SFU at TRECVid 2010: Surveillance Event Detection

Zhi Feng Huang
School of Computing Science
Simon Fraser University, Canada
zfh@sfu.ca

Greg Mori
School of Computing Science
Simon Fraser University, Canada
mori@cs.sfu.ca

Abstract

This paper describes the SFU entry for the TRECVid 2010 surveillance event detection challenge. We focus on detecting the "PresonRuns" event. The system has three stages. First, a pre-processing step is performed to extract moving candidate space-time regions using background subtraction, optical flow, and photogrammetric context. A short clip representation is constructed for each moving candidate region with a human figure detection. Motion features are extracted from the clip representation, and the features are scored by an AdaBoost classifier. Finally, a post-processing step produces temporally localized responses for evaluation.

1. Introduction

In the TRECVid 2009 event detection evaluation, Yang *et al*. [1] have made an attempt to detect events that comprise one or two individuals, namely "person running", "embrace", and "pointing". For this year's entry, we develop an enhanced system based on their framework. Instead of attacking the same events from last year, we focus on the "person running" event.

The dataset used for this year is unchanged from last year. It consists of surveillance camera footage which was acquired at London Gatwick airport. The development dataset is an approximately 100 hour video dataset, which consists of videos from five fixed-view surveillance cameras in the airport. The event detection only requires locating times (*i.e*. frames) at which the events occur, rather than accurate spatial locations. The final evaluation has an approximately 45 hour video dataset.

Compared with the benchmark human action datasets in common usage in the computer vision literature (*e.g*. KTH and Weizmann datasets), TRECVid is much more difficult and realistic. We list the main challenges of TRECVid below.

- **Rarity of events**: Detecting events in surveillance videos is a rare event detection problem. Finding instances of an event such as "person running" is akin to searching for a needle in a haystack. Most of people in the videos are walking and standing. Since there are too few positive samples (*i.e*. running events) compared with negative samples (*i.e*. walking events), the classification problem between running and walking becomes very difficult.

- **Cluttered background and occlusion**: The TRECVid videos all have cluttered backgrounds, which not only contain static background objects, but also dynamic ones, *e.g*. moving people. In addition, instances of events are often occluded by other people. Tracking in such environment is not reliable. Our system requires to track moving people (Fig. 1).

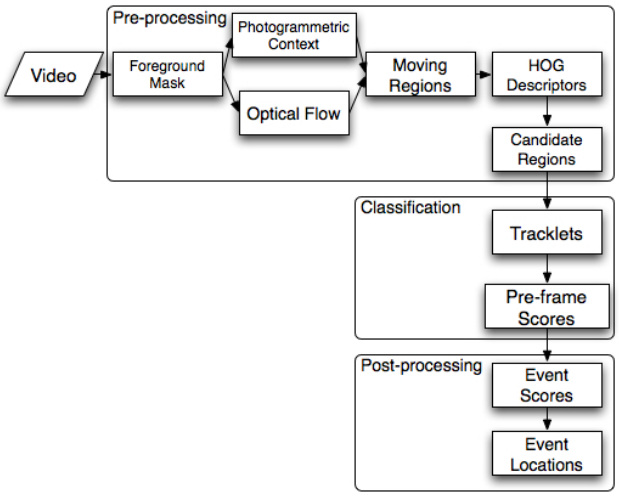

Figure 1. The general flowchart of our event detection system. Given an input video, we first extract the candidate moving regions by background subtraction, optical flow, and photogrammetric cue. We then use a HOG detector to find out if there are people inside the candidate regions. We then perform tracking and classification on the candidate regions with people, whose results are further processed by the post-processing step.

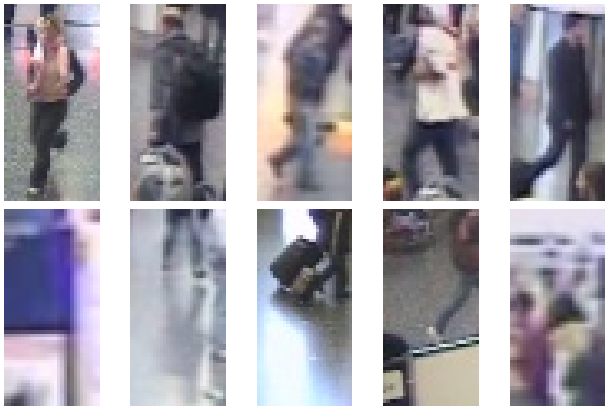

Figure 2. Sample images collected in moving regions. The first row contains the person images. The second row contains the not-person images.

- **Action variation**: Different people may perform the same action differently. Unlike many benchmark datasets, the actions in TRECVid are not choreographed, instead the captured actions are natural and realistic. This leads to a very large variation for the same pre-defined action. Furthermore, actions are captured from a variety of different camera viewpoints.

- **Incomplete annotation information**: For the development videos, the annotations only provide temporal locations of the pre-defined events, and no spatial bounding box information is provided. Our system requires the bounding box information to train a classifier.

Our system has three steps, a pre-processing step, a detection step, and a post-processing step. The detection step, following a typical machine learning framework, includes feature extraction, training, and classification components. The system uses a HOG detector to scan over the candidate moving regions returned from the pre-processing step. For each human figure found by the HOG detector, the system performs a short tracklet [5] of 15 frames by using both the HOG feature and color histogram. The classification framework by Fathi and Mori [3] is applied to each successful tracklet. There are only two classes in the classification problem, either running or not-running. The classifier in the framework gives every tracklet a score, which can be considered as a confidence measure of the classification. The system records the maximum score among all the scores from all the tracklets in each frame. The post-processing step thresholds the list of scores from the detection step to generate a list of discrete events for evaluation.

The reminder of this paper is organized as follows. In section 2, we present the system in detail. In section 3, we present some experimental results and the evaluation result on the formal run of TRECVid 2010. Finally, we conclude the paper in section 4.

2. Event Detection System

Given a pre-defined event, the objective of event detection is to temporally localize all similar events in test videos. We develop an event detection system for the TRECVid workshop evaluation that first detects running events in specific spatial and temporal locations in test videos, and then post-processes them to produce temporally localized output for evaluation. The system is based on the framework developed by Yang *et al*. [1] in 2009. We only keep their background subtraction, optical flow, and photogrammetric context components. The details of each step are provided below. An overall flowchart view of our system is provided in Fig. 1. Note that the system is implemented in C++ with OpenCV. The tracking and classification of each candidate region is independent. Thus, we use multithreads to speed up the processes in the detection step. The system takes 0.5s to 7s to process one frame depending on the number of candidate regions returned from the pre-processing stage.

2.1. Pre-processing

Our system uses a space-time window-scanning to locate people first. To reduce the searching space, we apply a pre-processing step using background subtraction, optical flow, and photogrammetric context to discard the majority of static (*i.e*. no motion) regions in a video. Foreground regions are detected using the standard Gaussian Mixture Model (GMM), and those which do not cover enough foreground are discarded. To further reduce the searching space, we employ photogrammetric context information to roughly estimate the human height on images and determine where events are likely occur. The photogrammetric context component is a linear regression model which estimate a person height by giving the person distance to the camera. The optical flow of every two adjacent frames is computed. Then the motion descriptors proposed by Efros *et al*. [4] are computed for each frame. The motion descriptors are four non-negative channels F_x^+, F_x^-, F_y^+, F_y^- represented the motions in horizontal and vertical directions. For every bounding box given by the photogrammetric context component, the system sums over the motion descriptors inside the box. A bounding box with motion is then evaluated by a HOG [6] detector (Sec. 2.2). The bounding boxes with moving people are considered as candidate regions and are passed to the tracking step (Sec. 2.3). The classification step (Sec. 2.4) scores a tracklet from the tracking component and it gives a score to a frame by recording the maximum score in the frame. Finally, the list of scores is post-processed to produce temporally localized output (Sec. 2.5).

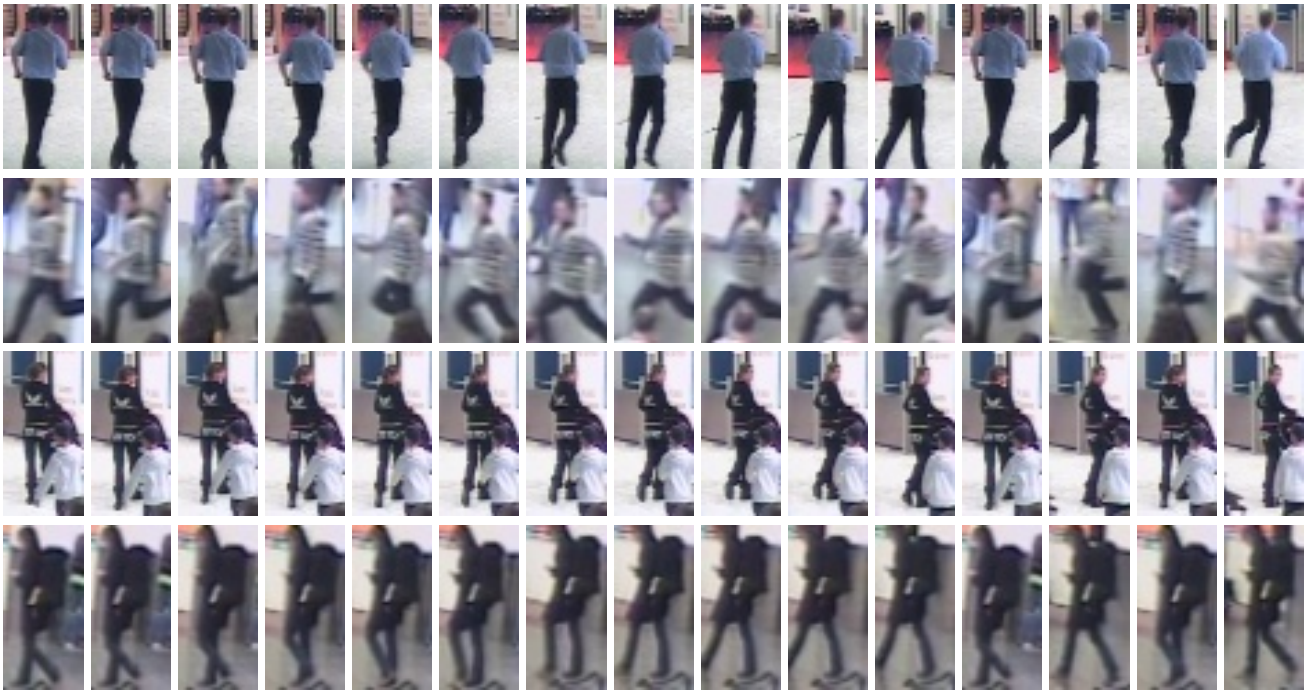

Figure 3. Sample tracklets. The first row is a tracklet of a running person in camera view 1. The second row is a tracklet of a running person in camera view 2. The third row is a tracklet of a walking person in camera view 1. The last row is a tracklet of a walking person in camera view 2.

2.2. HOG Detector

After the pre-processing step, the system has a list of candidate moving regions. The regions are defined by the bounding boxes given by the photogrammetric context. However, a region that has motion may not contain a person. For example, it can contain the shadow of a moving person, or it can contain a moving luggage. To further reduce the searching space, we only interest in the moving regions that contain people. Given a region (*i.e*. a bounding box), we first resize it to 29×60. We then extract the HOG feature from the resized image. A linear SVM is used to classify the HOG feature, either person or not-person. The linear SVM is from liblinear [2]. There are usually several moving regions detected for the same person. To solve this non-maximum suppression problem, we cluster the regions based on color histogram and spatial constraints. For each cluster we take only the region with the largest score from the SVM among all the regions in the cluster. After this pedestrian detection step, a list of candidate regions with moving people is passed to the tracking component for further analysis.

Some sample images of both person and not-person are shown in Fig. 2.

2.3. Tracking

The AdaBoost classifier in Sec. 2.4 requires a figure-centric volume. To construct such representation, we perform a short tracking of 15 frames on each candidate moving region that contains a person. The tracking algorithm we implement is very basic. We perform an exhaustive searching on each of these four directions: right, top, left, and bottom. The exhaustive searching is done in a frame-by-frame manner. In one tracking, given a moving region that contains a person in the current frame, we assume that the person either stays in the same region or moves no further than a certain distance (*i.e*. 10 pixels) in next frame and that the person does not change his or her moving direction rapidly. The certain distance defines a fixed searching space in the next frame for a given region and a given direction. We then use the same technique in Sec. 2.2 to try to find the same person in the searching space. If the same person is found in the next frame, the next frame becomes the current frame and the tracking is continued with the same procedure. The tracking is terminated when either a tracklet of 15 frames is constructed or the tracking person cannot be found by the system in the next frame. The tracklets that have fewer than 15 frames are discarded.

The tracking result is very good if the tracking person does not go behind other people or objects such that a large occlusion of the tracking person occurs. Some typical sam-

ple tracklets are shown in Fig. 3. Note that every image shown in Fig. 3 has a size of 29×60 since a detected region is resized to such size during the tracking (see Sec. 2.2). However, during the computation, the system only keeps the original size of each bounding box in a tracklet. The original size information is used to scale the low-level motion features extracted later during the classification described in next section.

2.4. Classification

After the tracking step described in the previous section, the system has a set of figure-centric tracklets. The tracklets can be directly fed into the classification framework by Fathi and Mori [3] with a small modification. In the classification framework, a variant of AdaBoost [7] is used, and we implement the same AdaBoost classifier. The classifier works on a figure-centric volume and every images in the volume should have the same size. However, the tracklets returned from the tracking component in Sec. 2.3 do not have such property. We decide to resize every image in a tracklet to 29×60. The resizing also requires to scale the low-level motion features in every image. The low-level motion features are the four non-negative motion channels F_x^+, F_x^-, F_y^+, F_y^- similar to Efros *et al*. [4] and a zero motion F_0 which is obtained by computing the L_2 norm of the four basic channels. The low-level features should have the same values if they represent the same amount of motion in the real world. In the system, the low-level features are computed from the optical flow of two adjacent frames in the video. The optical flow of the pixels that are far from the camera has smaller value than the the optical flow of the pixels that are close to the camera even if they represent the same amount of motion in the real world. To fix this issue, we first resize a given region to 29×60 by a linear interpolation and then scale the low-level features inside the region as the following. Let the original size of a region be $W_o \times H_o$ and the four non-negative channels be F_x^+, F_x^-, F_y^+, F_y^-. The two scaled horizontal channels are

$$\hat{F}_x^+ = F_x^+ \frac{29}{W_o}, \hat{F}_x^- = F_x^- \frac{29}{W_o} \quad (1)$$

and the two scaled vertical channels are

$$\hat{F}_y^+ = F_y^+ \frac{60}{H_o}, \hat{F}_y^- = F_y^- \frac{60}{H_o} \quad (2)$$

The scaled zero motion $\hat{F}_0$ is obtained by computing the L_2 norm of the four scaled channels.

After the resizing and scaling, the low-level motion features and the mid-level motion features of a tracklet can be extracted. These features are scored by the AdaBoost classifier. The AdaBoost classifier solves a two-class classification problem, running versus not-running. It computes a score for each tracklet found in a frame. The score can be considered as a confidence measure of the classification. A higher score means that the tracklet is more likely to contain a running person. For every frame, the system records the maximum score among all the tracklets found in the frame. The list of scores is then passed to the post-processing step.

2.5. Post-processing

The post-processing step thresholds the list of scores from the detection step to generate a list of discrete events for evaluation. Given a pre-defined threshold, the system merges a set of consecutive frames into one event. The starting frame and the ending frame of an event have scores larger or equal to the threshold. It is not necessary that every frame within an event has a score larger or equal to the threshold. The system merges a frame into an event if the gap between the frame and the last frame of the event is less than a pre-defined number of frames (*i.e*. 25 frames). This allows the system to tolerate some errors introduced by the tracking. For the same reason, the system also adds a pre-defined number of frames to both the starting and the ending of an event. The score of an event is the mean of the scores of the frames in the event which are larger or equal to the threshold. The system sorts the list of discrete events based on the scores of the events in descending order. It then takes $x\%$ events (*i.e*. $x = 10$) from the top of the sorted list for computing the actual DCR score in the evaluation. Since the AdaBoost classifier for each camera view is trained independently, the post-processing step also tries to normalize the event scores across camera views. We use a simple heuristic way to put the event scores into the same range. The system finds all the generated events for each camera view and calculate the mean event score $m_i (i = 1, 2, 3, 5)$ for each camera view. Each event score in camera view i is then divided by the mean event score m_i.

3. Experiment on Development Data and Evaluation Data

To evaluate our system, we train a HOG detector and an AdaBoost classifier for every camera view except camera view 4. There is very little activity in camera view 4. We focus on detecting running events that involve adults only. The TRECVid 2010 development dataset contains two parts: TRECVid 2008 development dataset and TRECVid 2008 evaluation dataset. Each set has 5 videos and each video has a length of 2 hours. We only use the TRECVid 2008 development dataset for our training. We first extract all running events from the dataset. Then, we separate the events in each camera view into two groups: adult and child. The adult group contains the running events such that the running person is an adult. The child group contains the running events such that the running person is a child.

To train the HOG detectors, we manually select person

	Person image	Not-person image
Camera view 1	2010	1450
Camera view 2	2308	2595
Camera view 3	2687	3651
Camera view 5	3016	3340

Table 1. The number of images selected to train the HOG detector for each camera view.

	Running tracklet	Not-running tracklet
Camera view 1	295	5721
Camera view 2	635	8698
Camera view 3	415	3554
Camera view 5	661	4345

Table 2. The number of tracklets selected to train the AdaBoost classifier for each camera view after the bootstrapping step.

images and not-person images from the extracted events in the adult groups. The images are generated from moving regions. Details of how many images have been selected is stated in Table 1. The kernel parameter C in the linear SVM is set by cross-validation. To train the AdaBoost classifiers, we manually select running tracklets and not-running tracklets from the extracted events in the adult groups. After the first round training, we perform a bootstrapping step to collect more not-running tracklets. We run the trained system on the TRECVid 2008 development dataset and select the not-running tracklets in the false positives that are given high scores. Details of how many tracklets have been selected in total is stated in Table 2. There are a few parameters in the AdaBoost classifier. The AdaBoost classifier is a two-layer AdaBoost classifier. The first layer consists of low-level features and the second layer consists of mid-level features. Each mid-level feature is a linear combination of 10 low-level features. The final classifier is a linear combination of 50 mid-level features. Each mid-level feature is represented by a cuboid in the figure-centric volume. The size of the cuboid is $5\times5\times15$. More detail of this AdaBoost setting can be found in [3].

We compare our system with the system from Yang *et al*. [1] in 2009. We run both systems on the TRECVid 2008 evaluation dataset, a part of the TRECVid 2010 development dataset. We only run the systems on camera view 1, 2, 3, and 5. However, we evaluate the systems with the ground truth from all camera views. The DET curves of these two systems are plotted in Fig. 4.

The result of our system on the TRECVid 2010 evaluation formal run is shown in Table 3. The baseline system has a similar minimum DCR on the formal run of TRECVid 2009.

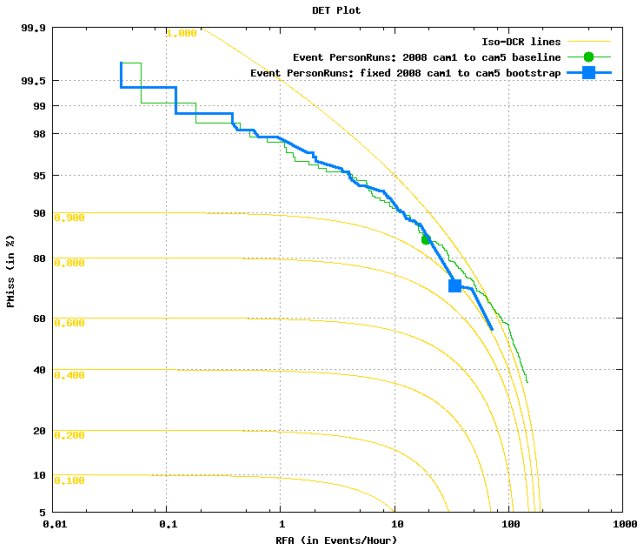

Figure 4. DET curves of our system and the baseline system from Yang *et al*. The blue curve represents the performance of our system, while the green one represents the performance of the baseline system.

	Actual DCR	Minimum DCR
Running	1.058	0.982

Table 3. Actual DCR and Minimum DCR of the running event on the formal run.

4. Conclusion

We present a system for the TRECVid 2010 event detection challenge. We focus on the "PersonRuns" event. The system utilizes pre-processing to extract candidate space-time regions based on background subtraction, optical flow, and a photogrammetric context cue. These candidate regions are further reduced by a HOG detector. For a candidate moving region with a person, the system performs a short tracking and a successful tracklet is scored by an AdaBoost classifier. Given a list of scores, a post-processing step produces temporally localized responses for evaluation.

References

[1] W. Yang, T. Lan, and G. Mori. SFU at TRECVid 2009: Event Detection. In *TRECVID Workshop*, 2009. 1, 2, 5

[2] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A Library for Large Linear Classification. In *Journal of Machine Learning Research 9, 1871-1874*, 2008. 3

[3] A. Fathi and G. Mori. Action Recognition by Learning Mid-level Motion Features. In *CVPR*, 2008. 2, 4, 5

[4] A. Efros, A.Berg, G. Mori, and J. Malik. Recognizing action at a distance. In *Proc. 9th Int. Conf. Computer Vision*, volume 2, pages 726-733, 2003. 2, 4

[5] V. K. Singh, B. Wu, and R. Nevatia. Pedestrian Tracking by Associating Tracklets using Detection Residuals. In *IEEE Workshop on Motion and Video Computing*, 2008. 2

[6] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *CVPR*, 2005. 2

[7] R. E. Schapire and Y. Singer. Improved Boosting Algorithms Using Confidence-rated Predictions. In Machine Learning, 37(3):297-336, 1999. 4