Machine Vision and Applications (2014) 25:133–143
DOI 10.1007/s00138-013-0516-y

SPECIAL ISSUE PAPER

# Hierarchical abnormal event detection by real time and semi-real time multi-tasking video surveillance system

**Sung Chun Lee · Ram Nevatia**

Received: 24 October 2012 / Accepted: 22 April 2013 / Published online: 29 May 2013


**Abstract** In this paper, we describe how to detect abnormal human activities taking place in an outdoor surveillance environment. Human tracks are provided in real time by the baseline video surveillance system. Given trajectory information, the event analysis module will attempt to determine whether or not a suspicious activity is currently being observed. However, due to real-time processing constrains, there might be false alarms generated by video image noise or non-human objects. It requires further intensive examination to filter out false event detections which can be processed in an off-line fashion. We propose a hierarchical abnormal event detection system that takes care of real time and semi-real time as multi-tasking. In low level task, a trajectory-based method processes trajectory data and detects abnormal events in real time. In high level task, an intensive video analysis algorithm checks whether the detected abnormal event is triggered by actual humans or not.

**Keywords** Video surveillance system · Real-time abnormal event detection · Human trajectory analysis

## 1 Introduction

Video surveillance system has been used for various applications, such as traffic monitoring, security system, post incident analysis, etc. Originally these video surveillance systems were designed for human operator to watch concurrently or to record video data as archive for later analysis. Watching surveillance video is a labor-intensive task, as a large number of cameras need to be deployed and monitored; furthermore, it is a rather tedious task so it is easy for human observers to loose attention. Recently, computer vision researches have been heavily involved in intelligent video analysis applications. Automation can help overcome both cost and performance issues and free security personnel from routine tasks so that they can focus on higher-level cognitive tasks that better utilize their abilities.

The goal of the proposed work is to detect and provide warnings about the occurrence of undesirable or suspicious events. It requires some models of the possible human activities as well as some model of what is normal in an examining area. One of the major requirements for the proposed system should be the capability of catching incoming streaming track data without loss and analyzing track data to check abnormal events without stopping. At the same time, finding whether the detected event is aberrant, or not, requires further analysis. In simple cases, this is indicated merely by the presence of a person, vehicle or another object in locations where they are not normally found. Such processing is computationally demanding and does not run in real-time on state-of-art desktop PC level computers, thus it would be used sparingly in an overall system, in an off-line mode.

Recently, spatio-temporal primitive features, such as STIP (Spatial Temporal Interest Points) have been often used for activity recognition [2,4,6,7,9]. Statistics of primitive features are collected and classified using various machine learning classification techniques. But, they are usually suitable for segmented videos with known action boundaries. In other words, it is hard to handle unsegmented live streaming surveillance video data.

Another group of human activity recognition research has explored towards the estimation of human pose [1,10,12]. But, pose estimation requires expensive computation and it

S. C. Lee (✉) · R. Nevatia
Institute for Robotics and Intelligent Systems,
University of Southern California, Los Angeles, USA
e-mail: SungChun.Lee@usc.edu

R. Nevatia
e-mail: nevatia@usc.edu

may not be suitable for the real-time video analysis application. In addition, it becomes very challenging problem under inter- or intra- occlusions.

In this paper, we propose a hierarchical abnormal event detection system that takes care of real-time and semi-real time event analysis tasks. In low level task, we implemented a trajectory based abnormal event detection method that can process in real time. In high level task (semi-real time process), we request 'Video On Demand' data for the time of before-and-after the detected abnormal event from the video storage server. On receiving the requested video data, we apply an intensive video analysis algorithm to confirm whether the detected abnormal event is triggered by actual humans or not. The activities to be recognized may be belonging to a certain object (such as unexpected non-entry zone appearance), between two or more actors (such as collision, line formation).

## 2 Definitions

In this section, we explain a few important terminology definitions which will be often used in this paper.

### 2.1 Abnormal event definitions

We define '*Event*' to be something that happens and to be abnormal or normal. The purpose of our system is mainly the detection of pre-defined abnormal events. The scenarios of the target abnormal events for the proposed system are as follows:

1. *Illegal entry*: Someone walks in through the pre-defined illegal entry zone.
2. *Person fell down after collision*: Two people walk toward and collide with each other. One person stumbles and falls down to the ground. The stumbled person lies down for a while.
3. *Line formation*: A group of people suddenly stop and form a line in middle of the pathway. It indicates the unusual event of pedestrian traffic flow in the pathway.

### 2.2 Track, trajectory, and event object

We define '*Track*' as human location information at each frame. Track contains *x*-*y* position in world ground plane, time stamp, and track ID. In other words, it captures the instant location of human at each frame. We also define '*Trajectory*' as the list of tracks with the same ID from its beginning to end. Trajectory represents spatial and temporal history of human's moving.

After analyzing the given trajectories and tracks, we create event information which is defined as '*Event object*'. Event object has more information than the event itself. The highlighted attributes of the event object are as follows:

- *Event type*: Event types of abnormal events or undecided event as 'unknown'.
- *Start and end time*: Start (birth) and end (termination) of the event
- *Event location*: Location of the event.
- *Trajectory list*: List of associated trajectories from its birth to the current time.

Event object may represent an abnormal event or not. When it is undecided event, it simply implies human's trajectory. We refer 'event object' without specifying 'abnormal' as undecided event (i.e., human's trajectory) in this paper. The event location attribute is used and updated differently according to the event type. For example, illegal entry saves one point location and requires continuous location update so that we can track down the intruder after he or she makes illegal entry. In case of line formation, it requires two location points and does not require location update.

## 3 System overview

We assume that human trajectory is provided in real-time by a baseline surveillance system [5,11]. Our goal is to detect abnormal events by processing the real-time track data from the baseline system. At the same time, we also want to improve the system reliability by reducing the number of false alarms (which is common problem in video surveillance system). For this purpose, we apply computationally expensive object detection algorithm to verify the initially detected events. In order to integrate between real-time baseline surveillance system and non-real-time proposed system, we implemented our system as a hierarchical level system that allows concurrent real-time and non-real-time operations as shown in Fig. 1.

### 3.1 Baseline human tracking system

We use a baseline video surveillance system that has capability of multi-camera human tracking system [5,11]. An initial one-time calibration of all cameras is required during system installation. The foreground / background modeling is continuously performed for human detection algorithm. Once the foreground map has been computed, the human detection approach examines whether foreground patches using geometric shapes that have size and shape similar to people. Human detection responses from all cameras are projected onto the ground plane at each frame. A centralized tracker collects the time-ordered

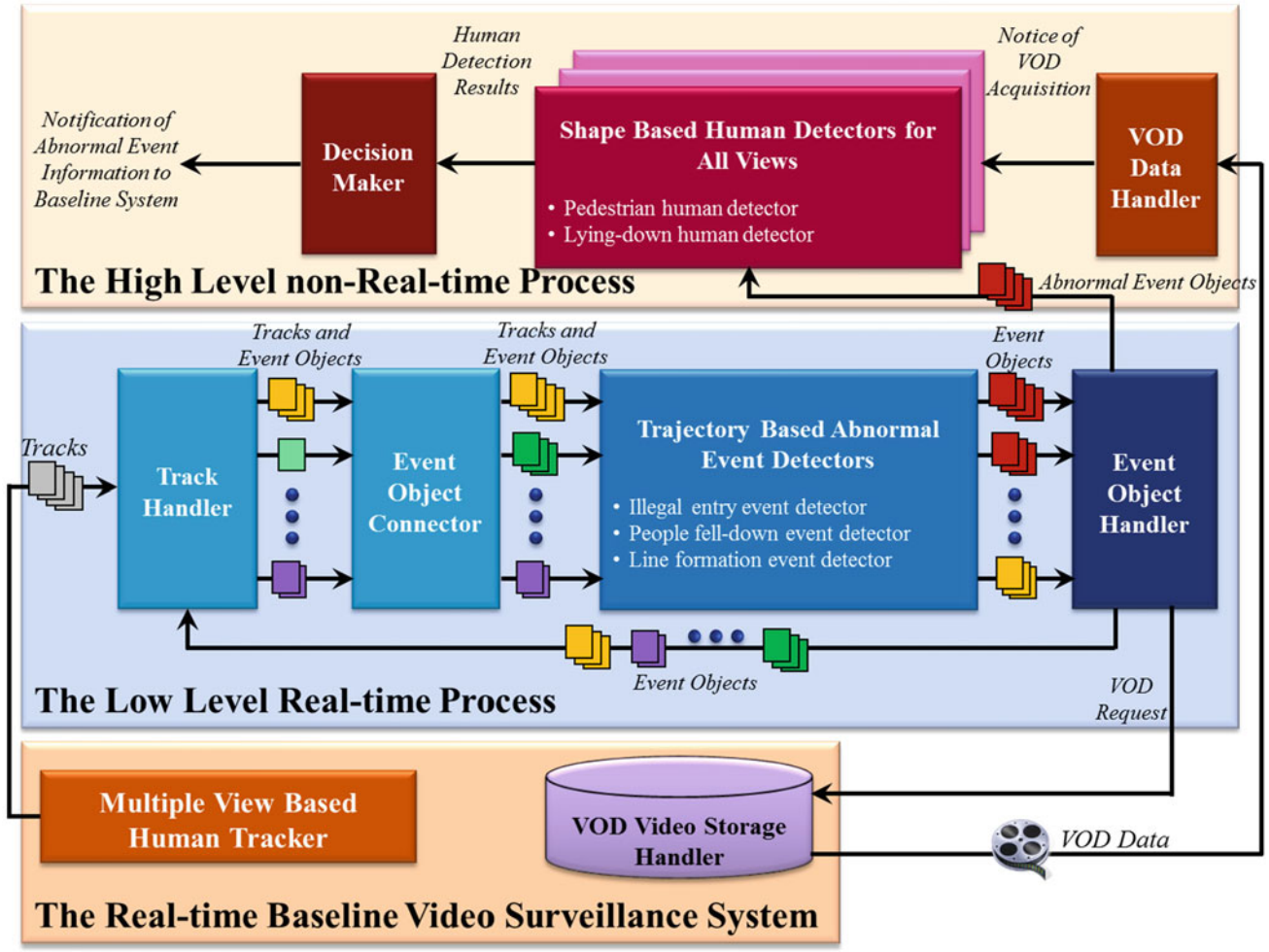



**Fig. 1** The overview of the proposed system and the baseline video surveillance system

detections and forms detection response into tracks. However, due to real-time processing constrains, there might be false alarms generated by video image noise or non-human objects. In addition, there might be missing humans caused by lack of motion foreground data (e.g., standing still).

### 3.2 The proposed system

We implemented two different processes inside the proposed system: *The low level real-time process* and *the high level non-real-time process*. *The low level process* is tightly coupled with the real-time baseline surveillance system and *the high level process* is loosely coupled. The processes are programmed as multi-threads to support a heterogeneous coupling system. Each process has subcomponent modules as shown in Fig. 1.

*Track handler* receives all incoming tracks from the baseline system at every frame and checks if the received track is associated with the already existed event objects as will be described in Sect. 4.1. *Event object connector* checks if multiple event objects are related and merge them to remove redundancy (in Sect. 4.2).

There are three *Trajectory based abnormal event detectors* that correspond to each target abnormal event (in Sect. 4.3). When the system detects any abnormal event, *event object handler* sends a VOD (Video-On-Demand) data request for time of before-and-after the detected suspicious event (in Sect. 4.4).

When *VOD data handler* in the high level process confirms the acknowledgment of receiving the requested VOD data, it calls *Shape-based human detectors* for all available views to verify the initially detected abnormal events (in Sect. 5.1). *Decision maker* collects all detection results and makes a decision whether the detected event is triggered by actual human and notifies the confirmed 'Abnormal Event' to the baseline system (in Sect. 5.2).

Experimental results and conclusion will be presented in Sect. 6.

## 4 Low level process

The baseline surveillance system has a capability of the scene calibration to extract the world ground coordinate for each track [13]. The baseline system sends a list of tracks at each frame if any. The low level process handles incoming tracks, detects an initial abnormal event, and notifies to the high level process for verification.

### 4.1 Track handler

*Track handler* collects the tracks from the baseline system and associates them with the already existed event objects. The association at this stage is simply checking whether they have the same track ID or not. If they are associated, we update the associated event object with the current track information. Single track can be associated with multiple event objects. For example, a track can associate with illegal entry as well as line formation events. In this case, we add the track to all associated events.

If a track does not find its associated event, then we create new event object for the unassociated track. Please note that new event object is undecided regarding being normal or abnormal at this stage. Once an event object is created, it should be continuously updated by incoming tracks. If it fails to be updated due to track loss from the baseline system, it becomes 'Missing event object' which will be handled by *Event connector* in the next Section.

### 4.2 Event object connector

Multiple event objects may represent the same actual event. For example, when two line formation events are close each other, they should be combined as one event. Upon merging events, trajectories with the same track ID in both events are also merged to become one trajectory.

In addition, we also connect a missing event object to one of the other updated event objects. There are two possible cases in missing event object. The first case is human's exit, i.e., human leaves the scene coverage of the camera. The second case is caused by track loss. Only the second case is required to connect to another existed event object. Assuming that the camera is static, we exploit scene knowledge regarding entrance and exit of the scene, such as at the left/right, the top/bottom fringes of images, or the pre-defined doorways in the middle of image. Under this assumption, when an event object disappears in non-exit areas, we consider the event object to be missed, not to be exited. In this case, we search for any nearby event objects. Especially, if the nearby event object is just created at its proximity, it is more likely for the missed event object to be reappeared. We connect the missed and the nearby event objects.

### 4.3 Trajectory-based abnormal event detectors

Given the list of trajectories in an event object, the system makes decisions regarding whether suspicious event is taking place or not. The system examines the trajectory not only in spatial relationship, but it also considers the temporal constraints, such as the duration of abnormal event. In this section, we describe how to detect different types of abnormal events using the proposed trajectory analysis method. Please note that the following three abnormal detectors are simultaneously running and single event object can contribute to multiple abnormal events.

#### *4.3.1 Line formation event detection*

People usually walk through the pathway. When the pedestrian traffic stops and forms a line, it indicates that an abnormal situation is happening. The system should notify to the human operator (security personnel) to pay more attention. There could be many false line formations when a group of people walk together or people walk separately, but accidentally form a line. It suggests that we should take into consideration not only the spatial relations, but also the temporal constraints to detect abnormal line formation event reliably.

When it comes to spatial inference, at each frame we form a line hypothesis by choosing three tracks such that the distance among each other is less than a distance threshold ($\theta_{close}$). We check any other human tracks by estimating the perpendicular distance error to the hypothesized line. When the distance error is less than another threshold ($\theta_{dist}$), we create an event object whose type is initial line formation. Here, the initial line formation event object is not abnormal event yet, since it needs more evidence. We add the human tracks that form a line, to the created event object. We control the spatial proximity constraint by using two distance thresholds $\theta_{close}$ and $\theta_{dist}$. The threshold depends on the domain knowledge of defining the abnormal line formation. For example, it may become higher value when the target site covers larger area and vice versa. The proposed system allows the domain user to provide these values. We will show the actual parameter settings for our evaluation sites in Sect. 6.

To handle the temporal constraint, we derived a voting method. Once an initial line formation event object is created, we accumulate votes from any other tracks that meet the spatial constraint to support the line formation. As time goes on, the line formation event object gets more votes when it finds more people coming closer and meet *spatial constraint* as the line color gets darker in Fig. 2.

Here, we introduce another threshold, $\theta_{vote}$ to address the issue of temporally stationary line. If the accumulated vote is higher than $\theta_{vote}$, it implies a group of people stay at the same location for a significant amount of time which indicates a real abnormal line formation situation. Due to this reason,

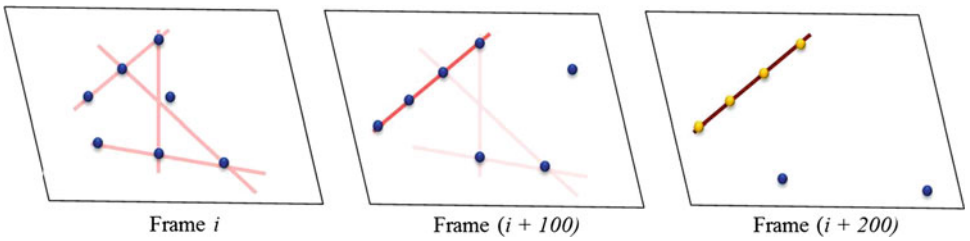



**Fig. 2** Illustration of line formation detection in spatial and temporal inference

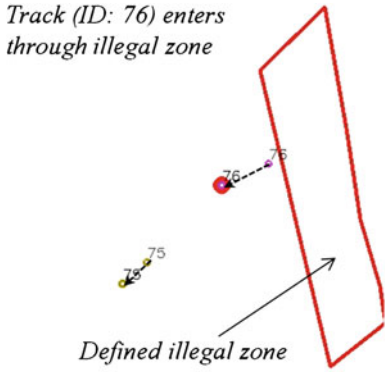



**Fig. 3** Initial illegal entry event detection

we do not update the location of line formation event object. On the other hand, when a group of people walk together, it is possible to create a line formation event object, but the line location of the event object never gets meaningful votes to become abnormal event because people move away from it. When there is an initial abnormal event detected, the system notifies to the high level module for further investigation.

The procedure of the proposed trajectory based initial line formation event detection is described in Algorithm 1. The computation time is $O(n^3)$, where $n$ is the number of human tracks at each frame. But please note that $n$ is usually small number (<10) in real video surveillance environment.

### *4.3.2 Illegal entry event detection*

Common people walk in and out through the permissible entrance or exit. If someone enters from a restricted (or illegal) area, it is an abnormal event situation.

First the user defines the illegal entry zones (list of polygons in the world ground coordinates as shown in Fig. 3). The system detects any event objects appearing from the predefined illegal entry zones. If the detected event object enters through the illegal entry zone, it becomes an abnormal event object and the system notifies to the high level module for further investigation.

It can be anything that triggers the illegal entry event at the low level. For example, some non-human objects flown by

**Algorithm 1** Line formation detection

**Require:** List of tracks at the current time: $T$,
list of event objects: $EObjects$
**Ensure:** Detected abnormal event object
$AbnormalEventList$ = Location: $L$, Trajectory: $Trj$, Time: $D_t$, Event type: $E_t$

```
if SizeOf(T) < 3 then
    return null
end if

for t1 ∈ T do
    for t2 ∈ T do
        for t3 ∈ T do
            if t1! = t2 and t2! = t3 and t1! = t3 then
                dist_1 = Distance(t1, t3);
                dist_2 = Distance(t2, t3);
                if dist_1 < θ_close and dist_2 < θ_close then
                    pDist = PerpDist(t1, t2, t3);
                    if pDist < θ_dist then
                        Create LineEvent(L, D_t, t1, t2, t3)
                        Insert LineEvent to EObjects
                    end if
                end if
            end if
        end for
    end for
end for

for EO ∈ EObjects do
    if EO.E_t == lineformation then
        for t ∈ T do
            dist_1 = Distance(t, EO.L.p1);
            dist_2 = Distance(t, EO.L.p2);
            if dist_1 < θ_close and dist_2 < θ_close then
                pDist = PerpDist(t, EO.L.p1, EO.L.p2);
                if pDist < θ_dist then
                    Cast a vote for EO;
                end if
            end if
        end for
    end if
end for

for EO ∈ EObjects do
    if EO.E_t == lineformation then
        if VotingScore(EO) > θ_vote then
            EO.E_t = abnormal line formation;
            Insert EO to AbnormalEventList;
        end if
    end if
end for
```

wind or jumping animals can activate the tracks by the real-time motion-based video surveillance system. This kind of ghost track may trigger initial event detection (false alarm). We need to verify the detected initial event by running intensive human detection (high level) based method which will be described in Sect. 5.

#### 4.3.3 Person fell-down event detection

It is not easy to detect people's collision and falling down using only trajectory information, because it additionally requires human's pose. However, it is not feasible to use such computationally expensive algorithm as pose estimation method in low level due to real time constraint. Instead, we first guess the collision detection when two separate tracks meet at one location.

As we mentioned in Sect. 3.1, the baseline surveillance system generates the tracks using object motion. When a person falls down and stays still, it is possible that his or her track may be lost. Using this characteristic of the baseline system, we can guess the initial collision situation. In other words, when two separate event objects meet at one location, we can roughly detect a collision event by checking any of these event objects is disappeared after the meeting.

First we need to detect a meeting event to proceed further people collision detection. Given trajectory in the event object, we used a rule-based meeting detection method by using following necessary conditions:

(1) *Encountering condition*
Pedestrians should be away from each other before they meet. A distance between pedestrians when they enter the scene (i.e., at the beginning of each trajectory) should be larger than $\eta_{close}$. Otherwise, they have already met each other, which is not a meeting event.
(2) *Closeness condition*
Pedestrians should stay close to each other during the meeting. The distance between pedestrians should be less than $\eta_{close}$, a closeness distance threshold.

Once a meeting event object is detected, the system keeps monitoring two trajectories in the event object to check whether one of them disappears or not. In some cases, a trajectory may disappear and reappear as new track due to occlusion at the meeting spot. In addition, even if a person falls down after collision, he or she may stand up and walk again which should not be an abnormal situation. To distinguish abnormal versus normal events, we need to use a temporal inference. If there is no new track appearing at the spot of the meeting (i.e., possible collision spot) in long time, $\eta_{elapse}$, the meeting event object becomes an initial abnormal event object.

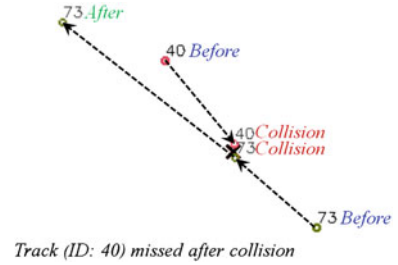



**Fig. 4** Initial person fallen down after people collision event detection

As illustrated in Fig. 4, the event object with track ID '40' is missed after colliding with another event object (ID: 73). If there is no new track showing up at the collision spot, the system creates the initial abnormal event object and notifies to the high level process for more accurate verification.

The procedure of the proposed trajectory-based initial people collision event detection is described in Algorithm 2. Ghost and missed tracks may trigger initial event detection (false alarm). We need to verify the detected initial detection by running intensive human detection (high level) method. This time, we use 'lying-down' human detector to confirm that someone is fallen down after the collision.

### 4.4 Event object handler

Once the system detects any abnormal event from the low level process, *Event object handler* sends a VOD (Video-On-Demand) data request for time of before-and-after the suspicious event to the baseline system which stores all archive video data. It also notifies to the high level process by sending the initially detected abnormal event object information.

In addition, event object handler updates the elapsed time for all event objects. The event elapse time is used for abnormal event's temporal constraint as described in Sect. 4.3. We remove the undecided events whose elapsed time is very long (e.g., 1 h), which have high chance of being false events. Even though they are not harmful to the system's precision in regards to abnormal event detection, they should be removed to save system's memory.

## 5 High level process

In this Section, we describe about sub-component modules in the high level process, especially focusing on *Shape-based human detection* and *Decision maker*.

### 5.1 Shape-based human pose detection approach

Ghost tracks may trigger initial event detection (false alarm) which requires further verification process at high level module. The problem of verifying unusual events (involving

**Algorithm 2** People collision detection

**Require:** List of tracks at the current time: $T$,
list of event objects: $EObjects$
**Ensure:** Detected abnormal event object
$AbnormalEventList$ = Location: $L$, Trajectory: $Trj$, Time: $D_t$, Event type: $E_t$

```
if SizeOf(T) < 2 then
    return null
end if

for t1 ∈ T do
    for t2 ∈ T do
        if t1! = t2 then
            Check c1 = ClosenessCondition(t1, t2);
            Check c2 = EncounteringCondition(t1, t2);
            if c1 and c2 then
                Create MeetEvent(L, D_t, t1, t2);
                Insert MeetEvent to EObjects;
            end if
        end if
    end for
end for

for EO ∈ EObjects do
    if EO.E_t == meetingevent then
        if EO.D_t < (currentTime − η_elapse) then
            Check n = AnyNewTrack(T, L);
            if n then
                Passing-By Case
            else
                Insert (L, D_t, t1, t2) to initial event list
            end if
        end if
    end if
end for
```

humans) is posed as a problem of detecting humans. Observed images from video stream depend on viewpoint, illumination conditions, clothing and other variables which makes the human detection hard problem. The more complex abnormal events require detection of humans not only in upright poses (e.g., standing or walking), but also in various other poses, such as lying down, etc.

To this effect, we train the cluster boosted tree object detector [14] which is capable of automatically partitioning the positive sample space whenever the complexity of the current classifier becomes high. Initially, a detector based on boosting of edgelet features [14] is used to train a standing pose human (or pedestrian) detector. For multi-pose human detection, it is required to train new classifier for our application. We collected a dataset of humans in varying poses from multiple cameras (multiple viewpoints) as shown in Fig. 5. The data was divided into two sets, the training set and the test set. For training the detector, we need to extract the normalized image that contains only human from the original image. We used a background subtraction method [8] on the training videos to extract the moving human blob, so that we can extract the bounding box of the foreground to obtain the training samples. The samples were normalized to the same orientation. The detector was trained for a false alarm rate of *2e-6*. The extracted training data examples are shown in Fig. 5b.

We also try to the classifier with another feature called, Joint Ranking of Granules (JRoG) features for training and classification [3]. A JRoG feature unites binary ranking results made by several granule pairs which are selected among thousands of gray-level granules in the granular space. Authors proposed a collaborative learning algorithm enhanced by a simulated annealing (SA) step in combination with a Real AdaBoost algorithm and demonstrated the advantage of the new method.

While the detection rate of the edgelet feature detector did not rise above 75 %, the detection rate of JRoG feature detector for lying-down pose was 90 % with a false alarm rate of 18 % per frame. Based on the results of our experiments, we have decided to use the JRoG feature-based classifier [3] for multi-pose human detection. Figure 6 shows examples of human detection results in standing and lying-down poses.

### 5.2 Decision maker

Because all cameras are calibrated, we can filter out unnecessary cameras whose field-of-view does not cover the suspicious event location. Upon finishing shape-based human detector for all covered views, *Decision maker* collects all detection results to check if there is enough evidence of human's involvement for the initially detected abnormal event.

To compute decision threshold value, we need to compute the expected number of human detections. The video-on-demand has a rate of 15 frames per second and the length of the requested video is different according to the type of abnormal event. For example, we need very short time (e.g., 1 s) for the illegal entry event, while people fell-down event requires longer (e.g., 10 s) length of video. The total expected number of human detections, $N_{detection}$ is derived by following equation:

$$N_{detection} = 15 \times N_{length} \times N_{views} \times N_{human}$$

where $N_{length}$ is the length of video in second, $N_{views}$ is the number of all available camera views, and $N_{human}$ is the number of trajectories in the event object.

Since, our human detector has 80 % of precision rate as we explained in Sect. 5.1, we set the threshold for decision making to be 80 % of $N_{detection}$. Once it confirms an abnormal event, it notifies the confirmed 'Abnormal Event' to the baseline system.

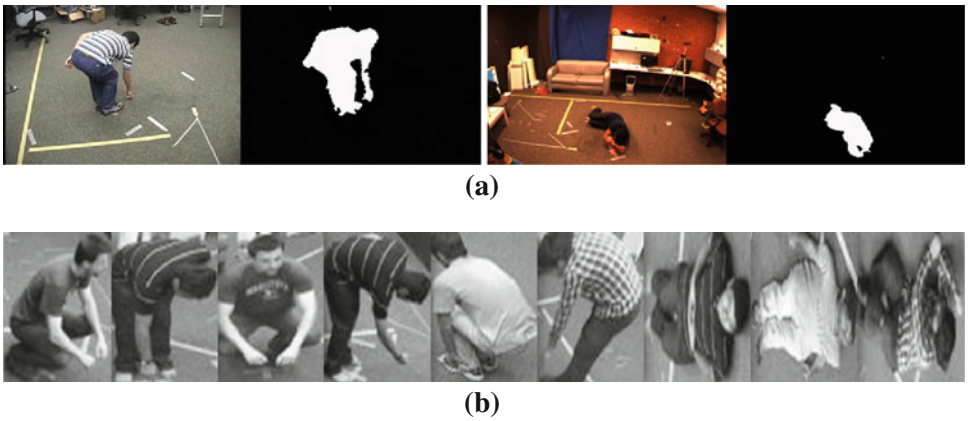


**Fig. 5** Training data collection for multiple pose human detection. **a** Original images and background subtraction results. **b** Collected training data

**Fig. 6** Examples of human detection in different poses (*left*: standing and walking, *middle* and *right*: lying down)

## 6 Implementation and evaluations

### 6.1 Domain parameters

As mentioned in previous Sections, the proposed system requires the domain-specific parameter settings. The parameter setting for our experiments is shown in Table 1.

In our experiments, without consulting the domain expert these parameters are chosen based on our experiments and used for all evaluations. When it comes to determine the domain-specific parameter, one important consideration is inaccuracy of trajectory data from the baseline surveillance system. Due to this reason, we set the distance related parameters ($\omega_{assoc}$, $\theta_{close}$, $\theta_{dist}$, $\eta_{close}$) as twice as the real distance parameters should be. For example, $\theta_{close}$ represents the distance between humans inside line formation and $\theta_{dist}$ is used for line fitting error. Ideally both parameters should be less than 1 m, but we increase as 2 m due to trajectory noise. $\omega_{assoc}$ is determined as the same way as 3–4 m search boundary in ideal situation becomes 7 m.

It is important to consider temporal constraint due to sensitivity in spatial constraints. The parameter, $\theta_{vote}$ in Table 1 indicates the length of time in counting the number of frames and it can be different based on the performance of the baseline surveillance system. For instance, if the baseline video surveillance system processes video data in approximately 10 frames per second, '*200*' means 20 s. Please note that the baseline system's performance rate (10 frames per second) and the archived video frame rate (15 frames per second) for VOD data are different. $\eta_{elapse}$ implies the amount of time that the human lies down after the collision. Under real environment, it could be 5–10 min. But, we simulate a brief version of people collision event as short as 10 s.

### 6.2 Evaluations on two sites

As we mentioned in Sect. 3, our system heavily coupled with the baseline surveillance system. For example, the low level process requires human tracks as input and the high level process needs VOD data of the initially detected abnormal

**Table 1** Domain-specific parameter description and setting

| Param. | Description | Setting |
|---|---|---|
| $\omega_{assoc}$ | Nearby distance threshold for track association | 7 (m) |
| $\theta_{close}$ | Closeness threshold for line formation | 2 (m) |
| $\theta_{dist}$ | Distance threshold for line formation | 2 (m) |
| $\theta_{vote}$ | Voting threshold for line formation | 200 (frame) |
| $\eta_{close}$ | Distance threshold for people collision | 1 (m) |
| $\eta_{elapse}$ | Elapse time for people collision | 10 (s) |

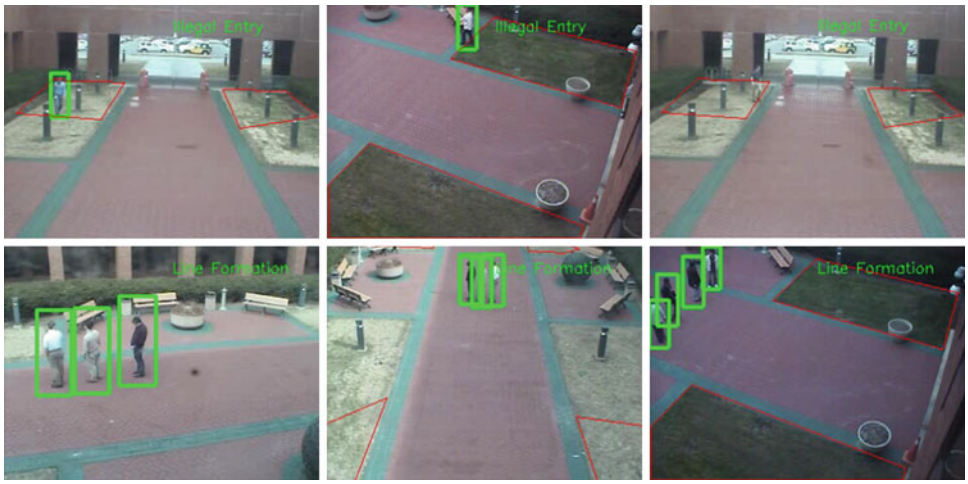



**Fig. 7** Event detection results (illegal entry event detection (*top row*), line formation event detection (*bottom row*)

events. Due to this reason, it is not easy for us to evaluate our system using public dataset. In addition, it is hard to find the exact same abnormal events as our targets from the publicly available dataset. Instead, we installed our system in two complex industrial sites for the development and testing purposes of the system.

We tested the proposed system in two sets of pre-recorded videos from both industrial sites; (1) building court yard and (2) outdoor street surveillance cameras. Even though they are pre-recorded, we take them as live streaming videos in real-time fashion. All abnormal events are staged for the evaluation purpose.

We tested '*illegal entry*' and '*line formation*' abnormal event detection from the building court yard surveillance camera which requires only standing human pose detection. There are 12 short video clips that contains abnormal activities, six for line formation and six for illegal entry zone. Each video takes 2–3 min and contains at least one or more abnormal events. Overall, there are 15 abnormal events in total as ground truth data. As a result, we were able to detect all 15 abnormal events without any false alarm in the low level analysis.

In order to test high level human detection, we picked one frame of the detected abnormal event from each VOD sequence and applied the human detection algorithm. There are 35 actual humans (ground truth) in 15 selected frames; our method finds 32 of them without any false alarms. The recall and precision rates of the human detection method are as follows:

1. *Recall (# of correct detection / # of ground truth) : 0.91*
2. *Precision (# of correct detection / # of total detection) : 1.0*

Some examples of event detection with the detected humans are shown in Fig. 7. The first column shows examples of illegal entry abnormal event detection. The pre-defined illegal entry zones are overlaid as red color polygons in Fig. 7. As shown the last image of the first row in Fig. 7, the system missed a human coming through illegal zone. However, it is hard even for human to separate this intruder from the background due to less contrast of his cloth. The bottom row shows the abnormal line formation event detection results.

In the second evaluation site (outdoor street surveillance camera), we tested three abnormal events; *illegal entry*, *line formation*, and *people collision*. This time we tested out system for longer system running time to test system's durability. For example, each video takes about 15 min of running time. We experimented each abnormal event separately. While we are running our system, actors repeatedly enter and exit to the covering stage and demonstrated abnormal behavior. Please note that our system tries to detect for all abnormal events even though the video contains only single abnormal event. For instance, the system may detect line formation event in the illegal entry video. Overall, there are 25 illegal entry events, 10 line formations, and 13 people fell-down events during totally 45 min of videos.

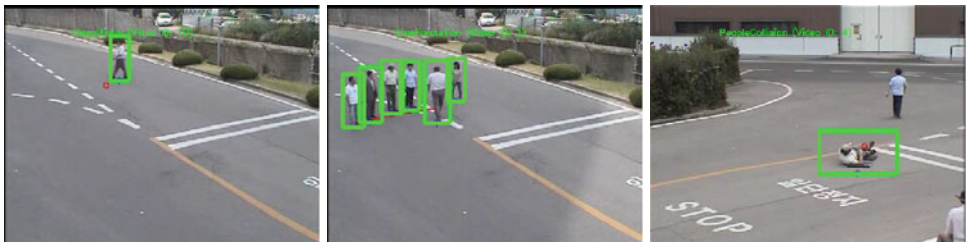
Table 2 shows the quantitative results of the proposed system. We divide the detection result into low and high levels, so that we can see the improvement of the hierarchical system. High level results are generated from the outcome of the low level results (SD + FA). For example, in case of illegal entry, the high level process takes 22 events (21 SD + 1 FA) from the low level and ends up 21 successful detection without false alarm. The missing cases are caused by not enough trajectory information for the low level process to analyze. We detected eight out of ten line formation events with two misses. People collision detection is very challenging problem. Out of 13 cases, we were able to detect nine correct ones. More graphical results are shown in Fig. 8. We also submitted video clips as supplementary material that was captured during real-time operation.

In order to test the system's reliability, we have experimented our system using the video clip that has no abnormal event. Here is a scenario: two people walk toward each other,

**Table 2** Quantitative results of abnormal event detection from outdoor street surveillance camera

| Abnormal events | GT | SD-L | MD-L | FA-L | SD-H | MD-H | FA-H | SD-F | MD-F | FA-F | Recall | Precision |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Illegal Entry | 25 | 21 | 4 | 1 | 21/22 | 0/22 | 0/22 | 21 | 4 | 0 | 0.84 | 1.0 |
| Line Formation | 10 | 9 | 1 | 0 | 8/9 | 1/9 | 0/9 | 8 | 2 | 0 | 0.8 | 1.0 |
| People Collision | 13 | 12 | 1 | 2 | 9/14 | 3/14 | 0/14 | 9 | 4 | 0 | 0.69 | 1.0 |

*GT* ground truth, *SD* succesful detection, *MD* missed dectection, *FA* false alarm, *L* low level result, *H* high level result, *F* final result




**(a)** **(b)** **(c)**

**Fig. 8** Examples of abnormal event detection results (*left* to *right*: illegal entry, line formation, people collision). **a** Illegal entry, **b** line formation, **c** people collision

but just pass by without collision. In this case, the low level process may detect them as people collision event. In our experiment, the low level process detects 3 false positives out of 17 normal passing-by cases as people collision events. False alarms are caused by noisy track data. For example, some tracks were lost after two people's meeting (i.e., occlusion). However, the high level process was able to verify all three false alarm cases by confirming no person's lying-down detection at the spot of the collision location.

Regarding computation time, the high level process takes the most computation time, since the low level process performs in real time with the baseline system. The overall average latency of the high level process is about 15 sec using Intel Xeon 12 cores CPU (with two processors) with 3.33 GHz clock speed.

### 6.3 Conclusion

As a conclusion, we presented a hierarchical abnormal event detection system that takes care of real-time streaming video surveillance data. The low level trajectory-based abnormal event detection analysis module can process real-time trajectory data and trigger the initial suspicious event detection in real time. The high level intensive video analysis module verifies whether the initially detected abnormal event is triggered by actual humans or not to reduce false alarms. We were able to evaluate the proposed system with various kinds of video surveillance data.

## References

1. Felzenszwalb, P.F., Huttenlocher, D.P.: Pictorial structures for object recognition. Int. J. Comput. Vis. **61**(1), 55–79 (2005). doi:10.1023/B:VISI.0000042934.15159.49
2. Gupta, A., Davis, L.: Objects in action: An approach for combining action understanding and object perception. In: IEEE Conference on Computer Vision and Pattern Recognition (2007)
3. Huang, C., Nevatia, R.: High performance object detection by collaborative learning of joint ranking of granules features. In: IEEE Conference on Computer Vision and, Pattern Recognition, pp. 41–48 (2010)
4. Junejo, I., Dexter, E., Laptev, I., Perez, P.: Cross-view action recognition from temporal self-similarities. In: European Conference on Computer Vision (2008)
5. Krahnstoever, N., Kelliher, T., Rittscher, J.: Obtaining pareto optimal performance of visual surveillance algorithms. In: IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS) (2005)
6. Laptev, I., Lindeberg, T.: Space-time interest points. In: International Conference on Computer Vision (2003)
7. Laptev, I., Marszalek, M., Schmid, C., Rozenfeld, B.: Learning realistic human actions from movies. In: IEEE Conference on Computer Vision and Pattern Recognition (2008)
8. Li, L., Huang, W., Gu, I.Y., Tian, Q.: Foreground object detection from videos containing complex background. In: Proceedings of the eleventh ACM international conference on Multimedia, pp. 2–10 (2003)
9. Niebles, J., Fei-Fei, L.: A hierarchical model of shape and appearance for human action classification. IEEE Conference on Computer Vision and, Pattern Recognition (2007)
10. Ramanan, D., Forsyth, D.A., Zisserman, A.: Tracking people by learning their appearance. IEEE Trans. Patt. Anal. Mach. Intell. **29**(1), 65–81 (2007). doi:10.1109/TPAMI.2007.22
11. Rittscher, L.G.J., Krahnstoever, N.: Multi-target tracking using hybrid particle filtering. In: IEEE Workshop on Applications of Computer Vision (WACV) (2005)
12. Singh, V.K., Nevatia, R., Huang, C.: Efficient inference with multiple heterogeneous part detectors for human pose estimation. In: Proceedings of the 11th European conference on computer vision conference on Computer vision: Part III, ECCV'10, pp. 314–327. Springer, Berlin, Heidelberg (2010)
13. Tu, P., Rittscher, J., Kelliher, T.: Site calibration for large indoor scenes. In: IEEE International Conference on Advanced Video and Signal-based Surveillance (AVSS), pp. 358–363 (2003)
14. Wu, B., Nevatia, R.: Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors. Int. J. Comput. Vis. **75**(2), 247–266 (2007)

## Author Biographies

**Sung Chun Lee** received the B.Eng. and M.Eng. degrees from the Department of Computer Engineering, Pusan National University, Korea, in 1995 and 1997, respectively, and the PhD degree from the Department of Computer Science, University of Southern California, in 2004. From 1997 to 1998, he was the Member of Technical Staff, the Korea Telecom Co., Korea. He has been a research associate at the Institute for Robotics and Intelligent Systems, Viterbi School of Engineering, at University of Southern California since 2004. His research interests include computer vision, video analysis, and geometry modeling. He is a member of the IEEE.

**Ram Nevatia** received the B.S. degree in electrical engineering from the University of Bombay, India, and the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Palo Alto, California. He has been with the University of Southern California, Los Angeles, since 1975, where he is currently a professor of computer science and electrical engineering and the director of the Institute for Robotics and Intelligent Systems. His research interests include computer vision, artificial intelligence, and robotics. He is the author of two books and has contributed chapters to several others. He is a fellow of the AAAI and the IEEE.