1. We see that false positives and negatives are present. False positives are generated due to ghosts, this are persons that stand still for a while, and thus become part of the background. When they start moving again, the background estimate updates to slowly, resulting in a false detection. A ghost detection system may bring a solution for this problem. False negatives occur with slow falling. The vertical projection histograms do not classify the event as a fall because the change in maximum value is to small within one second.

Table 1: Results

Standpoint	Correct	False pos	False neg
Side view	85%	0%	15%
Front view	78%	11%	11%

In figure 2 we clearly see the result of the shadow removal step and figure 3 shows the major processing steps. Image a) is the input frame, b) is the calculated background estimate using median filtering, c) represents the result of background subtraction, combined with morphological erosion and dilation and image d) is the resulting frame after shadow removal and ellipse fitting. This last frame is further used for fall detection. Some processing difficulties may be bypassed when colour images are used instead of grayscale video data. This simplifies shadow detection and will increase the performance of background subtraction.

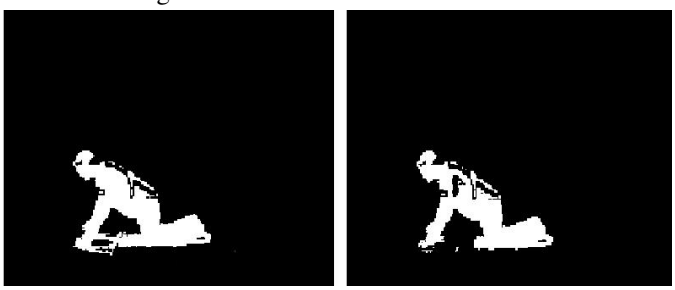

Fig. 2: Shadow detection

VI. Conclusion

In this project, we have combined different techniques proposed by different authors to become a working camera-based fall detection system based on grayscale video sequences. We do not get an accuracy of 100%, but the results are already hopeful. Further studies will be needed, especially on background subtraction and occlusion handling, which will be needed in real life applications.

Acknowledgement

The authors greatly acknowledge the financial support of this work by the Institute for the Promotion of Innovation through Science in Flanders (IWT-Vlaanderen), Belgium. Project number *IWT80150*.

References

1. U. Laessoe, H.C. Hoeck, O. Simonsen et al., 'Fall risk in an active elderly population can it be assessed?', Journal of Negative Results in BioMedicine, 6:2, Jan, 2007
2. V. Vishwakarma, C. Mandal, and S. Sural, 'Automatic detection of human fall in video', Lecture Notes in Computer Science, Pattern Recognition and Machine Intelligence, vol. 4815, pp. 616-623, 2007.
3. J. Willems, G. Debard, B.Bonroy et al., 'How to detect human fall in video? An overview', PoCA conference 2009, Antwerp Belgium, May, 2009.
4. G. Debard et al., 'FallCam: Practical Considerations in Implementing a Camera-based Fall Detection System', PoCA conference 2009, Antwerp Belgium, May, 2009.
5. S.-C. S. Cheung, and C. Kamath, 'Robust techniques for background subtraction in urban traffic video', Proceedings of the SPIE, vol. 5308, pp. 881-892, 2004.
6. J. Stauder, R. Mech and J. Ostermann, 'Detection of Moving Cast Shadows for Object Segmentation', IEEE Transactions on Multi Media, vol. 1, no. 1, pp. 65-77 March, 1999.
7. J.C.S. Jacques Jr, C.R. Jung and S.R. Musse, 'Background Subtraction and Shadow Detection in Grayscale Video Sequences', Computer Graphics and Image Processing, SIBGRAPI 2005, pp. 189 - 196, Oct, 2005.
8. C.W. Lin and Z.H. Ling, 'Automatic Fall Incident Detection in Compressed Video for Intelligent Homecare', Computer Communications and Networks, ICCCN 2007, pp. 1172 - 1177, Aug, 2007.