INTUVISION EVENT DETECTION SYSTEM FOR TRECVID 2008

Pradeep Yarlagadda[1] Meltem Demirkus[1,2] Kshitiz Garg[1] Sadiye Guler[1]

[1]intuVision, Inc.
10 Tower Office Park, Suite 200
Woburn, MA 01801

[2]McGill University
School of Computer Science
Montreal, Quebec, Canada H3A 2A7

{pradeep, meltem, kshitiz, sadiye}@intuvisiontech.com

Abstract:

In this paper, we describe the algorithms for three event detection tasks, namely "Elevator No-Entry", "Opposing Flow", and "Picture Take" and findings from applying these algorithms to the TRECVID airport data set. We ran a single run for each event detection task.

For *Elevator No-Entry* event detection we used Haar object pedestrian detection followed by histogram matching to find person not entering an elevator. Histogram matching is shown to perform well under pose variations as indicated by our results.

In the *Opposing Flow* event the main challenge was to handle severe occlusions and overlap. In dry run we used Haar based pedestrian detection algorithm but, in the actual run we decided to use background subtraction for detection and tracking. We minimized occlusions and overlaps caused by other objects by considering only the top portion of a door region. Also, heads and shoulder are smaller in dimension so reduce occlusion effects. Restricting the problem in this manner we were able to get good tracking and detected the tracks in the specified direction using our Panoptes "direction event spy". We incorporated a mechanism to reject outliers in direction event spy to better handle tracking errors which considerably improved the results.

Our *Picture Take Event* algorithm uses characteristic change in illumination caused by flash – large increase in image intensity followed by almost equal drop in intensity - for flash detection. Once a flash frame is detected we find the location of the flash and use simple matching algorithm to detect frames where the hands are steady. This approach is not suited for detecting take picture event without a flash, but since in most cases cameras and hands are just barely a few pixels in width and height, it is hard to detect a camera without the presence of flash.

1 Introduction

A variety of event detection tasks has been defined for TRECVID [Sme06] video event evaluation based on the available airport surveillance data set, ranging from multi-person level to body part and articulation level events. Based on our application focus in video content analysis for Security and Surveillance domain and our existing technology in single person level activity detection we identified "Elevator No-Entry" and "Opposing Flow" events as the core required events to participate in the evaluation. In both these events, the detection and tracking of persons in a crowded environment where several partial occlusions take place over one person's track poses a big challenge in maintaining a unique ID for each tracked person.

In crowded scenarios, objects merge and occlude each other. In such scenarios conventional detection methods such as background subtraction do not work well. Instead frame based object detection methods [Vio01, Lie02] or recognition based algorithms [Dal05, Serr05, Bil07] have shown to perform better. Due to computational reasons, we have chosen Haar-pedestrian algorithm [Vio01] which gives near real time performance for person detection. Additionally, we combine the algorithm with background subtraction for reducing false positives.

As the third required event we added the "Picture Take". Our approach is based on the detection of flash in a frame hence not suited for detecting picture take events involving detection of camera or articulated arm motion where no flash is used. In essence, our "Picture Take Event" relies on characteristic change in illumination produced by flash – large increase in image intensity followed by almost equal drop in intensity. Once a flash frame is detected we find the location of the flash and use it to find the frames where the hands are steady.

The remainder of the paper is organized as follows: In Section 2 we discuss the detection and tracking of people in crowded environments as the base component of event detection tasks. The algorithms we have used for three event detection tasks are described in Section 3. In Section 4 we present our experimental results on the TRECVID test data set and we make some concluding remarks in Section 5.

2 Person Detection and Tracking in Crowded Environments

In most surveillance applications, simple background subtraction is used for detection and subsequent tracking. However, in a crowded environment such as the airport sequences, objects merge and occlude each other frequently. In such scenarios, conventional background subtraction methods do not work as well. They tend to produce large bounding boxes including multiple persons and fail to resolve ambiguities that arise due to occlusions and merges. Over last few years, various single frame detection methods based on transform cascades [Vio01, Lie02] or recognition [Dal05, Serr05, Bil07] have shown promise for pedestrian detection in busy scenes with occlusion.

We have implemented the object detection algorithm in [Vio01] for people detection in crowded scenarios. While several other algorithms have been proposed and used for pedestrian detection most are too slow for real time application. For example Dalal and Triggs [Dal05] method which uses histogram of gradients to detect pedestrian, takes around 0.5 seconds for recognition of 128X64 size image frame. [Bil07, Serr05] uses biological inspired model for recognizing different classes including pedestrian but their algorithms are also slow. [Serr05] takes 2 seconds/frame while [Bil07] takes approximately 80sec. The algorithm we choose is near real time and has a high recall rate. This algorithm has been very successful in previously detecting human faces and has been used to detect other types of objects such as human hands [Bar05], people [Mon06], and upper body [Kru03]. The method uses simple Haar-features

to build an over-complete set of appearance features which are then used to train a classifier cascade using Adaboost training algorithm. Though training period is long, a trained classifier detects people at real-time speeds.

Figure 1 shows some of the results from pedestrian tracking. The algorithm is able to detect merged and partially occluded people as separate objects, as well as people of different sizes. It produces some false positives (see the coke bottles detected as pedestrian in right side of the image in Figure 1.c). This is a common problem with single frame detection methods. In [Vio03] authors have shown that use of motion information between two consecutives frames can considerably reduce the number of false positives.

(a) (b) (c)

Figure 1: Results of Pedestrian detection for 3 different scenes.

Although Haar classifier has a good recall rate it detects a lot of false p ositives due to its reliance solely on appearance cues. We have used motion cues (in the form of foreground information in each frame) to filter out the false positives generated by Haar classifier. In each window returned by Haar classifier, if the percentage of foreground pixels is less than a threshold then the window is rejected. Figure 2 contains three panels, the left panel showing the pedestrian detections obtained from the Haar Classifier in the dry-run data, the center panel shows the foreground and the right panel shows the detections left out after a filtering step based on foreground.

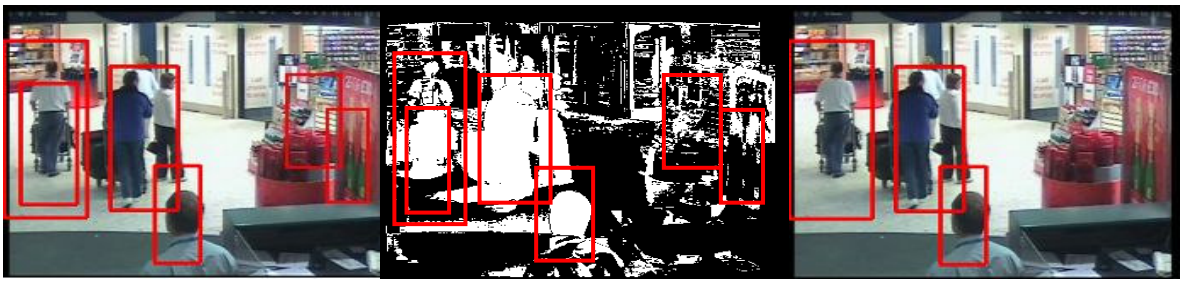


Figure 2: Results of person detection using foreground pixels to remove false positives.

3 Event Detection Algorithms

We chose the following events - “Elevator No-Entry”, “Opposing-Flow”, and “Picture-Take” for the TRECVID evaluation. Here we describe the algorithms used to detect these events.

3.1 Elevator No-Entry

Definition: The definition provided in the guidelines for the Elevator No-Entry event is as follows: Elevator door opens with a person waiting in front of them, but the person does not get in before the door closes. Start time of the event is the earliest time when the elevator doors are opening with person waiting in front of them. The end time is the earliest time that the doors of the elevator are fully closed.

Straight forward approach: A straight forward approach would be to track the person waiting in front of the elevator at the time the elevator door opens (say frame A) and checking if the person is still waiting in front of the elevator at the time the door closes (say frame B). However, this approach fails due to unstable tracking. While pedestrian detection algorithm works quite well, it misses detection in few frames, losing the track of an object and thus missing true events. Moreover, the dimensions of bounding boxes and variations in poses make tracking unreliable.

An alternative approach that does not rely on tracking is to match people waiting in front of the elevator at the time the door opens (frame A) to people waiting at the time the door closes (say frame B). An alarm is raised if a good match between the persons in frames A and B is found. The problem with this approach is that standard matching techniques such as template, and edge based matching fail to work under wide variations in object pose – a common occurrence in elevator no entry events. Figure 3 shows some frames between the Elevator door open and Elevator door close frames. Note the wide variation in person's pose as he waits near the elevator. Both the above approaches fail to work as was experimentally verified.

Elevator No-Entry Algorithm

The above mentioned problems can be tackled by changing the basic strategy and matching persons across all frames that lie between the door open and close event, instead of relying on single frame matching or frame to frame tracking. Using all frames ensures that pose changes are smooth. This gives better matching scores and also makes matching robust to detection errors. We use histograms for matching which is shown to be more robust to pose changes and partial occlusion.

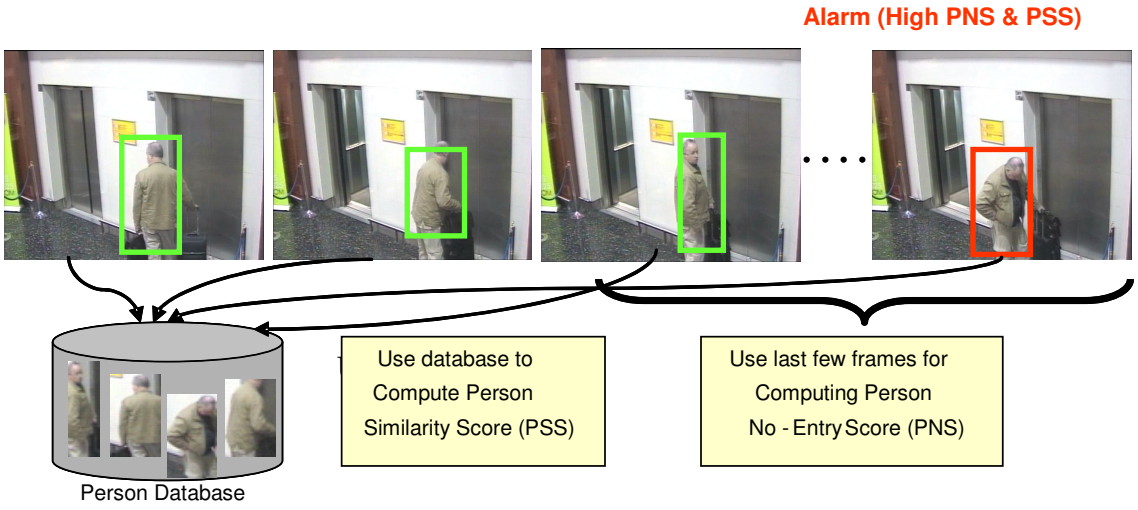


Figure 3: Algorithm for "Elevator Non Entry" event detection.

Our algorithm builds a database of people waiting at the elevator by applying pedestrian detection algorithm at each frame starting from door open to door close. The database is then used to compute a normalized "**Person Similarity Score"** by matching all pairs of objects in the database and dividing by the score with total number of matches. A higher person similarity score indicates that the person waiting near the elevator area throughout the frames is indeed the same person. It however, does not check the duration of stay. To find if the person entered the elevator before the door closed we calculate "**Person No-entry Score"** – that is the number of detections of the person in the last 10 frames preceding the elevator close event. The final score is a weighted combination of the normalized matching score and the consistency score. Below we briefly describe the main steps of the 'Elevator No Entry' event detection algorithm. Figure 4, shows the main steps of the algorithm.

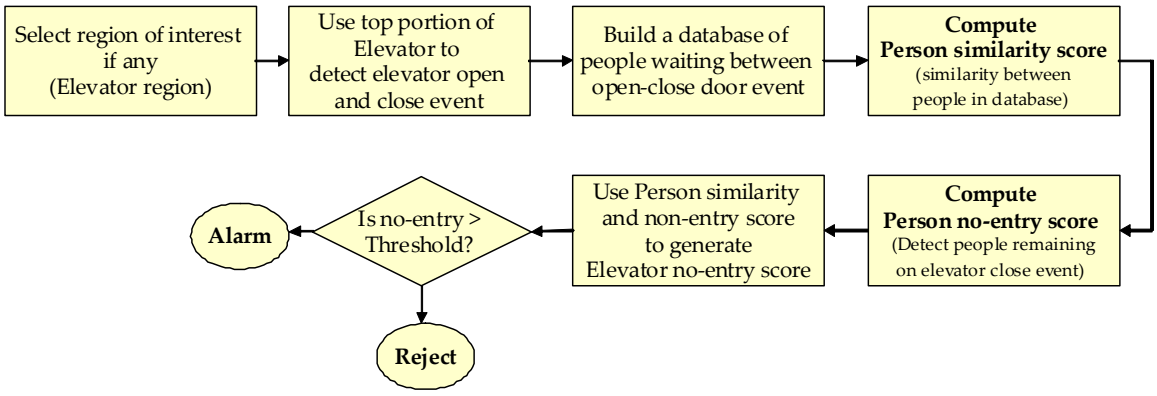


Figure 4. Basic steps of Elevator No Entry Event detection algorithm.

1. Select a region of interest (typically a rectangular region around an elevator). Selecting a region improves the speed and avoids interference from other near-by objects.
2. Use the top portion of the elevator to detect the frames A & B corresponding to the opening & closing of elevator door. We use simple comparison of the current frame of the video with the background image provided as part of the input data (Background image was obtained from running the median filter on the video).
3. Run the pedestrian detection system on the video between the frames A & B. The output of the system is a set of detection windows in each frame. The system output is used to build a database of detection windows.
4. Generate a Person Similarity Score (PSS) by finding the similarity score (SS) for all pairs of detection windows, summing up the scores and dividing by the total number of matches. The similarity score (SS) between a pair of windows is obtained by matching the color histograms of the respective windows.

   $$d(H_1, H_2) = \sum_I \min(H_1(I), H_2(I)) \ ,$$

   where, H_1 and H_2 denote the histograms of the windows and I spans across minimum bin intensity to maximum bin intensity.
5. Generate Person No-entry Score (PNS) by dividing the total number of detections in the last 'x' frames (usually 10) by the total number of frames.
6. Generate the final score as a weighted combination of similarity score and consistency score.

   $$S = W_1 * SS + W_2 * CS$$

W_1, W_2 have been empirically chosen as 0.7 & 0.3 respectively.

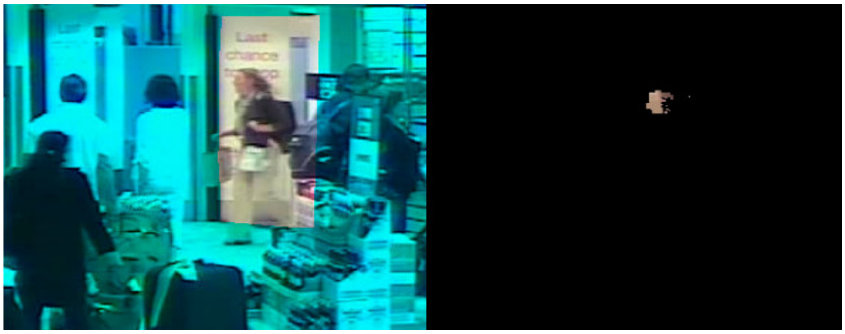
3.2 Opposing Flow

Definition: The definition provided in the guidelines for the Opposing flow event is as follows: Someone moves through a door opposite to the normal flow of traffic. Start time of the event is the earliest time when the person has begun to move through the door. The end time is when the person has fully passed through the doorway.

Straight forward approach: A straightforward approach to detecting an opposing flow event would be to track a person from the point the person appears at the door and track him till the person walks beyond the door. Motion direction can then be computed and if it matches opposing flow direction an alarm is raised. However tracking people in a crowded scene is a challenging task and the various difficulties in accomplishing this task have already been outlined. We tried two different approaches for opposing flow detection which are described below. The first algorithm (used in dry run) was improved upon to make opposing flow detection more effective in actual run.

Opposing Flow Algorithm for Dry Run

During the dry-run, Panoptes direction event spy [Int07] was used around the door region where opposing flow needs to be detected. The system uses the Adaptive background and Haar based pedestrian detection for detection and tracking of the pedestrians. We found that even though Haar-detection resolves partially merged pedestrian quite well, it does not provide good bounding box estimates and fails to detect objects in few frames. These errors produce unstable tracking (object IDs being frequently dropped or swapped on merges and occlusions). As a result, the motion direction estimates are often erroneous, producing many false alarms. To reduce these false positives we use skin color detection (see Figure 5). The idea is that since opposing flow is in the direction of camera (that is face is visible for true events and not for false alarms), a skin tone pixel color detector can be used to find number of pixels in top portion of rectangular people detection bounding boxes. A simple threshold on number of pixels can be used to eliminate some of the false opposing flow events.



(a) (b)

Figure 5. Example of opposing flow event (a) Un-shaded region shows the ROI (b) the detected skin pixels.

Opposing Flow Algorithm for Actual Run

The algorithm for actual run improves on two components of dry run algorithm. We realized that pedestrian detection algorithm used in dry run often misses objects and gives inconsistent bounding boxes, which produces unstable tracking results. Background subtraction on the other hand gives very stable tracking but does not work well with merges and occlusions. We minimized occlusions and

overlaps caused by other objects by considering only the top portion of a door region. Also, heads and shoulder are smaller in dimension so reduce occlusion effects.

User Inputs ROI and Opposing Flow Direction

Example of Opposing Flow Event. Using top portion minimizes chances of occlusion with near by objects, giving better tracking results.

(Frame n) ..... Alarm (Frame n + T)

Figure 6: Typical Opposing flow event detection.

We also made the direction spy more robust to tracking errors. We use a line fitting to find a best fit line to all the track data points. The best line fit is then used to reject outlier points which are far from the line. A best fit line is again fitted on the purged data giving much better and robust direction estimates. Below we describe the details of the algorithm. Figure 7 shows the basic steps of the algorithm.

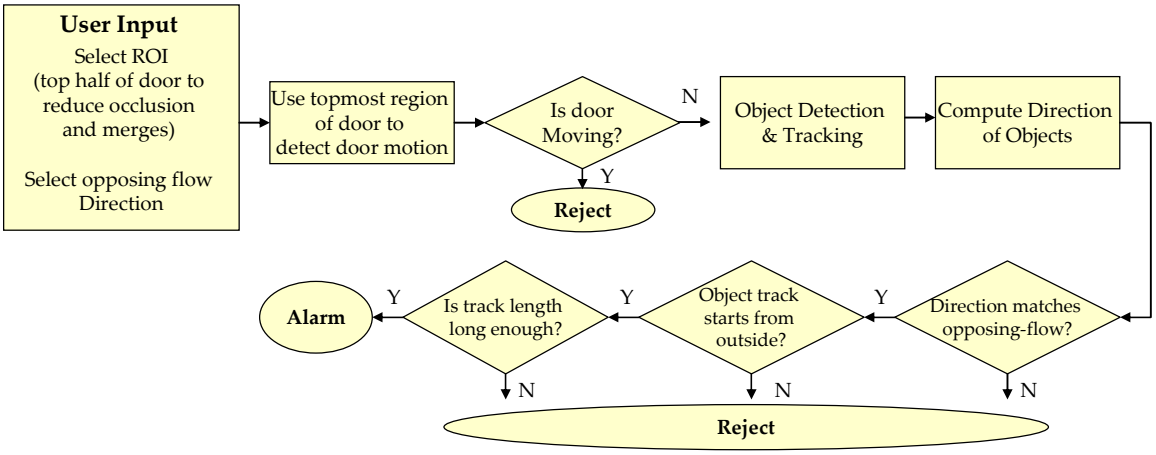


Figure 7: Basic steps of the Opposing Flow event detection.

Here are the main steps of the algorithm:

1. User selects the Region of Interest (top half of the door) and direction of opposing flow.
2. Use the topmost portion of the door to detect door motion. If door is in motion we don't process further. This step removes empty bounding boxes that arise due to motion of door.
3. If door is not moving track the objects within door region and compute the centroid of the bounding boxes.
4. Use the time ordered location of an object's center to find the motion direction. We fit a straight line to all the data points. This line is then used to find outliers (i.e. points 2-3 times farther than average distance from points). Once the outliers are rejected the direction of motion is again computed using the purged data. This way we obtain a more robust motion direction for objects.

5. Check if the line is within a tolerance angle to the direction of opposing flow, if yes then proceed further.
6. Check if the object tracks start from outside the region. This constraint removes some of the objects behind the door that are visible through the transparent glass window and false objects that may originate due to motion of door.
7. Check if length of the track is sufficient enough to trust the motion direction. If all the conditions 5, 6, 7 are met, alarm on the object.

3.3 Picture Take

Definition: Someone takes a picture. The start time is defined as the earliest time when a person holds a camera in a fixed position prior to activating it, while end time is defined as the earliest time when the camera moves away from a fixed position following the photograph.

For Picture Take event, we only consider picture events which are accompanied with a flash. Since in most scenes, cameras and hands are just barely a few pixels in width and height, it is hard to detect camera or articulated arm motion without the presence of flash. Our picture detection algorithm consists of two main parts. A flash detection part finds the flash frame, while hand detector finds the location of hands and computes the frames over which hands are stable. Here, we briefly describe the main steps of the algorithm:

Flash Detection

The algorithm relies on the fact that a flash causes sudden change in scene illumination, which results in flash frame to be much brighter than the neighboring frames and its transitory nature. In addition, the intensity pattern caused by flash is unique. It shows a bright region near the flash point which can be used to detect the location of flash source within the frame. The sudden change in intensity can be easily detected by computing frame difference among consecutive frames (see Figure 8). Flash produces much larger change in intensity than that produced by moderate level of activity within the scene. A running average can be automatically computed from the normal level of scene activity in a scene. Figure 8 shows the three flash frames and their differences. Flash produces a sharp change (order of magnitude higher than usual scene activity) in average intensity of frame. Moreover, a flash lasts for couple of milliseconds (i.e within a single frame). Therefore increase in intensity is followed by a similar decrease in average energy. Figure 8 shows difference among consecutive frames -- dark to bright and then bright to dark as shown in the plot in Figure 8. The plot on the right maps the average number of pixels with intensity greater than 20 gray levels with time. As one can see, flash produces a sharp increase followed by equally strong fall. Figure 10 shows the basic steps of the flash frame detection algorithm.

1. Flash Detection Algorithm

1. Compute frame difference between adjacent frames.
2. Threshold the difference image to find pixels with change in intensity greater than 20 gray levels and find the number of such pixels in a given frame.
3. Check if the number of pixels is greater than certain times (threshold that depends on camera view) the average number of difference pixels in typical frames. If true, proceed to next step otherwise use the current pixel number to update the estimate of average number of difference pixels.
4. Check if brightness change lasts for just a singe frame. If true, proceed to next step otherwise process a new frame.

5. Check if decrease in number of pixels after flash event is almost same as the increase in number of pixels, otherwise process a new frame.
6. If conditions 3, 4, and 5 are sufficiently met, a flash event is detected.

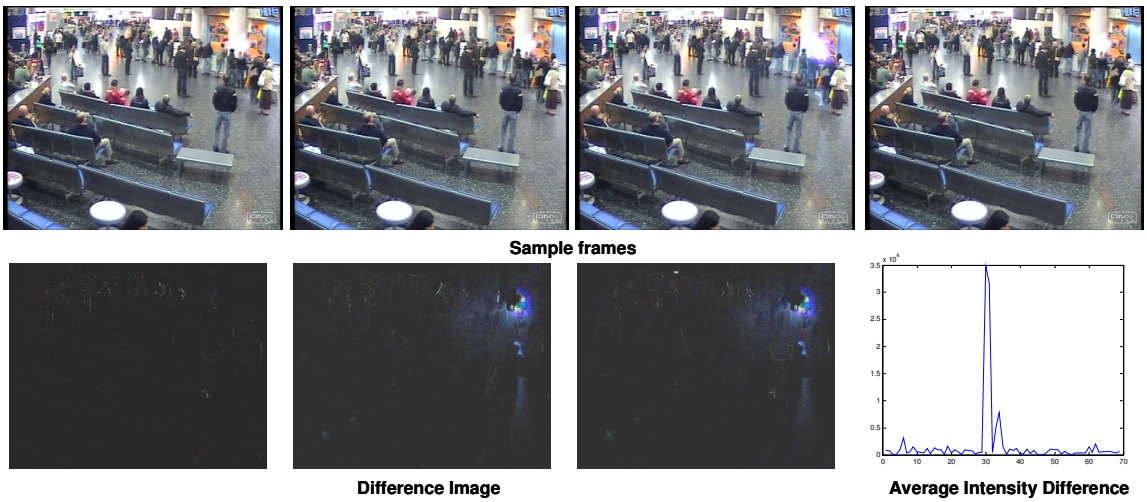


Figure 8. Sample frames (top row)and the difference between consecutive frames (bottom row), e.g., the first difference image corresponds to the difference between the first and second frames . The third frame contains flash event. The plot on bottom right shows the sharp change in pixel intensity on flash event.

2. Hand Stabilization detection

After flash frames are detected, we find the range over which the picture was taken by finding the frames over which a person's hand was stable as described in the event definition. We first compute the location of flash and hands within the frame by computing the centroid of the saturated pixel region in flash frame. The region below the bright flash spot is used as a person's hand template (please see small red rectangle in frames in Figure 9). A simple difference between template and corresponding region in other frames gives an estimate of hand motion over frames. The graph below shows the hand activity near the flash frame. A motion threshold depicted with dotted red line is used to find range of frames with no hand motion.

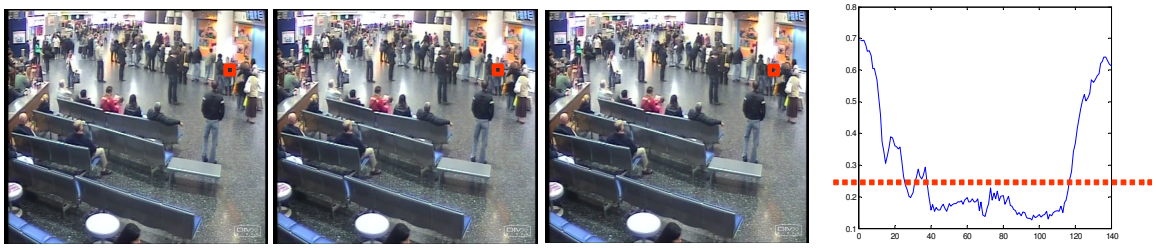


Figure 9. Hand motion is detected in a region (shown as red rectangle) and is plotted on the right. Red line shows the threshold used to find the range of frames with no hand motion.

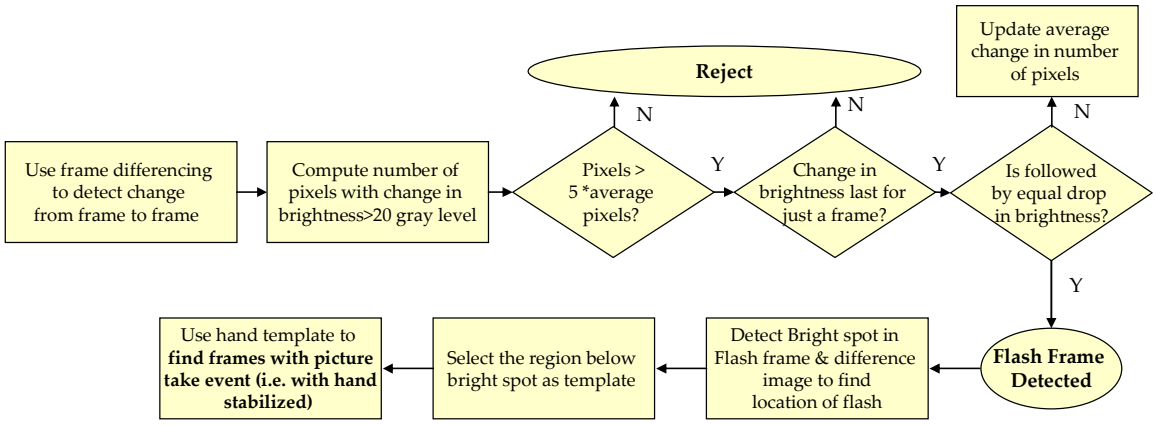


Figure 10. Basic steps of take picture event detection.

Here we briefly describe our Hand detection component part of the algorithm. The bottom row of Figure 10 shows the main steps of this component

1. Detect the bright spot in flash frame by finding saturated pixels and compute its centroid.
2. Select region below the centroid of bright spot and use the region as a hand template.
3. Compute difference between the hand template and corresponding region in other frames to get an estimate of hand motion.
4. The earliest frame with no hand motion is the start event while the last frame with no hand motion is the end of the event.

4 Experimental Results

Experimental results obtained on the airport surveillance video data provided by NIST as part of TRECVID 08 are provided in this section. All the alarms obtained by the intuVision event detection algorithms for the three chosen events are listed in a tabular form. Typical example events are provided in the figures.

4.1 Elevator No-Entry

Table 1 shows the result of our *Elevator No-Entry* algorithm. Figures 11 and 12 show selected frames for the elevator no-entry true events detected by our algorithm. Our algorithm detects all the true events; however it also detects a few false positives (the top 3 events in camera LGW_20071206_E1_CAM3 are false positives). Most of the true events have high confidence; however, the last event in LGW_20071206_E1_CAM3 has low confidence value. This is because the person detection system misses the person waiting in front of the elevator due to low contrast between the foreground and background. This results in missing the person in last few frames and hence a low person No-Entry score is generated, leading to a low confidence.

Figure 11. Person waiting in front of the elevator and triggering the No-Entry alarm in LGW_20071123_E1_CAM4.

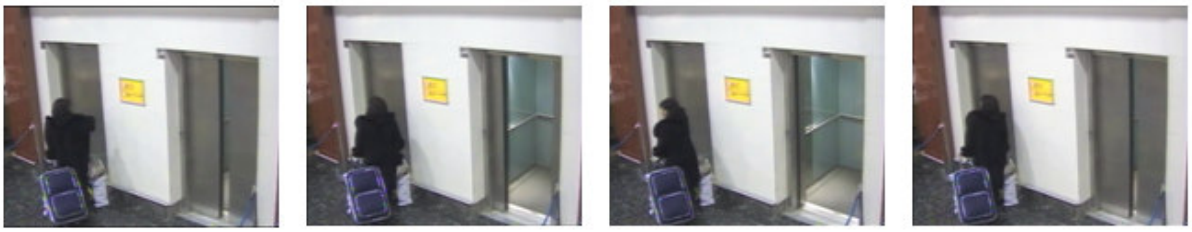


Figure12. Person waiting in front of the elevator and triggering the No-Entry alarm inLGW_20071206_E1_CAM4.

TABLE-1. Results of "Elevator No-Entry" event detection

Camera	Start Frame	End Frame	Confidence
LGW_20071123_E1_CAM3	91968	92253	0.68
LGW_20071123_E1_CAM4	91937	92323	0.71
LGW_20071130_E1_CAM3	15463	15761	0.62
LGW_20071130_E1_CAM4	15379	15754	0.68
LGW_20071206_E1_CAM3	26275	26513	0.38
	35502	35691	0.49
	85950	86220	0.44
	184996	185287	0.29
LGW_20071206_E1_CAM4	184996	185277	0.55

4.2 Opposing Flow

Table 2 shows the results of our *Opposing Flow* event detection. Figures 13 and 14 show few frames for the typical Opposing Flow true events detected by our algorithm. Our algorithm detects most of the true events. However, it misses true events in the cases when the person involved in the opposing flow is mostly occluded by other people. We also have few false alarms due to tracking errors caused by swapping of object IDs.

Figure13. Example opposing flow event in LGW_20071130_E1_CAM1.

Figure14. Example opposing flow event in LGW_20071207_E1_CAM1.

TABLE-2. Results of "Opposing Flow" event detection

Camera	Start Frame	End Frame	Confidence
LGW_20071123_E1_CAM1	12678	12724	0.78
	44366	44377	0.73
	44385	44398	0.8
	52706	52743	0.92
	66177	66198	0.77
LGW_20071130_E1_CAM1	105691	105731	0.95
	136923	136938	0.91
	160455	160483	0.46
	173455	173491	0.81
LGW_20071130_E2_CAM1	58143	58160	0.94
	70917	70927	0.75
LGW_20071206_E1_CAM1	6109	6134	0.93
	58679	58697	0.92
	60248	60287	0.88
	61276	61305	0.21
	84223	84242	0.85
	126962	126974	0.81
LGW_20071207_E1_CAM1	34223	34255	0.92
	35945	35961	0.96
	51661	51674	0.88
	91661	91690	0.98

4.3 Picture Take

No take picture event was detected in camera views corresponding to camera1 and camera 4 and those cameras not listed in the table. Our results show that the algorithm detects most of the true flash events. However, as mentioned before, the algorithm relies on flash for detection of take picture event so at present it cannot detect flash events which are taken without a flash. We believe that detecting camera or human hands at such zoomed out view is a difficult task. Table 3 shows the result of our *Picture Take* event detection algorithm. An example picture take with flash event is shown in Figure 15.
The algorithm does produce a few false positives. These happen when the flash event occurs outside the camera view. Reflection of flash light on shiny surfaces is falsely detected as a bright spot producing false positives. This drawback can be eliminated by using skin pixel detection to find if the region contains skin color pixels (hand). This will ensure that reflection from shiny surfaces does not cause false alarm.

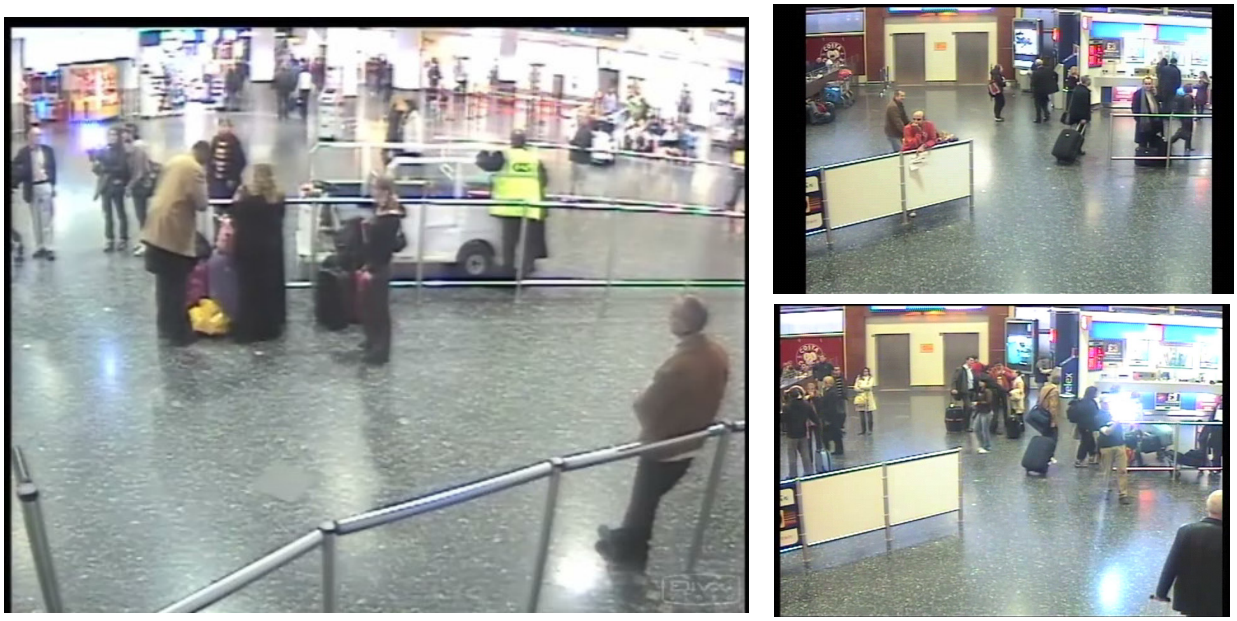


Figure15. Examples Picture Take events in camera 5 LGW_20071130_E2_CAM5 (left) and in camera 3 LGW_20071130_E1_CAM3 (right, top and bottom).

TABLE-3. Results of "Picture Take" event detection.

Camera	Start Frame	End Frame	Confidence
LGW_20071206_E1_CAM2	46689	46753	0.75
	61801	61845	0.67
	84601	84673	0.98
	84803	84881	0.95
LGW_20071207_E1_CAM2	131333	131405	0.87
LGW_20071130_E1_CAM3	105985	106076	0.97
LGW_20071206_E1_CAM3	45540	45597	0.71
	61765	61870	0.95
LGW_20071130_E2_CAM5	114231	114317	0.97

5 Concluding Remarks

We developed the algorithms for three event detection tasks, namely "Elevator No-Entry", "Opposing Flow", and "Picture Take". For *Elevator No-Entry* event detection, we used Haar based pedestrian detection followed by histogram matching to find person not entering an elevator. For *Opposing flow,* we used the top portion of a door region as the region of interest to minimize occlusions and overlaps caused by other people. We then used a robust direction estimator to detect opposing flow in this region of interest. In *Picture Take Event,* we relied on characteristic change in illumination caused by flash and simple template matching for hand motion detection.

In our experiments we found the main hurdle in event detection was lack of a stable pedestrian tracker in crowded scenes such as the airport data used for this evaluation. Even though our system is designed for outdoor surveillance and was not intended to handle very crowded environments, it performed quite well under such conditions. Participating in the TRECVID event, helped us improve on several algorithms such as pedestrian detection in crowded scenarios, robust direction computation with limited length tracks, skin color detection and flash detection. We are planning to transfer and implement some of the specific algorithms into our existing real-time video monitoring product.

References

[Bar05] A.L.C. Barczak, F. Dadgostar, and M.J. Johnson. "Real-Time Hand Tracking using the Viola and Jones Method", *Signal and Image Processing*, 2005

[Bil07] Bileschi, S. Wolf, L. "Image representations beyond histograms of gradients: The role of Gestalt descriptors", CVPR 2007

[Dal05] N. Dalal and B. Triggs. "Histograms of oriented gradients for human detection", *CVPR*, 2005

[Int07] N. intuVision, Inc.. "Panoptes: the all seeing, User's Brief", *Copyright 2007*, intuVision, Inc.

[Mon06] G. Monteiro, P. Peixoto, & U. Nunes, "Vision-Based Pedestrian Detection Using Haar-Like Features", *Robotica,* 2006.

[Kru03] H. Kruppa, M. Castrillon, S.B. Schiele, "Fast and Robust Face Finding via Local Context", *Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2003.

[Lie02] R Lienhart and J. Maydt. "An Extended Set of Haar-like Features for Rapid Object Detection". *IC IP* 2002.

[Ser05] T. Serre, L. Wolf, and T. Poggio. "Object recognition with features inspired by visual cortex", *CVPR*, 2005.

[Sme06] Smeaton, A. F., Over, P., and Kraaij, W. 2006. Evaluation campaigns and TRECVid. In Proceedings of MIR '06.

[Vio01] P. Viola and M. Jones. "Rapid object detection using a boosted cascade of simple features", *CVPR*, 2001.

[Vio03] P. Viola, M. J. Jones, D. Snow: "Detecting Pedestrians Using Patterns of Motion and Appearance". ICCV 2003.