Fall Detection Project

Adrian Sochaniwsky
McMaster University

Abstract

This paper presents a fall detection algorithm that uses motion history images and an SVM classifier. This approach features contour-based features and PCA reduction. The results were compared to a deep learning-based approach. The SVM achieved a sensitively of 81% and a specificity of 66%, the CNN achieved 94% and 78%.

1. Introduction

Falls are a public health concern because they are a major cause of deaths and injury [1]. The consequences of falling for the elderly are particularly great, due to longer recovery time, loss of independence, and higher morbidity rates. These consequences combined with a growing population of older adults are increasing health care costs and putting a strain on long-term care facilities. However, it has been proven that falls that are reported earlier increase the probability of recovery. Robust fall detection systems can restore confidence in the older population, providing them with more independence and a better quality of life.

There are many existing methods for fall detection, they are categorized into three groups: wearable, ambient and vision systems. Wearable systems include sensors like accelerometers and gyroscopes which use inertial measurements to make a fall determination. The user is required to consistently wear them, many people may be uncomfortable and resist having them, or may simply forget. Ambient systems analyze signals in the environment such as radar, infrared, and pressure to determine if a fall has occurred. Ambient algorithms can be complicated and require multiple sensors. The final category is vision-based methods, which invlude the use of single RGB, 3D, or stereo cameras. Methods such as image processing and deep learning can be used to identify falls. This paper will discuss a vision-based fall detection system using a single RGB camera system.

A fall detection system requires multiple features to ensure it is acceptable. First, it must be able to run at real-time speeds to be a viable online detector and not miss a fall event. Second, the system must be robust to lighting changes, as the average home does not have a static lighting scene. Also, periods of inactivity and other daily living tasks must not trigger the system. Because cameras capture so much information, privacy is a concern when systems are designed to operate in a users home. Thus it is beneficial to process video locally on an edge device and only upload the fall determination to a monitoring system. In order to perform image processing locally, algorithms benefit from being lightweight and not requiring large computational resources.

After discussing relevant works in Sec. 2, the fall detection method is detailed in Sec. 3. Experimental results are discussed in Sec. 4.

2. Related Works

Single camera fall detection has been thoroughly studied due to its low cost and simple setup. Many of the methods can be grouped into three categories: shape related feature analysis, motion analysis, and inactivity analysis. Shape related features include measuring the width to height ratio of extracted silhouettes. Mirmahboub et al. [5] used background estimation to find a human silhouette, then trained a *support vector machine* (SVM) to classify frames based on extracted features. Nasution et al. [7] extracted projection histograms of a segmented person and classified with a k-Nearest Neighbour algorithm.

Motion analysis uses spatio-temporal based features to determine fall events. Nunez-Marcos [9] used optical flow images fed into a *convolutional neural network* (CNN) with strong results. Haraldsson [4] created a *Motion History Image* (MHI) by subtracting the difference between consecutive frames and summing a window a frames with a decay factor. Then applied the MHIs to a CNN to classify falls from regular movements. Rougier et al. [10] employed a calibrated camera to track the 3D position and velocity of the human head, the characteristics of the velocity vectors are used for fall detection.

An example of inactivity detection presented by Nater et al. [8] used multiple tracking algorithms tuned for specific normal activities. When an activity is abnormal it is categorized as a fall. Nait-Charif et al. [6] used ceiling-mounted cameras and developed learned models in conjunction with trackers to classify falls based on spatial context.

3. Proposed Method

Two fall detection methods will be compared in this paper, both will employ motion analysis through an MHI. The first will be a traditional computer vision method that uses image features and an SVM classifier. The other utilizes a deep learning-based approach, through the use of a CNN.

The input to the algorithms is a 25 frames per second video stream and the frames are resized to 128x128 pixels. The procedure for generating the MHI is in Fig. 1. First, the absolute difference between frames is computed on an interval of every two frames, this creates the temporal difference. Then a binary threshold is computed to determine if a significant amount of movement has occurred between the frames. The resulting frame is added to a sliding window of the previous 40 MHI result multiplied by a decay factor.

The first method is displayed in Fig. 3, incoming frames will be used to compute an (MHI), then contour features are extracted. The process is shown in Fig. 2, it begins with a binary threshold on the MHIs. The binary images have the closing morphological operator applied, followed by dilation to create a single blob to extract the contour from. Once the contour is detected, the following features are extracted: area, non-zero pixel count, arc length, bounding, box width, bounding box ratio, and the moment values. In the final step before classification with the SVM, the features pass through a *principle component analysis* (PCA) reduction algorithm. PCA finds maximizes the variance of the data by creating a linear combination of the data, while reducing the dimension of the feature vector. Finally, the reduced feature vector is classified using an SVM. A *Radial Basis Function* (RBF) kernel was selected for its ability to map the data into an infinite-dimensional space to create separation. This is important when the data does not have a clean linear separation as seen in Fig. 9.

The second approach is shown in Fig. 4. The MHI is passed directly to a CNN for classification. In this project transfer learning was used on MobileNetV2 [11], which was pretrained with ImageNet [3]. Transfer learning is a technique that uses a network pre-trained on large datasets, and allows the use of a limited dataset to train only the layers closer to the output. In this case, we are leveraging the upper layers which detect basic features and using our relatively small dataset to tune the final layer to make a fall or not fall determination based on the MHI input. MobileNetV2 was used due to its relatively small architecture, which allows the algorithm to run in real-time, its architecture can be seen in Fig. 5.

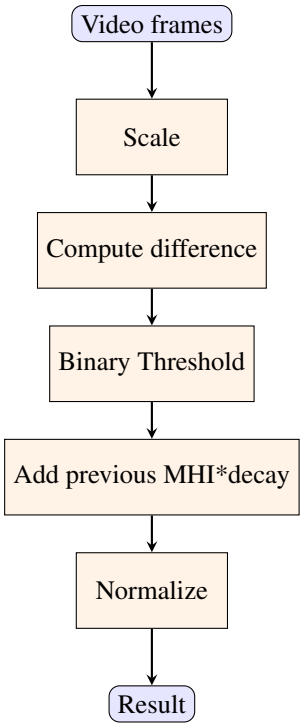


Figure 1. Block diagram of the MHI generator.

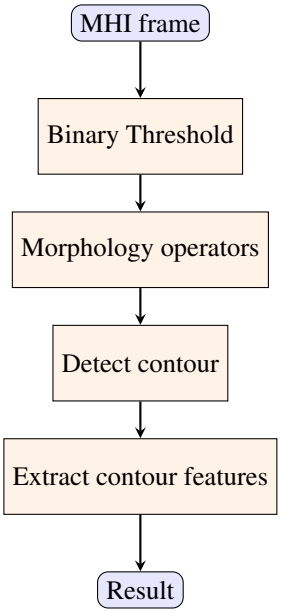


Figure 2. Block diagram of the feature extractor.

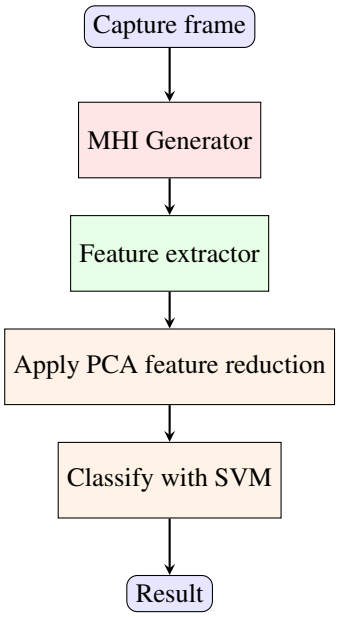


Figure 3. SVM Based classifier

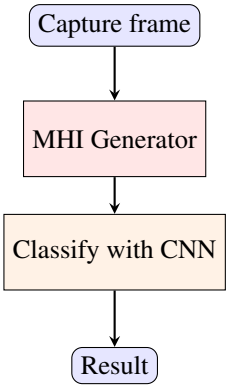


Figure 4. CNN based classifier.

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Figure 5. The MobileNetV2 [11] architecture.

(a)

(b)

Figure 6. Example frames from dataset

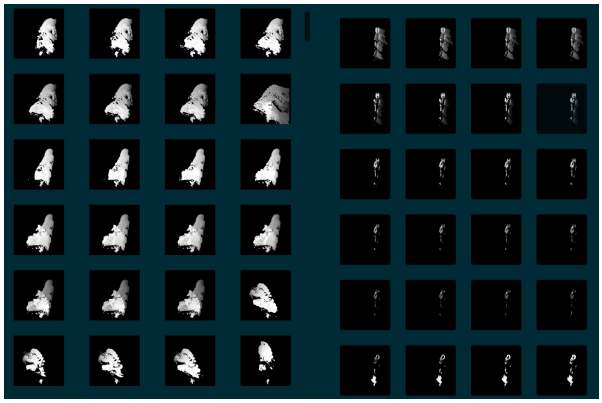
4. Experiment

4.1. Dataset

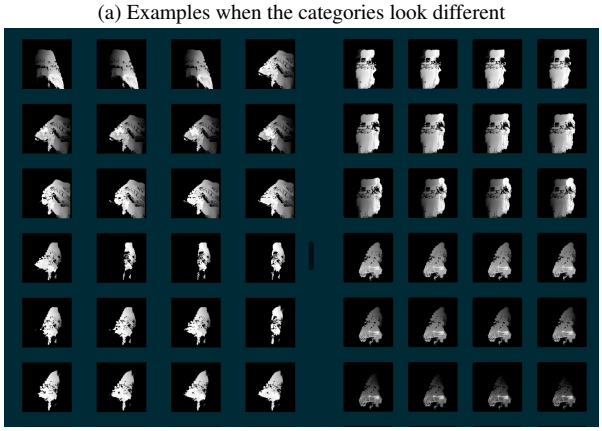
Charfi et al. introduced a public dataset [2] with multiple locations and subjects performing normal activities and falls. Fig. 6 demonstrates sample frames from the dataset. Using the provided annotations for which frames include a fall, videos were processed to create MHI and separated into fall and not fall categories. The MHIs were then processed to extract contour features and saved into csv files.

4.2. Feature Extraction

The first processing stage is the creation of the MHI using the steps outlined in Fig. 1. Fig. 7(a) demonstrates computed MHIs for the fall category on the left side and not fall on the right, where the subject is stationary. In (b) the not fall category on the right has the subject moving, but not falling.



(a) Examples when the categories look different



(b) Examples when the two categories look similar

Figure 7. Example frames from dataset, left in each image is 'fall', right is 'not fall'

The next stage is extracting contour features as per Fig. 2. In Fig. 8 the right image is the binary threshold result, this value was chosen to eliminate the grey pixels seen on the right side of Fig. 6(b). The middle image shows the morphology result, its goal is to create a single blob for for better contour detection. Finally, the left image is the resulting bounding box of the detected contour drawn on the original MHI frame.

The extracted features undergo a principal component analysis (PCA) which is used to reduce the feature set. The PCA reduction helps decrease training time and memory use. Fig. 9 shows the resulting data points from this reduction. This is the feature vector used for training the SVM classifier.

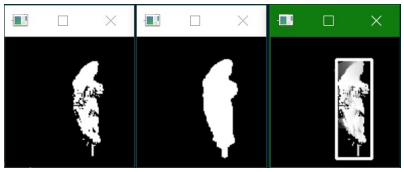


Figure 8. Processing the MHI image

4.3. Training

The fall detection dataset has approximately ten times more 'not fall' frames than 'fall' frames. This imbalance requires the classifiers to have class weights assigned during training to ensure that the fall data points are learned. Both the SVM and CNN performance suffered without the adjusted weights. The weights were calculated using Eq. (1).

$$[w_{fall}, w_{not}] = [\frac{N_{fall} + N_{not}}{N_{fall}}, \frac{N_{fall} + N_{not}}{N_{not}}] \quad (1)$$

The RBF kernel has two hyperparameters, C and γ. C determines the amount of regularization, a smaller C value results in a smoother decision surface and this reduces overfitting. γ defines the influence each sample has on the training, it was calculated using Eq. (2).

For the CNN, 20 epochs were selected. The optimizing algorithm is stochastic gradient descent (SGD), with the learning rate set to 0.001, and the momentum value set to 0.9. The momentum parameter helps SGD converge quicker by smoothing the estimated gradient with a weighted average.

$$\gamma = \frac{1}{N_{features} * \sigma^2_{features}} \quad (2)$$

SVM		CNN	
Kernel	RBF	Learning Rate	0.001
C	0.2	Momentum	0.9
γ	$\frac{1}{N_{features} * \sigma^2_{features}}$	Epochs	20

Table 1. Classifier parameters

The SVM training completed in approximately 4 seconds. The CNN required approximately 2 hours and 20 minutes for training.

4.4. Results

In order to effectively compare both approaches to fall detection we will compare the sensitivity, specificity because they are not biased by the class imbalance. Sensitivity is the classifiers ability to predict a no-fall event. Specificity evaluates the performance of detecting falls. These metrics are define below:

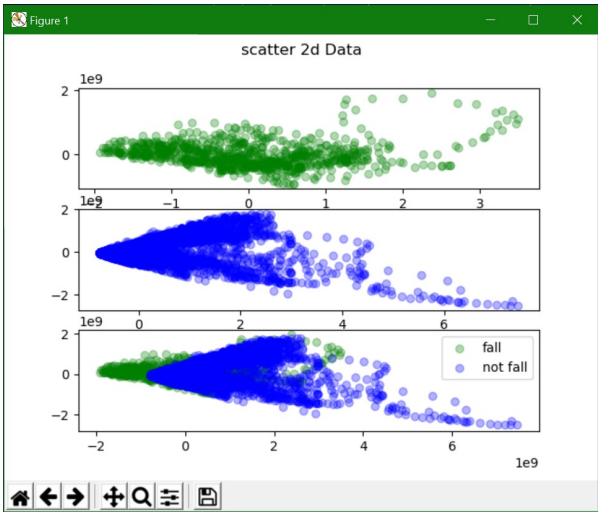


Figure 9. Plot of the Fall vs. Not Fall data

$$Sensitivity = \frac{TP}{TP + FN}$$
$$Specificity = \frac{TN}{TN + FP} \quad (3)$$

4.4.1 SVM Results

Tab. 2 shows the results of the SVM classifier on the training and test data sets. There is a slight drop in performance from the training set to the test set, however it is small enough to conclude that the SVM generalizes fairly well with the test data. Specificity is lower than the sensitivity, this is expected because of the imbalanced training data. The full normalized confusion matrix is in Tab. 3.

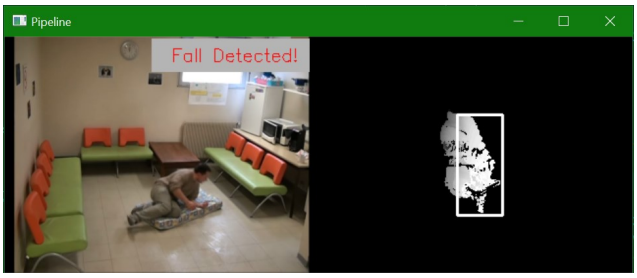
SVM	Sensitivity	Specificity
Training	83%	69%
Testing	81%	66%

Table 2. SVM Results

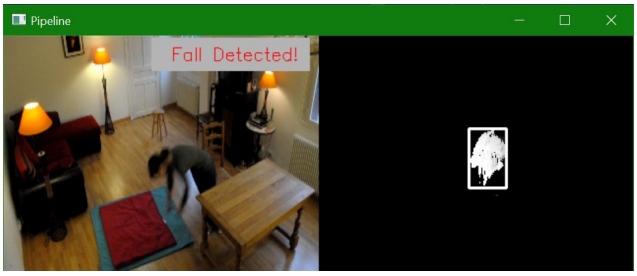
	Not Fall	Fall
Not Fall	81%	34%
Fall	19%	66%

Table 3. SVM Confusion Matrix

Examples of correct detections from the SVM can be seen in Fig. 10. The (a) and (b) are detected falls during the actors motion of falling. The third image demonstrates a correct classification of not-fall during a normal walking sequence. An incorrect detection from the SVM can be seen in Fig. 12. Here the actor is standing up after a fall, the SVM classifier determines this to be a fall.



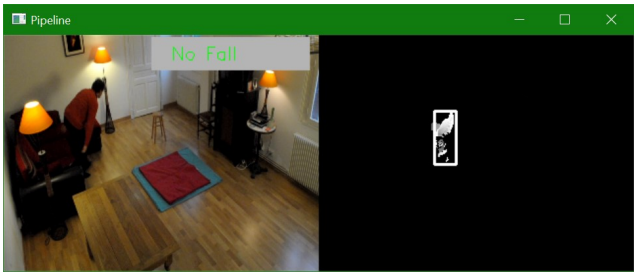
(a) Male subject falling down.



(b) Female subject falling from the desk.

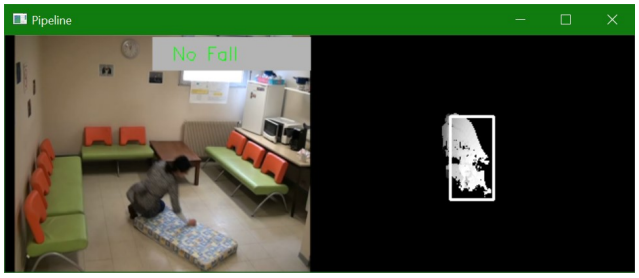
Figure 10. Correct 'fall' classification examples.

(a) Male subject walking.

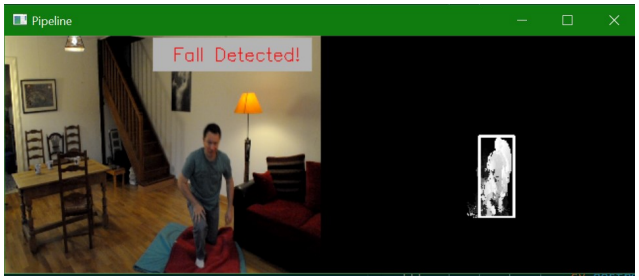


(b) Female subject standing up from a chair.

Figure 11. Correct 'not fall' classification examples.



(a) Female subject falling sideways.



(b) Male subject standing up.

Figure 12. Incorrect classification examples.

4.4.2 CNN Results

Tab. 4 shows results for the CNN, where there is a large decline in the ability to detect falls. This could be due to overfitting on the limited dataset set. The full normalized confusion matrix is in Tab. 5.

CNN	Sensitivity	Specificity
Training	95%	95%
Testing	94%	78%

Table 4. CNN Results

	Not Fall	Fall
Not Fall	94%	22%
Fall	6%	78%

Table 5. CNN Confusion Matrix

5. Conclusion

We proposed a method of fall detection using motion history images and an SVM classifier. The CNN classifier outperformed the SVM by approximately 12% in each category. This result is expected because the CNN can create more complex features that can better identify falls. However, there is a trade-off between accuracy and speed and complexity, the SVM executes faster and trains faster than the CNN approach.

Many future directions remained unexplored. This includes using different motion analyses to create spatio-temporal images to generate better features. Additionally, other features can be extracted from the spatio-temporal images for better classification including a histogram of gradients. Furthermore, camera calibration can be explored to take advantage of depth estimates to better estimate distances and velocities to aid in classification. Lastly, a larger dataset would enable better generalization and classification performance.

References

[1] Anne Ambrose, Geet Paul, and Jeffrey Hausdorff. Risk factors for falls among older adults: A review of the literature. *Maturitas*, 75, 03 2013. 1

[2] Imen Charfi, Johel Miteran, Julien Dubois, Mohamed Atri, and Rached Tourki. Optimized spatio-temporal descriptors for real-time fall detection: Comparison of support vector machine and adaboost-based classification. *Journal of Electronic Imaging*, 22:041106–041106, 10 2013. 3

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 2

[4] Truls Haraldsson. *Real-time Vision-based Fall Detection: with Motion History Images and Convolutional Neural Networks*. PhD thesis, 2018. 1

[5] Behzad Mirmahboub, Shadrokh Samavi, Nader Karimi, and Shahram Shirani. Automatic monocular system for human fall detection based on variations in silhouette area. *IEEE transactions on bio-medical engineering*, 60, 11 2012. 1

[6] H. Nait-Charif and S.J. McKenna. Activity summarisation and fall detection in a supportive home environment. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 4, pages 323–326 Vol.4, 2004. 1

[7] Arie Hans Nasution and Sabu Emmanuel. Intelligent video surveillance for monitoring elderly in home environments. In *2007 IEEE 9th Workshop on Multimedia Signal Processing*, pages 203–206. IEEE, 2007. 1

[8] Fabian Nater, Helmut Grabner, Tobias Jaeggli, and Luc Van Gool. Tracker trees for unusual event detection. pages 1113 – 1120, 11 2009. 1

[9] Adrián Núñez-Marcos, Gorka Azkune, and Ignacio Arganda-Carreras. Vision-based fall detection with convolutional neural networks. *Wireless Communications and Mobile Computing*, 2017:1–16, 12 2017. 1

[10] Caroline Rougier, Jean Meunier, Alain St-Arnaud, and Jacqueline Rousseau. Monocular 3d head tracking to detect falls of elderly people. *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, 1:6384–7, 02 2006. 1

[11] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 2, 3