Active Surveillance in Public Environments

Timothy Sweet and Mircea Nicolescu
Department of Computer Science and Engineering, University of Nevada, Reno, Nevada, USA

Abstract—The growing interest in automated surveillance in the context of homeland security, civilian, and military applications is fueling demand for robust and efficient surveillance software. This capability is particularly desirable in public environments such as federal buildings or airports where human operators have to monitor a large number of camera feeds. Humans have been known to be unreliable and inconsistent in this multitask-oriented environment. In this paper, we present a framework for automatically detecting several types of interactions involving human agents and objects they carry. The proposed system uses a novel combination of vision-based techniques that allow real-time detection, as well as offline searching for instances of events with specific attributes in a large video. We validate the system on a set of sample video sequences exhibiting common events we expect to see in public environments.

Keywords: surveillance; computer vision; tracking

1. Introduction

Interest in automated surveillance has shown a significant growth as the government, military, or private companies seek to deploy robust vision-based surveillance systems in a wide range of environments [1], with applications that include behavior monitoring and theft identification, both in real-time and offline. Solutions have ranged from simple intruder detection [2] to complex tracking of abnormal behavior of individuals among crowds [3]. We propose a system which detects relevant activities/events in complex scenes such as federal buildings, airports, shopping malls, and other public places. By developing a method to reliably detect specific actions, security personnel no longer need to continuously watch video monitors and can focus on less monotonous, more important security tasks. The system receives video from one or many cameras and tracks people and objects over time, and can alert a human operator when a certain activity is identified.

Prior work in autonomous video surveillance systems vary largely in implementation, application, and sensor models [4] [5]. There are few examples of autonomous video surveillance systems deployed in real settings. Currently, security personnel in a sensitive public facility such as a federal building or airport use a large number of monitors to manually scan the live video for suspicious events [1] [4]. This repetitive, monotonous, and multitask-oriented environment is not well suited for human attention [6]. Basic identification of relevant events involving human agents and objects they carry can instead be tasked to an automated system, which can accomplish this task more quickly and sometimes more accurately [5], and free the security personnel to investigate suspicious activity when alerted by the system.

Possible applications of the proposed system fall into two general categories: live detection and offline review. Live detection corresponds to the typical job of security personnel: watching for any instance of suspicious activity as it unfolds. Offline review occurs when security or law enforcement personnel receive information about a past event, and wish to see surveillance footage from the event. For example, a witness to a crime may report that they saw a person with a black shirt and blue pants walking away with a brown bag. The task of searching in surveillance footage from a large array of cameras to find such a specific event is well suited for our system.

Our previous work [7] provided a high-level description of a modular system for analyzing human behavior in the context of security applications. This paper is focused on the specific vision-based capabilities for such a system and presents a novel combination of various methods into a practical, functional system for monitoring interactions involving humans and relevant objects. Our system is designed to be applicable to public surveillance scenes such as airports.

The rest of this paper is organized as follows. Section 2 presents previous work related to vision-based surveillance and activity recognition. Section 3 provides a technical description of the methods used in segmentation and tracking. Section 4 presents the approach used for event detection. Section 5 provides an experimental evaluation of the system. The paper is concluded in Section 6.

2. Related Work

Räty provided a survey of the state of the art in surveillance systems in [5]. This describes the “3rd generation” multisensory systems which combine information from multiple cameras to produce intelligent information. Common applications of multisensory surveillance systems include systems to detect intruders or track objects. Other relevant surveillance systems are described in [8] [9] [10] [11]. Haritaoglu et al. [12] demonstrated a system for detecting people by predicting positions of body parts. This led to the ability to track people in groups and across occlusions. Their work focuses exclusively on monochromatic, low resolution images, and relies on head detection for groups of people. This assumes a nearly horizontal view of the scene, and
Fig. 1: The process architecture of our system

is not suited well for security cameras which are generally mounted on or near ceilings such that they are looking down on a crowd of people.

Zhao et al. [13] presented a method for detecting and tracking multiple humans outdoors by explicitly providing the camera’s extrinsic parameters and time-of-day/weather information (for shadow removal). Their work shows promising results in human body detection but requires a non-trivial setup in physically calibrating the camera and environment to be observed, while also using non-vision information such as sun position and cloud coverage. This becomes prohibitively complex and time consuming with security cameras numbering in the thousands at large public scenes. When considering possible environment changes in a dynamic scene such as an airport, modeling a scene becomes impractical and a vision system designed for surveillance should be able to adapt automatically to these changes.

Related to the capabilities proposed in our system that involve interactions with objects, Cutler and Davis demonstrated in [14] a method for identifying periodic motion using low resolution video by Fourier analysis. This work was extended by [12] to detect baggage carried by a person.

3. Segmentation and Tracking

We use a Microsoft Kinect for Xbox camera to capture information about a scene. The Kinect provides a color image at VGA quality (640 x 480 pixels) as well as a depth image at VGA quality which provides information about the distance of objects from the camera. Both data streams are processed at 30 frames per second. The color and depth images are both utilized to obtain a robust image segmentation. The Kinect in particular is not critical to this application: Zhao et al. presented in [13] a method for segmentation and depth-position computation under the assumption of a static camera and known ground plane. Images are processed using the Open Source Computer Vision (OpenCV) library.

The block diagram in Fig. 1 shows the process architecture of our system. In the first stage, the system uses both color and depth information to segment images into background and foreground regions. In the second stage, the foreground pixels are grouped into blobs, and the identity of each blob is maintained across frames. In the third stage, the blob descriptions are analyzed by two independent modules: the event module (section 4.1) and the Backpack module (section 4.2). The results of these modules are then sent to the user interface, either live or offline.

3.1 Background Modeling and Foreground Segmentation

For each frame, a color and depth image is acquired from the camera and the system performs foreground-background segmentation to determine potentially relevant foreground objects. We use a Mixture of Gaussians (MOG) model as modified by Zivkovic in [15] as the basis for background subtraction with added higher-level recovery filters.

The MOG model for background subtraction uses Gaussian intensity distributions to model the image background [15]. Zivkovic’s method additionally allows detection and segmentation of shadows in the color image. Fig. 2 b-d shows the result of the color and depth background segmentations.

Some additional filtering is applied in addition to the MOG model. For every incoming color frame \( I \) with \( m \) rows and \( n \) columns the average intensity \( Q \) of the image is computed over every pixel \( I(x, y) \) as:

\[
Q = \frac{\sum_{y=0}^{m} \sum_{x=0}^{n} I(x, y)}{m \times n}
\]  

Given the intensity of the previous frame \( Q_{p-1} \) and the current intensity \( Q_p \) an intensity scale factor is computed as:

\[
F = \frac{Q_p - Q_{p-1}}{r}
\]  

where \( r \) is the history length of the MOG model. This scale factor is then broadcasted together with the color image. The effect is that sudden, global changes in lighting are smoothed across many frames and generally do not cause false positives in the foreground detection.

Occasionally, a major global lighting change, for instance a bright light bulb being turned on or off, cannot be smoothed by the intensity filter because the camera’s exposure level needs time to adjust to the new lighting levels, thus there is insufficient quality data to observe from the camera (this typically manifests itself as a completely black or completely white image).
Fig. 2: The stages of blob tracking. (a) The color input from the camera; (b) the color segmentation, notice many shadows are present; (c) the depth segmentation; (d) combined segmentation; (e) detected regions are outlined in a unique color; (f) The people are marked and labeled.

3.2 Blob Finding

A blob $B_p$ is a group of pixels in a locally dense area of an image which are considered to represent a single object. Pixels are initially classified into blobs using the border following approach from [16]. Partial and insignificant blobs are then merged and deleted using higher-level recovery filters described in Section 3.3.

The segmentation produced in Section 3.1 often fails in regions of the image where there is a sharp change in depth causing a shadow. One common example of this is the region where a person’s shirt sleeve meets his or her arm. There will often be three distinct blobs in the torso region: the torso itself and the two arms, with the arms separated from the torso by a thin line of background pixels at a sleeve.

To merge blobs which might be part of the same object a simple region-growth algorithm is implemented. Given a set of blobs in the image, each blob is dilated by one pixel and the border following approach from [16] is repeated. If two blobs are now connected they are merged into a single, larger blob. This process is repeated until a single blob remains for a maximum of ten iterations.

We define the minimum blob size $S_{min}$ as the minimum apparent area of a human at the far range of the Kinect’s depth stream (approximately six meters), experimentally found to be 400 pixels. If a blob $B_p$ does not satisfy:

$$\text{area}(B_p) > S_{min}$$

the blob is considered to be insignificant and is discarded.

3.3 Blob Classification and Tagging

Each blob in the list of significant blobs is then mapped to an existing blob or added as a new blob. At a high level, the system projects the position of each blob from the previous frames into the current frame and searches for a blob which is sufficiently similar to the known blob. The similarity comparison is performed by comparing the image histograms using the Chi-Squared method from [17]. At frame $p$, for blobs that have been tracked longer than 120 frames we also require the similarity between a previously observed blob $B_p'$ and a candidate matching blob $B_p$ to be within one standard deviation of the mean prior similarity. This extra element increases robustness to blobs occluded by similarly colored blobs. We define $x(B_p)$ as the bounding box of $B_p$ (with no filtering). A Kalman Filter [18] is used to filter the four components of $x(B_p)$:

$$x(B_p) = \{i, j, width, height\}$$

where $i, j$ is the location of the center of $x(B_p)$. At frame $p$ the a priori estimate of $x(B_i)$ is updated with the current observation to compute the filtered bounding box of $B_p$ called $\bar{x}B_p$.

This filtering affords the system greater fault tolerance by allowing tracking to appear to continue even if a blob is
temporarily not detected by the blob tracker. A temporary tracking failure might occur due to events such as a global lighting change (either due to ambient light changing or camera exposure adjustment), or occlusion by another person.

To update the previously observed blobs $B$, the system attempts to find a mapping between the list of currently observed blobs $B_p$ and the known blobs $B_p'$ using our Blob Candidate Matching algorithm in Fig. 3.

Every blob in $B'$ is next updated giving the mapping between $B$ and $B'$. A blob is updated with our Blob Update algorithm in Fig. 4.

4. Event Detection

We define an event as any of six detectable situations, which include people and their interactions with each other and baggage. The Event Module detects and generates information for five of these events where the people and baggage are visually separated, while the Backpack Module does the same for baggage which is attached to a person (such as a backpack).

4.1 Event Module

For each frame, once the system has a list of currently visible blobs $B'$ (more precisely: blobs which were matched successfully in the past 30 frames) a set of events can be identified. The blobs’ histograms, filtered positions in the image plane, and average depth position are used to identify events. Five classes of events are detected, as described below. In each case, the system marks a blob or set of blobs involved in each event type in the user interface when the event is detected.

- Person (SP). A user can describe a person of interest as a bimodal color distribution representing the lower and upper halves of the body, typically corresponding to shirt color and pants color. The user provides these colors as samples using either a standard HSV color picker or a previously acquired sample.
- Two People Meeting (TP). A user can describe two people of interest in a similar fashion to the single person detection. A meeting between two blobs representing these people is then characterized as:
  - the distance between the blobs’ geometric centers in 3D is no greater than the sum of their apparent widths in the image plane.
  - the blobs remain in this configuration for longer than one second.

The first criteria distinguishes between people meeting versus simply being in the same frame by coincidence and provides simple scaling in apparent size due to a blob’s distance from the camera, and the second criteria avoids detecting transient passing as a meeting.
- Bag Unattended (BA). Baggage is detected similarly to humans. A bag is characterized as:
- For each blob $B_p$ in $B$ and $B_p'$ in $B'$:
  - Where $D_{min}$ is the apparent depth of a body, if:
    \[
    center(x(B_p)) \in \bar{x}(B_p') \text{ or } center(\bar{x}(B_p')) \in x(B_p) \tag{4}
    \]
    and
    \[
    |\text{depth}(x(B_p)) - \text{depth}(\bar{x}(B_p'))| < D_{min} \tag{5}
    \]
  - Store $B_p$ as a candidate match for $B_p'$. This ensures the two blobs are close to each other in 3D.
- For each $B_p$ which is a candidate match for $B_p'$:
  - Compute the similarity between $B_p$ and $B_p'$ using either a standard HSV color picker or a previously acquired sample.
  - Accept $B_p$ as a match and update the blob (procedure described in Fig. 4).
  - Otherwise:
    - Compute $\text{mean}(S_{B_p,B_p'})$ and $\text{std.dev}(S_{B_p,B_p'})$ from the last 120 frames, and accept the current blob as a match if:
      \[
      S_{B_p,B_p'} > \text{mean}(S_{B_p,B_p'}) - \text{std.dev}(S_{B_p,B_p'}) \tag{7}
      \]
    - If $B_p'$ is not matched with any blob in $B$ but has been updated in the last 30 frames: add an artificial $S_{B_p,B_p'} = 0$ to its list of prior similarities. This prevents the known blob’s prior similarity from becoming arbitrarily precise and allows for tracking lapses of up to one second.

Fig. 3: The Blob Candidate Matching algorithm.

For every $B_p'$ and its match $B_p$:
- Store $S_{B_p,B_p'}$ from the Blob Update algorithm.
- Update its Kalman Filter $\bar{x}(B_p')$ with the new blob’s bounding box.
- Update its $U(B_p')$ and $L(B_p')$ as the average of the previously observed histograms.
- Update its depth as the average of the previously observed depths.

Fig. 4: The Blob Update algorithm.
- being one of a pair of blobs whose geometric centers in 3D are no further apart than the sum of their apparent widths in the image plane.
- being less than half the height of its companion blob.
In each pair, the taller blob is flagged as the owner blob and the shorter blob is flagged as the bag blob. Following a detection of a person and a bag, an unattended bag is characterized as: an owner blob becomes separated from its bag blob according to the 3D distance metric.

- **Two people exchanging a bag (BE).** Following the detection of an owner/bag pair and a second non-owner person blob, an exchange is characterized as:
  - an owner blob becomes separated from an owned blob while a different person blob is within owning distance of the bag blob.
- **A person stealing a bag (BS).** Following the detection of an owner/bag pair, a bag theft is characterized as:
  - a bag is abandoned by an owner. At this instant its bounding box in the image plane is recorded.
  - the bag’s geometric center leaves its recorded bounding box before the owner is reassociated with the bag.

As a user interface element, the closest person blob to a stolen bag will be reported as the thief, however the thief blob is not used for detection of the event.

### 4.2 Backpack Module

We implemented the periodicity analysis method from [14] as modified by [12] to detect baggage which is directly on a person’s body. This method utilizes the observation that the human silhouette is mostly symmetric when walking, and its non-symmetric regions are exhibiting a periodic motion. Thus non-symmetric, non-periodic regions of the silhouette are known to be not part of the body, typically an object such as a backpack or briefcase. This class of objects is characterized as:

- **Backpack (PK).** A backpack is a portion of a blob which is asymmetric and does not exhibit periodic motion.

The detection is implemented with our Backpack Detection algorithm in Fig. 6.

### 4.3 Live and Offline Interface

The six events which can be detected in live mode (five from the Event Module and one from the Backpack Module) also have a comparable offline mode. This mode is designed to be run on pre-recorded sequences in order to answer attribute-based queries such as “Did person with description X ever exchange a bag with person with description Y?”

The detection algorithms are performed offline and the result is a list of timestamps of when described event occurred. Once the detection is performed once, the event timestamps are saved and can be reused for future analysis. The offline interface is shown in Fig. 7.

- For each human blob $B'_p$:
  - Scale $B_p$ to 9x15 pixels and back to the original size to obscure details
    - If $B_p$ has not been seen before, store the location of its two left-side corners and centroid.
    - If $B_p$ has been seen before, warp it using an affine transformation such that its two left-side corners and centroid are in the same locations as the first time the blob was seen. This aligns the blob in each subsequent frame.
  - Compute the blob’s symmetry axis $V$ (vertical line in Fig. 5b) as the maximum value of the vertical projection, where $y_0$, $y_1$ are the left and right bounds of the blob, respectively:
    \[
    V(x) = \int_{y_0}^{y_1} B(x, y) \, dy
    \]
  - Remove portions of which are reflections across the blob’s symmetry axis within a margin of error, $E$ (white in Fig. 5b). We use $E = 5\%$.
  - Store the remainder of the silhouette, i.e. the asymmetric region (grey in Fig. 5b), call it $S_p$.
  - Compute and store the similarity $M_p$ between $S_p$ and every previously stored region $S'_p$ using (10), noting that $S_p$ is a binary image:
    \[
    M_p = \int_{x_0}^{x_1} \int_{y_0}^{y_1} S_p(x, y) - S'_p(x, y) \, dy \, dx
    \]
  - Compute the mean power spectrum $P$ of all the stored $M_p$ values for this silhouette.
  - Where $D$ is a scalar, if there is a peak in $P$:
    \[
    P(x) > mean(P) + D \times std\_dev(P)
    \]
    then mark $S_p$ is asymmetric, aperiodic and believed to be a backpack. We use $D = 5$, chosen to scale $std\_dev(P)$ to only mark peaks varying more than 99.5% from the mean.

![Fig. 5: (a) A person is tracked along with his backpack. A blue square is drawn around the backpack. (b) The internal representation of each blob in the backpack module. The dark grey vertical line represents the computed symmetry axis, the white regions are considered symmetric, and the grey regions are considered asymmetric.](image)

![Fig. 6: The Backpack Detection Algorithm](image)
5. Experimental Evaluation

To validate our approach we collected sample sequences of people walking around, interacting with each other, and interacting with baggage. Two examples of each event type were recorded. In each pair of sequences the environments differ in room color/design, lighting, camera position, and subjects' clothing color, in order to demonstrate the system robustness to various environments. Each video is manually tagged to determine when each event instance begins and ends, and this is used as the ground truth for our validation. The system is then evaluated on three metrics from [19]:

- **Accuracy** = the percentage of observation sequences in which the system’s selected event matches the ground truth.
- **Early detection** = $S' - S$, where $S$ is the start time of the event according to the ground truth and $s'$ is the start time detected by the system, in frames. Note that negative values are possible if the system detects the event before the ground truth.
- **Correct duration** = $\frac{T'}{T}$, where $T$ is the duration of the event according to the ground truth and $T'$ is the duration of the event detected by the system. Note that values greater than 1 are possible if the system detects the event longer than the ground truth.

For reliable event detection, the system should have high accuracy, a small value (close to zero) for early detection, and a large percentage (close to 1) for correct duration. The accuracy rate of our system is 100% in every sequence tested. Note that redundant events are not reported, e.g. an instance of two people meeting is also two separated instances of a single person. Table 1 shows the values for early detection and correct duration for each sequence.

For 10 of 12 cases the system detected the event within 16 frames (approximately one second) of the ground truth start time. The backpack module takes longer to begin tracking an event by design: periodicity is a function of time, thus strong peaks in the power spectrum of the backpack Fourier analysis only begin to appear about 60 frames (approximately two seconds) after the object tracking begins.

Event start and end times for the ground truth were taken to occur when a person made a visible attempt to commit the action, for instance reaching out to take the bag in the bag exchange and steal events. The variance for early detection is primarily due to these actions not necessarily corresponding to the method for determining events as described in Section 4. For event end times (which affect correct duration) there is variance among different interaction types. For instance, in BE2 the system detected a duration almost two times longer than the one indicated in the ground truth. This was observed because the people in the sequence walked past each other after the bag exchange, thus they were still in close proximity to each other and the bag, even though the exchange had finished.

We believe these results accomplish the objective of correctly identifying events to draw a human operator’s attention. Slight variances in start and end times are insignificant in this case because an operator could simply rewind the video and play back as much of the sequence as they desire. There is an ambiguity in some event classes which was removed before determining a ground truth for some events. For a single person, two people meeting, and bag unattended it is easy to determine when the events start and end. The other events are more difficult to codify: does a bag theft happen when the owner initially leaves the bag unattended? Or when the thief approaches the bag? Or when the thief actually moves the bag? To disambiguate these circumstances the following rules are applied:

- A bag theft begins when a thief moves a bag from the location in which it was abandoned, and ends when the thief replaces the bag or the thief is no longer visible.

<table>
<thead>
<tr>
<th>Event Type</th>
<th>Early Detection (frames)</th>
<th>Correct Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP1</td>
<td>3</td>
<td>1.0000</td>
</tr>
<tr>
<td>SP2</td>
<td>4</td>
<td>0.9591</td>
</tr>
<tr>
<td>TP1</td>
<td>2</td>
<td>1.1210</td>
</tr>
<tr>
<td>TP2</td>
<td>16</td>
<td>0.9012</td>
</tr>
<tr>
<td>BA1</td>
<td>5</td>
<td>0.9896</td>
</tr>
<tr>
<td>BA2</td>
<td>5</td>
<td>0.9829</td>
</tr>
<tr>
<td>BE1</td>
<td>-1</td>
<td>1.1905</td>
</tr>
<tr>
<td>BE2</td>
<td>2</td>
<td>1.7407</td>
</tr>
<tr>
<td>BS1</td>
<td>-8</td>
<td>1.0096</td>
</tr>
<tr>
<td>BS2</td>
<td>2</td>
<td>0.9551</td>
</tr>
<tr>
<td>PK1</td>
<td>56</td>
<td>0.6419</td>
</tr>
<tr>
<td>PK2</td>
<td>61</td>
<td>0.6324</td>
</tr>
</tbody>
</table>
We plan to expand this system by adding an autonomous robotic element to the detection system in order to provide information that cannot be obtained by a static camera. For instance, a robot could be deployed to a certain location to get a better view of a particular event, resolve occlusions, or provide a more detailed view of a person involved in a specific action.

6. Conclusion and Future Work

In this paper we presented a robust system for detecting human agents and interactions with the objects they carry. We use computer vision techniques to segment and track people and bags, and infer the relationships between them. We also use periodicity and symmetry information to perform a Fourier analysis on human blobs to determine if the person is carrying a backpack or other large item. The proposed system demonstrates the potential of fully automated, vision-based solutions for detecting pre-defined behaviors and reporting instances of these events to a human operator. Such system would be of use to security personnel in sensitive public places such as airports or federal buildings. The system would significantly reduce the workload of human operators required to manually detect security events from live video streams. Additionally, the offline mode allows security or law enforcement personnel to search in recorded video for specific events much faster than replaying and watching an entire video.

References