ON-LINE FALL DETECTION VIA A BOOSTED CASCADE OF HYBRID FEATURES

Haifeng Liu[1], *Dong Liu*[1],*Xiaoyan Sun*[2], *Feng Wu*[1], *Wenjun Zeng*[2]

[1] University of Science and Technology of China, Hefei, China,
[2] Microsoft Research Asia, Beijing, China
lemt@mail.ustc.edu.cn, {dongeliu,fengwu}@ustc.edu.cn,{xysun,wezeng}@microsoft.com

ABSTRACT

In this paper, we present a cascaded learning approach for online fall detection in streaming videos. In the cascaded approach, we propose using hybrid features ranging from Haar-like features to motion boundary histogram (MBH) rather than involving only one single type of features as in traditional cascaded methods. The simple features are employed at earlier stages to rapidly filter out the majority of none fall video clips which are relatively easy to detect, while complex features are used incrementally at later stages to handle samples that are difficult to distinguish. Similarly, the classifier at each stage is trained from simple to complex based on the gradient boosted tree. We further conduct a fall dataset with 400+ hours indoor videos to provide enough samples for the training. Experimental results show that our proposed fall detector achieves significantly higher performance than state-of-the-art approaches in term of both accuracy and speed.

Index Terms— Fall detection, cascade boosting, rare event, cascaded detector

1. INTRODUCTION

Fall detection has drawn increasing interests and public awareness. Studies have shown that fall is one of the most dangerous actions of senior citizens at home [1]. Fall can also cause serious damage to babies, patients, disabilities, and normal person as well. It is thus becoming one of the badly desired features of automatic e-health care as well as video surveillance systems.

To detect fall automatically, several kinds of methods have been presented in the literature. They can be roughly categorized into three classes: wearable-device based, ambience-device based and vision based methods [2]. Wearable-device approaches rely on garments with embedded sensors like accelerators and gyroscopes [2, 3] to detect the motion and location of the body of the subject. Ambience-device based attempt to fuse audio and visual data and event sensing through vibrational data [2, 4]. Vision based methods automatically detect the fall using a monocular camera or multiple cameras by analyzing the motion information. In this paper, we focus on the vision-based methods, especially, methods with the only monocular camera as monocular camera is cheaper, more popular, and easier to be installed. Moreover, a practical fall detection system often requires timely responses, especially in streaming videos. A constantly streamed video input poses a big pressure on the detection efficiency as delays in any clips will be accumulated to slow down the following responses. Therefore, one key problem should be solved is the good trade-off between detection accuracy and speed in streaming videos.

Detecting fall using a monocular camera is challenging. Researchers propose fall detection methods by analyzing the features of human shape via bounding boxes [5], estimated ellipses [6]. Features of human motion have also been studied via motion history image [6], and distance of Silhouette edge point [6] between consecutive frames. The simplicity of the shape and motion features makes these fall detectors run extremely fast but limits their detection capacity. Advanced algorithms, including using features extracted by the deep learning model PCANet [7] and analyzing the dynamic shape and motion of human bodies on Riemannian manifolds [8], help to boost the performance of detection, but greatly increase the complexity and thus are hard to be used in online fall detection.

On the other hand, we notice that the cascaded boosting framework first presented for face detection [9] achieved a good trade-off between efficiency and accuracy. Later, this elegant work was extended to videos for action detection [10] in which the 2D box features in images were extended to 3D spatio-temporal volumetric features. Future improvements have been presented by introducing new features and alternative structures [11]. However, these approaches always employ one kind of features, either simple or complex. Simple features, e.g. Harr features, helps to further accelerate the detection speed, while complex features, e.g. dense trajectory[12] , provides enhanced accuracy in the cost of real-time efficiency.

Motivated by the high efficient cascaded boosting framework, we propose an online fall detection method by a boosting cascade of hybrid features to address the aforementioned limitations. Our hybrid features contain very different complexities ranging from Haar-like ones to motion boundary histogram (MBH). In our framework, simple features are used in the earlier stages to quickly reject most of irrelevant videos,

978-1-5386-0560-8/17/$31.00 ©2017 IEEE

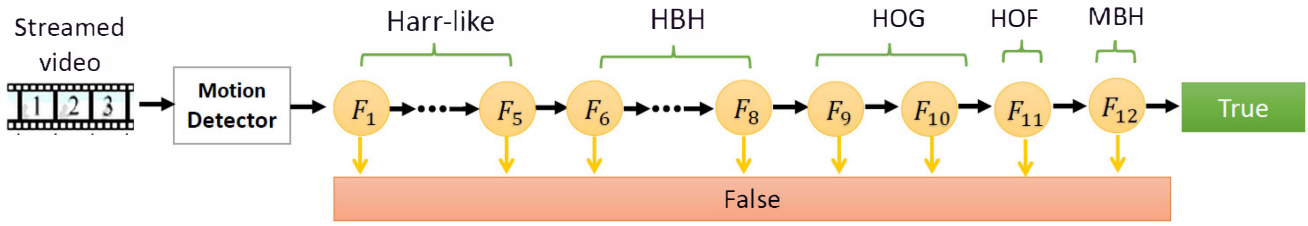

Fig. 1. Framework of our cascaded fall detection. The cascade contains 12 stages which are denoted by $F_i, i = 1, 2, ..., 12$. Each video clip generated by temporal sliding window is fed to the cascaded stages. A false prediction causes the detector to return false and stop further processing this clip while a true prediction passes the input clip to the next stage. Rather than a single kind of features, we propose using five kinds of features jointly in our framework. They are Haar-like features, Haar boundary histogram (HBH), Histogram of gradient (HOG), Histogram of optical flow (HOF) and Motion boundary Histogram (MBH). In this cascade detector, the later the stage is, the more complex classifier as well as features are used.

while complex features are involved in the later stages incrementally to help distinguish from increasingly ambiguous negative video clips. Moreover, the features on each stage are selected and learned by Gradient Boosted Trees (GBT) to form a binary classifier. In this way, our proposed fall detector is able to rapidly ignore the massive irrelevant videos by simple features meanwhile accurately identify fall actions via advanced ones under the complexity constraint.

Another challenging problem in fall detection is the limited training data. Large scale training samples are necessary for fall detection, especially in the cascaded boosting framework. We need enough positive and negative video samples so as to train stages of a cascade. Therefore, we conduct a large video dataset containing 400 hours' daily living activities captured by four cameras. Incorporating existing public fall datasets, our cascade involves millions of video clips containing a variety of continuous actions and activities that makes the learned classifiers more robust and realistic. Experimental results show that our fall detector via a boosted cascade of hybrid features achieves a high detection rate of 0.95 at a low false-alarm rate of 10^{-3} in real time (69 fps) in streaming videos, which largely pushes the fall detection into practice. We also compare to the state-of-the-art fall detection methods and obtain large improvements than these methods in both accuracy and speed.

2. FRAMEWORK

In this section, we first give an overview of our framework, then present our designed features followed by the training process.

2.1. Overview

Fig. 1 illustrates the framework of our presented fall detection method in which 12 stages of classifiers are involved in the cascade. Given a streaming video, fall is detected in each video clip generated by sliding the window of the streamed video. Before the fall detection, a motion detector is introduced to filter out the static clips. Then the 12 classifiers denoted as $F_i, i = 1, 2, ..., 12$, is used in a cascaded fashion for fall detection. A false decision halts further processing of this clip. A true decision allows the clip to go through the next classifier in the cascade. If all the classifiers vote true then the video clip is classified as a fall clip. In this cascade detector, the later the stage is, the more complex classifier as well as features are used. To achieve a good trade-off between detection accuracy and speed, our target is to reject as many false clips as possible in the earlier stages while leaving as much as possible true clips to the latest complex stage to detect.

In our fall detection method, we use five kinds of features, Harr-like, Haar Boundary Histograms (HBH), HOG , HOF [13], and MBH [14], sequentially. These five kinds of features will be introduced in detailed later. In fact, many other complicated features, like deep learning features or improved dense trajectory [12], can be employed in the framework in the pursuit of high accuracy. However, for fall detection in streaming videos, high detection speed is one of the key issues to ensure real time surveillance. Constantly streamed videos place a real-time constraint on the processing speed of all the stages.

We also introduced a motion detector [15] in our framework as shown in Fig. 1. It helps to speed up the detection process by rejecting static video clips in the streamed video. We further utilize the output of motion detector to select the motion area in each video clip. Thus our fall detection only needs to process motion areas rather than whole video clips all the time.

In the following, features as well as training process used in the presented framework will be introduced in details.

2.2. Feature Design

In our proposed fall detection method, we involve five kinds of features to exploit motion, appearance, and joint motion-appearance patterns of video sequences. Specifically, they

are: Haar-like, Haar-Boundary Histogram, HOG, HOF and MBH [14], from simple to complex corresponding to the stages from weak to strong in our cascaded boosting framework.

Harr-like feature Theoretically, the number of Harr-like features of a video clip can be infinite. In our scheme, we focus on the motion information between frames for super fast fall detection. The Harr-like features are produced only between adjacent frames for each $n \times n$ patch sampled by the boxing sampling in multiple scales [13]. In fact, the calculated Harr-like features encode both spatial and temporal information of one video clip.

To further enhance the robustness of the extracted features, we introduce a temporal pooling function to generate the final harr-features. The pooling function is defined as average pooling. It is performed in every t frames at the interval of k. Table 1 evaluates the performance of the pooling function. By using the pooling function, we enhance the performance of our Harr-like features by 2.5% percent in AUC score.

Haar-like features provide coarse temporal and spatial motion information which are able to identify the most irrelevant clips rapidly. On the later stages, we extract more complicate local features from video clips and then generate the final representation by bag of visual words [13].

Table 1. Evaluation of pooling of Harr-like features. Testing B and Testing A show the results by using dataset A (our generated dataset) and dataset B (three public dataset together) as training dataset, respectively. The performance is evaluated in the Area Under Curve (AUC) score.

Setting for Haar feature	Testing B	Testing A	Average
No temporal pooling	90.78	80.57	85.68
Temporal pooling 3	91.55	82.48	87.02
Temporal pooling 5	90.74	85.70	88.22

Given the video clips, we follow the dense sampling strategy [13] which is verified to be effective than other sampling methods to extract video blocks at regular positions and scales in space and time. In our experiments, the minimum size of a 3D patch is 30×25 pixels with 10 frames. We follow the same sampling strategy for the following local features. More details for the local features are given below.

Haar Boundary Histograms The Harr-like features, though efficient, provide only very limited motion information between adjacent frames. A relatively fine-grained motion descriptor is needed. Since we target for the surveillance videos, we find the temporal haar of consecutive frames contains rich information like the salient motion region and motion intensity.

Given a sampled video block, we calculate the haar map of each consecutive regions and divide the haar map into sub-units (2×2) and extract the histogram of gradient of from each sub-unit, we set the bin size of histogram as 8, finally we concatenate each histogram along the temporal frames into a long vector as the local feature. We call this family of features *Haar Boundary Histograms* (HBH). This feature is similar to MBH [14], they both characterize the motion boundary but the MBH feature is based on advanced optical flow, HBH is based on simple Haar, which is appropriate in the earlier stages. To support for fast feature calculation, we introduce integral images as [9] did.

HOG We introduce the HOG descriptor to measure the appearance of human body to be a complement to motion descriptors. For each region of the sampled video block, it is first divided into sub-units (2×2) from which the histogram of gradient orientations over the pixels are computed. In the implementation of the HOG features, we find all the background will be taken into the HOG features calculation and it may dominate the HOG feature which makes the HOG features over-fit to the scenes. With the guidance of motion masks, most of the background can be removed efficiently, which will significantly boost the performance. HoG feature can also be accelerated by integral images.

HOF We propose to adopt HOF[13] to distinguish between the fine-grained motions but with relatively heavy feature computation. HOF is calculated from optical flow map which depicts the accurate motions. Given a video block, we first calculate the optical flow of each consecutive frames then calculate the histogram of its gradient in the same setting of HOG.

MBH MBH [14] is also calculated from optical flow but MBH targets on the gradient of optical flow (horizontal (MBHx) and vertical (MBHy) descriptor) to focus on the motion boundary. For a sampled video block, we extract the feature vector of MBHx and MBHy separately as HOF does and then concatenate them into a final local feature.

The current speed of optical-flow on the videos with resolution (320×240) is around 26 fps, with previous stages, it cannot support real-time processing for streaming videos. Guided with motion masks, the speed of optical flow can be accelerated by only calculating the motion regions, but the acceleration varies among videos. To ensure the real-time processing, we down-sample the video to lower resolution for optical flow calculation, then up-sample the optical-flow maps to the original sizes. In our setting, the down-sample ratio is 0.5, which can accelerate the speed of optical-flow calculation by more than four times.

Visual Words Assignment Several visual words assignment strategies can be chosen. In our scheme, we choose the K-means as our visual words assignment for the following reasons: 1) it's simple, effective and widely used. 2) K-means with gradient boosted tree can be extremely fast in the testing stage because only selected visual words will be considered which significantly reduces the similarity matching time. We set the number of visual words to 2000 which has shown to empirically give good results for a wide range of datasets.

Commonly, before words assignment, the size of feature vector will be reduced to half by the PCA to reduce the noise and make the feature robust. In our scheme we replace the PCA by the average temporal pooling with a size of 2. We compare these two settings in Table 2 and find that generally the performance of temporal pooling doesn't degrade significantly and for the HBH feature the performance is even higher than that of PCA, but the complexity drops dramatically from $O(N^3)$ to $O(N)$. Based on these analysis, we choose the temporal pooling for feature length reduction.

Table 2. Comparison of PCA and temporal pooling for local features before visual words assignment.The performance is evaluated by the AUC score which is an average from dataset A and B. The complexity cost means that for the length of N feature vector, the cost of each setting to reduce half of the dimension.

Setting	HBH	HOG	HOF	MBH	Cost
PCA	89.94	91.02	91.34	92.02	$O(N^3)$
Pooling	90.58	90.79	91.25	91.75	$O(N)$

2.3. Training process

2.3.1. Training Cascade

We follow the traditional training pipeline[9] to train our cascade. The training process uses Gradient Boosted Trees(GBT) to select a subset of features and construct the classifier. Each classifier in the cascade is trained to achieve very high detection rates, and modest false positive rates. The positive samples will stay unchanged in each stage while negative samples in each stages are selected from unrejected samples by previous stages.

Given the structure of the cascade, each stage acts to reduce both the false positives and the detection rate of the previous stage. The key is to reduce the false positive rate more rapidly than the detection rate.

3. EXPERIMENTS

In this section, we first present the dataset we used in our experiments, and then describe the details of our implementation of our framework. Finally, we give the experimental results and analysis.

3.1. Falling Down Dataset

To the best of our knowledge, there is no public fall dataset containing large amount of streaming video data in surveillance scenarios that can be used for our cascade training. we build a new dataset that contains more than 400 hours' streaming videos. We conduct and capture the activities of daily life with four cameras with different views as illustrated in Fig. 2.

There are 10 actors contributing to the dataset. The actors were asked to do the following actions: ReadBook, Sleep, SitDown, Lie, DrinkWater, ThrowAway, Walk, SitUp, MakePhoneCall, and the most important one Fall. Each one contribute the whole set of actions. In our setting, we did not restrict the number of people in the video, more than 400 hours of data were recorded with the resolution of 1280 $\times$ 720 at 30 FPS, in the form of 8-bit 3-channel color images. To simulate the falling down action in different scenarios, we consider forward fall, backward fall, loss of balance fall and fall in a faint.

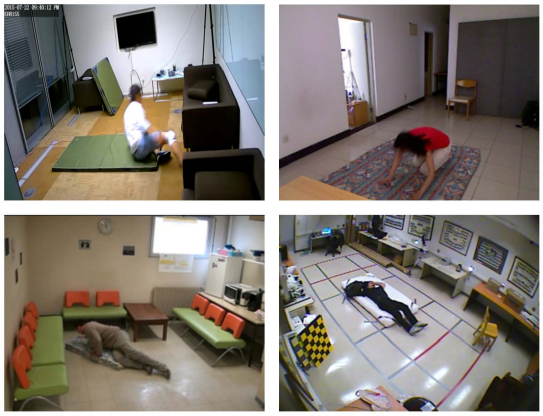

Fig. 2. Collected datasets. Top left: Our 400 hours' dataset. From top right to bottom right are ACT4 dataset, Fall dataset and Fall detection dataset respectively.These three datasets either are strictly trimmed or its total length is less than three hours, which is far less than our dataset.

To come up with a more practical solution, we include three public fall datasets to build a more diverse fall dataset, they are: ACT4 dataset [16], Fall dataset [17] and Fall Detection Dataset [1] as illustrated in Fig. 2. The four datasets generate 12528 positive samples among which our dataset contributes 7975 positive samples.

3.2. Implementation Details

For training, we split the positive video clips into four folds according to different people. The negative samples are randomly cropped from the streaming videos. We set the video clip length fixed at 45 frames which is enough to contain almost all of falling down actions.

We automatically train the cascade and choose the features in each stage. We rank the complexity of these features from simple to complex. In each stage if the false-alarm is larger than 0.7, the feature type will be abandoned and replaced with the next feature type for this stage. For In each stage the training data contain 9000 positive samples and at least 9000 negative samples. In the training process, each stage will stop when the false-alarm rate reaches 0.3 while the hit-rate is maintained at above 0.9960 or the number of learned trees exceeds the maximum.

[1]Fall Detection Dataset:http://le2i.cnrs.fr/Fall-detection-Dataset?lang=fr

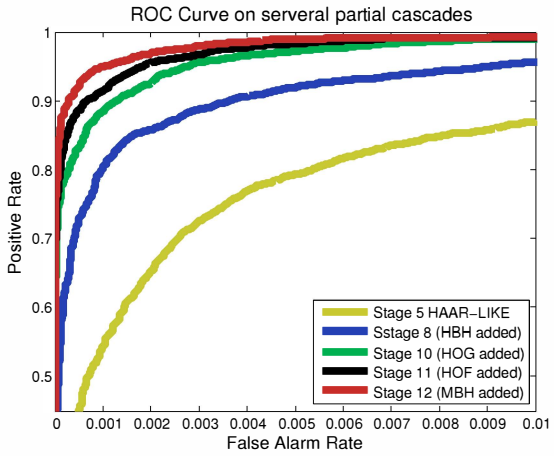

Fig. 3. ROC curve on several partial cascades.

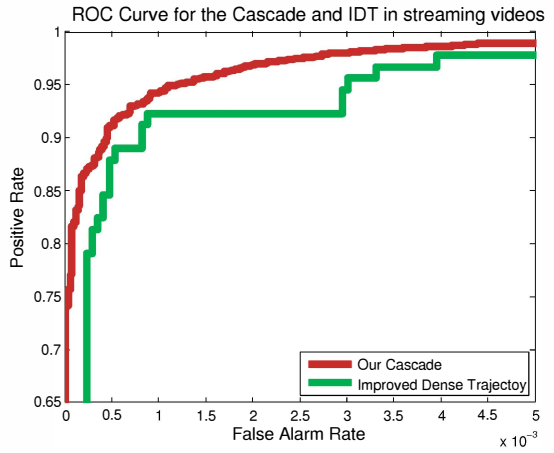

Fig. 4. Testing ROC curve on streaming video.

3.3. Cascade analysis

We have built a cascade with several stages. To verify the effectiveness of features in the cascade, we draw the partial cascade in testing dataset using ROC metric in Fig. 3. We can observe that the performance is increasing with the number of the stages increasing, which illustrates the effectiveness of our proposed features.

We further evaluate our scheme on the streaming videos. In this experiment, we captured a more than one hour's streaming video where the actor conducted six fall actions and other daily life actions as mentioned in Sec.3.1. With frame by frame sliding window, the streaming video produces 117531 video clips with 100 positive samples, the result is illustrated in Fig. 4. To verify our system performance, we also choose the leading approach IDT [12] as our reference for the following considerations: 1) IDT code can be accessed while there is no other state-of-art fall detection code released. 2) IDT representation is powerful and complicated containing HOG,HOF,MBH features. From Fig. 4 Our cascade has better performance than IDT due to our advanced framework and efficient feature design.

Table 3. Comparisons of existing methods on two datasets. C or D means training and testing are conducted in the dataset C or D with half by half splitting.

Method	TPR(%)	TNR(%)	Train/Test
Auvinet 2011 [18]	80.6	**100.00**	C
Rougier 2011 [6]	95.4	95.8	C
Yun 2016 [8]	**100.00**	**100.00**	C
Proposed	**100.00**	**100.00**	C
Cheng 2012 [16]	89.06	92.71	D
Yun 2016 [8]	98.13	94.48	D
Proposed	**100.00**	**98.28**	D

3.4. Comparisons with existing methods

We compare our method with four existing methods on public fall datasets. Following the evaluation strategy outlined in [8], we choose ACT4 dataset [16] (Dataset C) and Fall dataset [17] (Dataset D)for comparisons. The performance is evaluated by the true positive rate (TPR), true negative rate (TNR) and False Negative Rate(FNR). We have two settings for evaluation, the first setting is evaluation within one dataset which splits the training and testing half by half as shown in Table 3, the second setting is training with one dataset and testing with the other dataset as shown in Table 4. Under the same evaluation metric, our approach achieves significant improvement over the state-of-the-art methods, especially when testing on videos with different scenes under the second setting. These promising results further demonstrate the effectiveness of our cascaded framework.

Table 4. Comparisons with state-of-the-art approach [8] under the second setting that the model is trained on one dataset but tested on the other dataset.For fair comparison, we keep false negative rate (FNR) the same.

Method	TPR(%)	FNR(%)	Train	Test
Yun 2016 [8]	49.87	0.26	C	D
Proposed	**100.00**	0.26	C	D
Yun 2016 [8]	59.90	0	D	C
Proposed	**95.21**	0	D	C

3.5. Processing speed comparison and analysis

To verify our cascade efficiency, we evaluate our system speed on three videos from the public dataset [17] and compare the state-of-the-art method [8]. Our results are listed in Table 5, since the source code of Yun 2016 [8] is not publicly available, we provide the reported feature extracting speed 0.076 fps as reference, this indicator is tested on a PC with

an Intel i7-2600 3.40GHZ processor and 16GB RAM while our detection speed is tested on a PC with dual 3.74 GHz Core Intel i-7 CPU and 16GB RAM, which is quite similar. From the results we can find that our system obtains a much faster speed than Yun et al [8] which verifies that our cascade is highly efficient and can well support streaming scenario.

Besides, we also evaluate the speed of each feature extracting and list in Table 6. In the implementation of our cascade, we fully utilize the characteristic of streaming scenario and reuse the history feature. In the cascade, the extracted raw features/maps in the earlier stages can also be used for the next stages. For each feature design, we also highly optimize for the good trade off between accuracy and efficiency as showed in Sec..

Table 5. Processing speed comparison on three videos from public dataset [17]

Method	Video1	Video2	Video3	Average
Proposed (fps)	72.5	64.7	70.7	69.3

Table 6. Time cost of feature extracting. (Metric:fps, MoD denotes motion detector.)

Method	MoD	Haar	HBH	HOG	HOF	MBH
Speed	330	2250	450	450	183	244.4

4. CONCLUSION

In this paper, we introduce the first cascade boosting framework for fall detection in streaming videos. The cascade framework contains efficiently designed features from simple to complex and can reject the negative data in an effective way which matches the rare property of fall action. With our contributed dataset that consists of 400 hours' indoor videos, our trained cascade detector achieves high performance with a false-alarm rate of 10^{-3} and a hit rate of 95% in real time. Our framework achieves significantly higher performance than state-of-the-art approaches both on accuracy and speed, which further verifies the effectiveness of our framework.

5. REFERENCES

[1] X. Yu, "Approaches and principles of fall detection for elderly and patient," in *HealthCom 2008*. IEEE, 2008, pp. 42–47.

[2] M. Mubashir, L. Shao, and L. Seed, "A survey on fall detection: Principles and approaches," *Neurocomputing*, vol. 100, pp. 144–152, 2013.

[3] Noury et al, "Fall detection-principles and methods," in *2007 IEEE Engineering in Medicine and Biology Society*. IEEE, 2007, pp. 1663–1666.

[4] Henry et al, "Detection of falls among the elderly by a floor sensor using the electric near field," *IEEE Trans. Information Technology in Biomedicine*, 2010.

[5] H. Qian, Y. Mao, W. Xiang, and Z. Wang, "Home environment fall detection system based on a cascaded multi-svm classifier," in *ICARCV 2008*. IEEE, 2008, pp. 1567–1572.

[6] C. Rougier, A. St-Arnaud, J. Rousseau, and J. Meunier, *Video surveillance for fall detection.*

[7] S. Wang, L. Chen, Z. Zhou, X. Sun, and J. Dong, "Human fall detection in surveillance video based on pcanet," *Multimedia Tools and Applications*, pp. 1–11.

[8] Y. Yun and I. Gu, "Human fall detection in videos by fusing statistical features of shape and motion dynamics on riemannian manifolds," *Neurocomputing*, 2016.

[9] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *CVPR 2001*. IEEE, 2001, vol. 1, pp. I–511.

[10] Y. Ke, R. Sukthankar, and M. Hebert, "Efficient visual event detection using volumetric features," in *ICCV'05 Volume 1*. IEEE, 2005, vol. 1, pp. 166–173.

[11] Ivan Laptev and Patrick Pérez, "Retrieving actions in movies," in *2007 IEEE 11th International Conference on Computer Vision*. IEEE, 2007, pp. 1–8.

[12] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *Computer Vision (ICCV)*. IEEE, 2013, pp. 3551–3558.

[13] H. Wang, M. Ullah, A. Klaser, I. Laptev, and C. Schmid, "Evaluation of local spatio-temporal features for action recognition," in *BMVC 2009*, pp. 124–1.

[14] N. Dalal, B. Triggs, and C. Schmid, "Human detection using oriented histograms of flow and appearance," in *ECCV*. Springer, 2006, pp. 428–441.

[15] O. Barnich and M. Van Droogenbroeck, "Vibe: A universal background subtraction algorithm for video sequences," *IEEE TIP*, vol. 20, no. 6, 2011.

[16] Z. Cheng and et al., "Human daily action analysis with multi-view and color-depth data," in *ECCV Workshop.*, 2012.

[17] E. Auvinet, C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau, "Multiple cameras fall dataset," 2010.

[18] Auvinet et al, "Fall detection with multiple cameras: An occlusion-resistant method based on 3-d silhouette vertical distribution," *IEEE Trans on Information Technology in Biomedicine*, 2011.