PKU_OS @ TRECVID 2012 : Surveillance Event Detection

Liqun Peng[*], Yaowen Guan[*], Haiqian He[*], Xu Chen[†]

[*]*Institute of Computer Science & Technology,Peking University*

Beijing, 100081, P.R China

[†]*School of Mathematical Sciences,Peking University*

Beijing, 100081, P.R China

Abstract

In this paper, we describe our system for surveillance event detection task in TRECVID 2012. "PersonRun", "Pointing" and "ObjectPut" are the 3 events we detect in our system. For "PersonRun" event, there are three steps to make a decision. Firstly, background subtraction and person detection are applied to the frame sequence. Secondly, particle filter is used to track the moving humans. Thirdly, the system makes a decision based on the result of trajectory analysis. In "Pointing" event detection, a novel feature is proposed and a classifier using support vector machine is applied. To detect "ObjectPut" event,the downward optical flow is calculated and a further analysis step is utilized. We get more familiar with the techniques used in this domain after this evaluation.

1. Introduction

Recently, the amount of accessible surveillance video recording is increasing rapidly, especially in public areas such as roads,airports and banks. Thus automatic detection of semantic events in video will be very useful. The TRECVID SED task aims at detecting visual events in a large collection of streaming video data. It is the first time our team participates in TRECIVD evaluation and the SED task. We went through some previous notebooks for SED task by other teams and realized that the problem is not trivial. Different methods have been proposed in the task and they share two common features. First the modeling of high-level events directly on low-level features to get a unified framework like Informedia@TRECVID 2011. Second some of these systems tend to engineer the analysis process with very specific domain knowledge and deal with the detection of each event separately. Our system belongs to the second one. Therefore we will briefly describe the methods we used to finish each task separately. This paper merely demonstrates the approaches used in retrospective task for there is not enough time for us to finish the interactive task.

In the following section, we will describe our methods to detect *PersonRuns*,*Pointing*,*ObjectPut* events. After that, the experimental results will be given. Finally, we will come to the conclusion of this paper.

2. Our Methods

2.1. PersonRuns

We implement the standard pipeline proposed by many other teams with some changes for detecting the *PersonRuns* event only, which incorporates person detection, person tracking and trajectory analysis models.The system framework is illustrated in Figure 1.

Firstly, videos are decoded to separate frames. Both background subtraction and person detection are applied for a frame. Secondly, after a candidate tracking region generated based on person detection, a new tracker is created for tracking the person. Lastly, we use some rules to analyse the tracked trajectory and generate the output result. We will give some details in the remaining of this section.

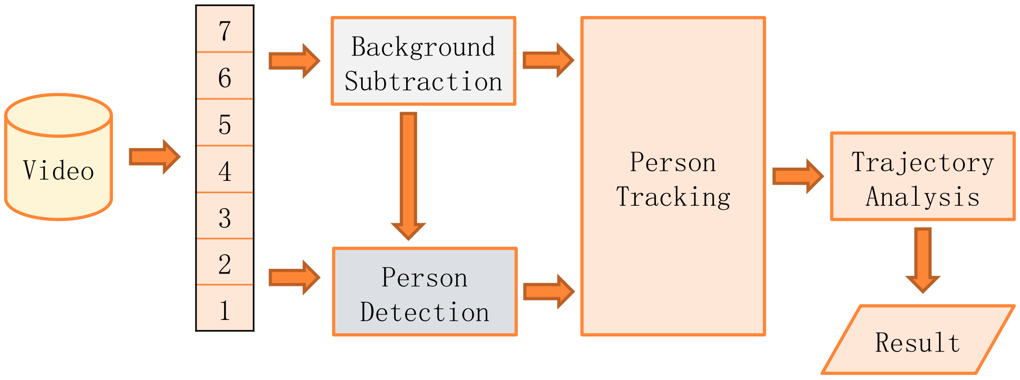

Figure 1. The framework of *PersonRuns* event detection

2.1.1 Background subtraction

In surveillance video event detection, since the background did not change much, the background subtraction procedure will greatly filter the background objects and remain the foreground objects, especially rapidly moving humans. We compared various background subtraction methods and used the one prospered by Z.Zivkovic [5]. A MoG model is constructed for each pixel in each frame and the number of modes of Gaussians is adaptive to the $\frac{w}{\sigma}$, where w is the weight of the mode and σ is the standard deviation. Shadow detection in paper [2] is applied to remove background noise.

In the experiments, we found that the final result is sensitive to the update rate of the background model. If the model updates too slowly, it will get low detection sensitivity and otherwise it will adapt to the targets themselves easily. Since we just focus on the rapidly moving objects (*e.g.* person run.), a big update rate is preferred.

2.1.2 Person Detection and Tracking

Person Detection is an important step in our system. We apply head-shoulder detection instead of human body detection for there are many occlusions in the TRECVID datasets.Histogram of Oriented Gradient feature [1] is the most popular features used in person detection. In practice, we use about 1500 head-shoulders as positive training samples and 2000 other images without head-shoulders from the video image sequences as negative training samples.Then the HoG feature extractor is used on the samples to get the feature vector with the demission of around 6000. At last, the linear SVM classifier is used for classification. After detection, we use Non-maximum suppression method [3] to fuse the result regions.

Detection on the full image is very time-consuming and not necessary. IRDS-CASIA in TRECVID 2011 built a probability map for every event in every scene. We took this map and the scene structure in consideration and just applied detection algorithm on the predefined fixed region of a frame. The regions for each camera is shown in Figure 2.(Since camera 4 has no *PersonRuns* event, it is omitted.) We also remove candidate regions that have little foreground with the help of background subtraction module.

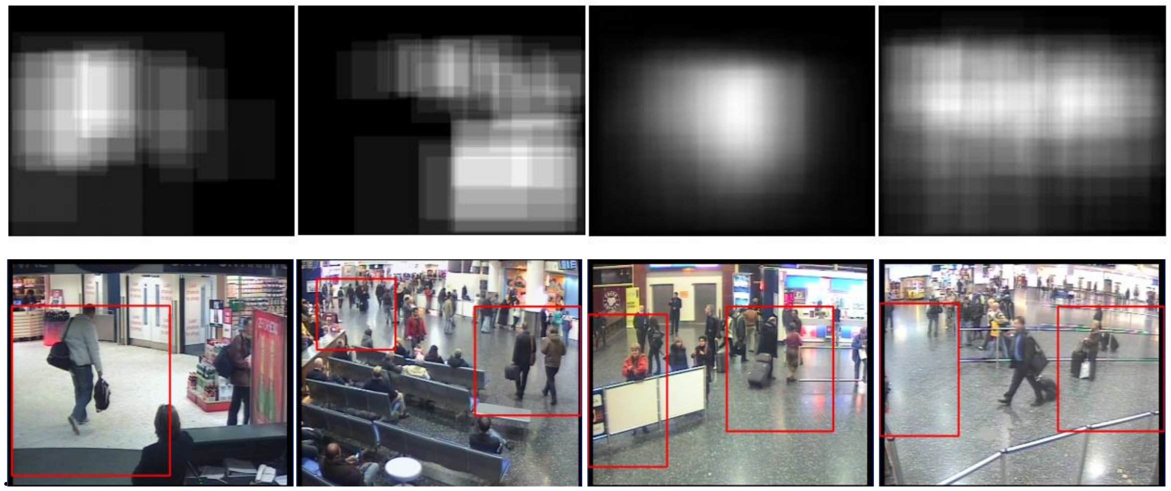
Figure 2. the probability maps of the 4 different scenes (the first row)and the predefined regions (the second row)

Every time a new person is detected, the system creates a new tracker to follow the person. The tracking method we use is particle filter. We utilize a motion model based on motion estimation from paper [4]. But we reduce the velocity coefficient predicted from the preceding time step. The motion model is given by

$$\begin{cases} x_t = x_{t-1} + 0.5\varphi_x + P \\ y_t = y_{t-1} + 0.5\varphi_y + Q \end{cases} \tag{1}$$

where $\varphi_x = x_{t-1} - x_{t-2}, \varphi_y = y_{t-1} - y_{t-2}$,$P$ and Q are assumed to be a zero-mean white Gaussian sequence. We fuse background subtraction result, color information and appearance information to calculate the likelihood model for each particle.

Another problem is when to remove a particle filter. We take the following conditions into consideration to roughly remove some filters that is definitely not person run. If one of the following conditions is satisfied in several successive frames, the system will remove the corresponding filter.

- The person walks slowly: the distance the person moved is too small.
- The person walks out of the scene: the position of the person is close to the boundary of the scene.
- The filter tracks to the background: sometimes the filter will tracks to the background,but it maybe tracks back to the person in a short time. If the filter tracks to the background too long,it can hardly turn around. Thus we remove it.

- The filter stays too long in the scene: Person runs in such a limited scene will soon disappear in sight therefore the filter stays too long will be deleted.

2.1.3 Trajectory analysis

It is time to analyse the trajectories after tracking. Each trajectory is a sequence of <frame-index, x, y, weight, is-background >. We use some empirical rules to handle this problem and then use the development corpus to tune the parameters and output the decision score for each trajectory sequence. The following specific rules are considered: The length of the trajectory should not be too big or too small. The speed with smooth is the most important rule to generate the final decision score. Only the overall trajectory direction within a certain directions will be kept. The rules used in removing a particle filter will also be used with a stricter criterion. After all this processing, we will merge the remaining trajectories by location and frame index.The overall result of *PersonRuns* can be shown in the Experimental Results section.

2.2. Pointing

To detect *Pointing* event, a classifier is employed. We first adopt an adaptive background mixture model [5] to gain the foreground information. Then edge extraction method is applied to the binary image, as Figure 3 shown.

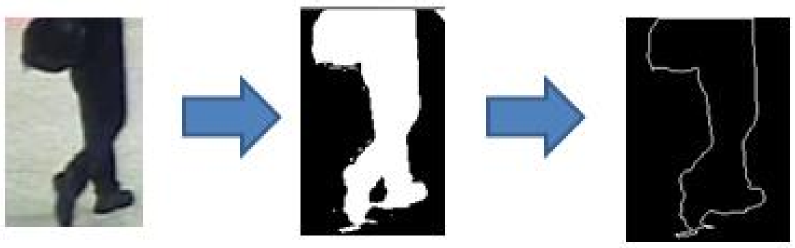

Figure 3. Background subtraction & edge extraction example

Our approach to object detection is based on scanning a detection window over the image at multiple positions and scales and in each position runs an object/non-object classifier. We calculate features from detection window as Figure 4 shown. Support vector machine is used to obtain the classifier.

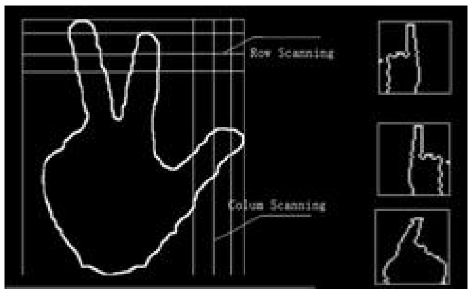

Figure 4. Calculate features from detection window and some positive examples

Once the fingers are recognized in the image, color and temporal information are merged to detect *Pointing*. In Table 1 , the formal evaluation result is listed.

2.3. ObjectPut

For the *ObjectPut* event, we simply calculate the downward optical flow to get a confidence score. In order to reduce false alarms and computation, we set some ROIs for each camera. After calculating the optical flow in the ROI, we search the ROI with predefined rules that can restrain the results. The rules mainly consider three aspects of the optical flow; they are the length of the flow, the direction of the flow, the consistency and duration of the flows in a small area.

The length of the flow can represent the velocity of object moving and the direction of the flow can represent the direction of object moving. We use the consistency of flows mainly because it can remove some noise witch introduced by calculating the optical flow. Generally, the *ObjectPut* event only lasts a very short time, so we adopt the duration of the flows to remove other long time movings in the camera. The following section shows the results.

3. Experimental results

The overall result of *PersonRuns*,*Pointing* and *OjbectPut* in formal evaluation with the Actual Decision DCR Analysis is shown in Table 1. As for *PersonEun*, there is 8 correct detected events while the system outputs 356. The false alarm rate is fairly high. One reason is that there is too many steps for detection but each step is not trivial. The *Pointing* event is slightly better. However the system outputs little observation as we limited the decision score to get a lower false alarm rate. The performance would be better if we collected more positive samples for the classifier. It is extremely hard to detect the *ObjectPut* event for the event duration is too short and there are times when it is difficult for us the recognize the *ObjectPut* event. Thus the result for *ObjectPut* event is not good. Another reason is that the detection algorithm is too simple.

Table 1. The formal evaluation result

Title	#Targ	#NTarg	#Sys	#Cor!Det	#FA	#Miss	RFA	PMiss	DCR
PersonRuns	107	356	364	8	356	99	23.34852	0.925	1.0420
Pointing	1063	36	38	2	36	1061	2.36109	0.998	1.0099
ObjectPut	621	62	63	1	62	620	4.06631	0.998	1.0187

4. Conclusion

We propose the event detection model for TRECVID 2012 surveillance event detection. *PersonRuns*, *Pointing*, *ObjectPut* are the three events we detect. Since it's the first year our team participates in TRECVID SED task, we use many existing models to solve the problem and little novel model is proposed. In the next time, we hope we can bring forth some original ideas in this task.

References

[1] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[2] A. Elgammal, D. Harwood, and L. Davis. Non-parametric model for background subtraction. *Computer VisionłECCV 2000*, pages 751–767, 2000.

[3] A. Neubeck and L. Van Gool. Efficient non-maximum suppression. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 3, pages 850–855. IEEE, 2006.

[4] B. Yang, X. Pan, A. Men, and X. Chen. A robust particle filter for people tracking. In *Future Networks, 2010. ICFN'10. Second International Conference on*, pages 20–23. IEEE, 2010.

[5] Z. Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 2, pages 28–31. IEEE, 2004.