Real-time Event Detection and Its Application to Surveillance Systems

Hong-Yuan Mark Liao, Duan-Yu Chen, Chih-Wen Su, and Hsiao-Rang Tyan

Abstract—In recent years, real-time direct detection of events by surveillance systems has attracted a great deal of attention. In this paper, we propose a new video-based surveillance system that can perform real-time event detection. In the background modeling phase, we adopt a mixture of Gaussian approach to determine the background. Meanwhile, we use color blob-based tracking to track foreground objects. Due to the self-occlusion problem, the tracking module is designed as a multi-blob tracking process to obtain similar multiple trajectories. We devise an algorithm to merge these trajectories into a representative one. After applying the Douglas-Peucker algorithm to approximate a trajectory, we can compare two arbitrary trajectories. The above mechanism enables us to conduct real-time event detection if a number of wanted trajectories are pre-stored in a video surveillance system.

I. INTRODUCTION

Real-time video-based surveillance systems have become increasingly important in recent years due to increasing crime. In the past, video-based surveillance systems relied primarily on human operators to observe several, or dozens, of monitors simultaneously. However, this kind of monitoring is not practical, because a human operator cannot watch so many TV screens simultaneously and observe everything. Furthermore, unlike machines, human operators get tired, especially after working long hours. Therefore, it is essential to develop a smart real-time video-based surveillance system; however, some difficult design issues must be addressed. First, the motion of a human being is highly articulated, which makes the description task very difficult. Second, in order to correctly identify an event, comparison of two arbitrary motion sequences is necessary.

In the past decade, extensive research have been conducted into surveillance-related issues [1-11][17]. In [1], Wren et al. proposed a statistical background model to locate people and utilized 2D image analysis to track a single person in complex scenes with a fixed camera. Lipton et al. [2] proposed the use of a human operator to monitor activities over a large area using multiple cameras. Their system can detect and track multiple people and vehicles in crowded scenes for long periods. Grimson et al. [3] established a multiple-camera environment to learn patterns common to different object classes and detect unusual activities. Bobick et al. [4] proposed a combination of a Hidden Markov Model and stochastic grammar to recognize activities and identify different behavior based on contextual information acquired by a static camcorder. Kanade et al. [5] proposed the use of multiple sensors to detect and track moving objects. Haritaoglu et al. [6] proposed the detection and tracking of multiple people and monitoring of their activities in an outdoor environment. Stauffer and Grimson [7] focused on motion tracking in outdoor environment. They used observed motions to learn patterns from different kinds of activities. In [8], Zelnik and Irani proposed a non-parametric approach to characterize video events by spatio-temporal intensity gradients calculated at multiple temporal scales. Medioni et al. [9] used a set of features derived from multiple scales to stabilize an image sequence. They proposed the extraction of trajectories of moving objects using an attribute graph representation. Davis and Bobick [10] proposed an activity recognition method based on view-dependent template matching. In [11], Davis et al. proposed the representation of simple periodic events, such as walking, by constructing dynamic models based on computing periodic patterns in people’s movements. In [17], Su et al. proposed the concept of motion flow and used it to represent an event directly in a compressed domain. They also proposed a scale and translation invariant matching algorithm to compare an unknown event and database events.

In this paper, we propose a new video-based event detection system. In the background modeling phase, we adopt a mixture of Gaussian approach to determine the background. Through this simple modeling, we can separate foreground objects from the background efficiently and accurately. To track foreground objects, we use color blob-based tracking. However, the method is not perfect due to the effect of self-occlusion or mutual occlusion among various body parts. Therefore, the tracking process is designed as a multi-blob tracking process that generates multiple similar trajectories. We have also designed an algorithm that merges these trajectories into one representative trajectory and apply the Douglas-Peucker algorithm to approximate it. To compare a trajectory extracted from a real-time environment with those stored in the database, we propose a translation and scaling invariant metric to execute the matching task. Using the above procedure, we can detect abnormal intrusion events in real time by pre-storing a number of possible intrusion trajectories in a local computer linked to a camcorder monitor. If the path of an intruder is close to one of the pre-stored trajectories, the system sends a signal directly to the control center. The contribution of this work is twofold. First, it simplifies the real-time event detection problem by comparing the degree of similarity between two arbitrary trajectories. Second, the proposed method is translation and scaling invariant. It can therefore be applied to different application domains, such as home security systems, complex surveillance systems, or parking lot management systems.

The remainder of this paper is organized as follows. In Section 2, we describe how background modeling and foreground detection are achieved. In Section 3, we introduce a novel hybrid object tracker. In Section 4, we introduce a novel comparison algorithm that can calculate the degree of similarity between two arbitrary trajectories. Finally, we present our experiment results and conclusions in Section 5 and Section 6, respectively.
II. FOREGROUND AND BACKGROUND ESTIMATION

Since our objective is real-time event detection using a stationary camera, distinguishing foreground objects from the background is an important step in our surveillance system. Foreground estimation is relatively easy in an indoor environment, because the illumination conditions do not change significantly. An outdoor environment, on the other hand, is much more complicated, as varying weather and sunlight affect the correct detection of foreground. Some researchers have adopted an adaptive Gaussian approach to model the behavior of a pixel. However, the background region of a video sequence often contains several moving objects. Therefore, rather than explicitly estimating the values of all pixels as one distribution, we prefer to estimate the value of a pixel as a mixture of Gaussians [15]. In this approach, pixel values that do not match the weighted sum of the background distribution are considered foreground objects.

In [12], the probability that an observed pixel will have an intensity value \( x_t \) at time \( t \) is estimated by \( K \) Gaussian distributions defined as follows:

\[
P(x_t) = \sum_{i=1}^{k} \frac{\omega_i I(x_t)}{(2\pi)^{n/2}} \exp \left( -\frac{1}{2} (x_t - \mu_i)^T \Sigma_i^{-1} (x_t - \mu_i) \right),
\]

where \( \omega_i \) is the weight of the \( i \)th distribution of pixel \( x_t \)'s mixture model; \( \mu_i \) is the mean of the \( i \)th distribution; and \( \Sigma_i \) is its covariance matrix, where \( \Sigma_i = \sigma_i^2 I \). \( \sigma_i \) is the standard deviation of the \( i \)th distribution, and \( I \) is an identity matrix. To update the model, each new pixel is checked to see if it matches the existing Gaussian distributions. To adjust the weight of each distribution, the weight \( \omega_i \) is updated by

\[
\omega_i = (1 - \alpha) \omega_i + \alpha (M_i, t),
\]

where \( \alpha \) is the learning rate that controls the speed of the learning; \( M \) is a Boolean value indicating whether or not a match is found. The definition of \( M \) is as follows:

\[
M_{i,t} = \begin{cases} 1 & \text{when a match is confirmed on the } i \text{th distribution at time } t; \\ 0 & \text{otherwise, } \end{cases}
\]

The parameters \( \mu \) and \( \sigma \) can be updated as follows:

\[
\mu_{i,t} = (1 - \beta) \mu_{i,t-1} + \beta x_t,
\]

\[
\sigma_{i,t}^2 = (1 - \beta) \sigma_{i,t-1}^2 + \beta (x_t - \mu_{i,t})^T (x_t - \mu_{i,t}),
\]

where \( \beta = \alpha \sigma_i^2 \). In each frame, pixels far away from the learned and distributions recognized as foreground. A connectivity algorithm is then applied to identify possible objects in motion. It is widely recognized that an overly segmented result may break the detected objects into pieces. Therefore, morphological operations must be applied to fix the completeness of foreground objects. A detected foreground object is considered as a blob and characterized by its position and color distribution to support the subsequent tracking process.

III. COLOR-BASED BLOB TRACKING

A. Color-based Blob Tracker

In order to satisfy the real-time constraint, we employ color histograms to characterize the extracted blobs. Although a color histogram is not the best nonparametric estimator [13], it is a good target tracker due to its low computation cost. Moreover, since it disregards all geometric information, it is robust against complicated non-rigid motions. Using a color histogram, a target \( R \) and its corresponding candidates \( P \) can be modeled as

\[
\text{Target: } R_t = \{r_{i,t} \}_{i=1}^{n}, \text{ where } \sum_{i=1}^{n} r_{i,t} = 1.
\]

Corresponding candidates:

\[
P(x,y) = \{p_{i,j}(x,y) \}_{i=1}^{n}, \text{ where } \sum_{i=1}^{n} p_{i,j} = 1.
\]

\[
\begin{align*}
\text{For } & P(x,y) \text{ and } p_{i,j}(x,y), \\
& \text{where } P \text{ is the candidate located at } (x,y) \text{ and the value of its } i \text{th bin at time } t, \text{ respectively; and } r_{i,t} \text{ represents the value of the } i \text{th bin at time } t. \\
& \text{The histograms with } n \text{ bins of } R \text{ and } P \text{ are normalized by their own total pixel numbers.}
\end{align*}
\]

Next, we employ a similarity function, called the Bhattacharyya coefficient [14], to calculate the degree of similarity between the target and a candidate in the database. A Bhattacharyya coefficient is a divergence-type measure with a straightforward geometric interpretation. It is defined as the cosine of an angle measured between two \( n \)-dimensional unit vectors, \( (\sum_{i=1}^{n} p_{i,j}) \) and \( (\sum_{i=1}^{n} r_{i,j}) \). Detailed definitions of the Bhattacharyya coefficient can be found in [15]. The formal definition of the distance between a target at time \( t \) and a candidate at time \( t+1 \) is

\[
d(x,y) = \sqrt{1 - \sum_{i=1}^{n} p_{i,j}(x,y) r_{i,j}}.
\]

Therefore, to find the best match among a set of candidates located in the subsequent frame, the candidate that has the smallest distance with the target is chosen. We then use the color distribution of this chosen candidate to update the color distribution of the target. The tracking procedure is iterated until the target cannot find any further match. Since a color-based blob tracker may generate more than one trajectory due to occlusion or the distinct movements of different body parts, we now discuss how to merge these trajectories into one representative trajectory.

B. Single out A Representative Trajectory from A Group of Trajectories

Since the movement of an articulated object like a human being may cover more than one trajectory due to occlusion or the movements of different body parts, it is possible to derive more than one trajectory from a moving person. Therefore, we adopt an algorithm proposed in our previous work [17] to single out one representative trajectory from a set of derived trajectories. Initially, we select a trajectory, \( a \), that has the longest duration in a tracking sequence. Suppose \( a \) starts from time \( t_s \) and ends at time \( t_e \). \( B_d \) is the set of trajectories whose start or end times are within the duration. We merge \( b \) (b is a member of \( B_d \)) with \( a \) if it satisfies the following condition:

\[
\sum_{i=1}^{n} \left( a(t_i) - a(t_{i-1}) \right) \left( b(t_i) - b(t_{i-1}) \right) \leq \varepsilon,
\]

where \( t_i \) and \( t_{i-1} \) are respectively the start and end times of trajectory \( b \). Let \( a(t) \) and \( b(t) \) be the spatial position of \( a \) and \( b \) at time \( t \), respectively; and let \( a(t_i) \) and \( b(t_i) \) be the spatial position of \( a \) and \( b \) at time \( t_i \), respectively. If the average of the relative spatial distances between \( a \) and \( b \) is smaller than a threshold, \( \psi \), we consider \( b \) a highly correlated trajectory of \( a \). Therefore, the trajectory \( a \) is updated by the average position of trajectories \( a \) and \( b \) using Eq.9.

\[
g(t) = \begin{cases} h(t), & \text{if } a(t) \in \phi \psi, \\
(\alpha a(t) + \beta b(t))/2, & \text{otherwise,} \\
\end{cases}
\]

where \( \phi \) represents an empty set. We consider \( a \) as the representative trajectory of \( B_d \) and store it temporarily. Next, we select the longest trajectory from the remaining trajectories, and repeat the above process. If the trajectory under consideration is longer than the temporary one, we
control points of the trajectory. The selected points and the two end points form the set of the farthest intermediate point. This process continues until all than a threshold, we split the trajectory into two segments via between any intermediate point to the anchor line is larger end points of the trajectory. If the perpendicular distance for comparing two distinct trajectories.

detail, the proposed scaling and translation invariant algorithm any pre-trained trajectories. In the following, we describe, in tracker, we can warn the human operator immediately if a surveillance systems usually have a large number of monitors in the real world, compare the degree of similarity between this trajectory and tracking process, we need an efficient algorithm that can

to make a stairway. Due to the occlusion of the handrail, the person is split into three blobs in Figs.1(a) and 1(b), and two blobs in Fig.1(c). From the blob-moving sequence, our algorithm detects several trajectories, as illustrated in Fig.1(d), and singles out a representative trajectory as shown in Fig.1(e).

Fig. 1. An example showing a representative trajectory is singled out from a group of trajectories. (a)-(c): tracking of a person whose body is split into three parts due to occlusion of the handrail; (d) the group of calculated trajectories; and (e) the representative trajectory derived by applying the proposed algorithm.

IV. REPRESENTING A TRAJECTORY AND MATCHING

After obtaining a trajectory calculated by a real-time tracking process, we need an efficient algorithm that can compare the degree of similarity between this trajectory and the trajectories stored in the local database. In the real world, surveillance systems usually have a large number of monitors located in a control center and handled by a human operator. By using the above mentioned real-time motion trajectory tracker, we can warn the human operator immediately if a tracked trajectory caused by an abnormal event is similar to any pre-trained trajectories. In the following, we describe, in detail, the proposed scaling and translation invariant algorithm for comparing two distinct trajectories.

We adopt the Douglas-Peucker algorithm [16] to select the necessary control points from a trajectory. The algorithm starts by using a straight line segment to connect the start and end points of the trajectory. If the perpendicular distance between any intermediate point to the anchor line is larger than a threshold, we split the trajectory into two segments via the farthest intermediate point. This process continues until all perpendicular distances are smaller than the pre-set threshold. The selected points and the two end points form the set of control points of the trajectory.

Fig. 2. The alignment task between two trajectories.

We use five positive real numbers \((x^+, x^-, y^+, y^-, d)\) to represent the position of a control point. Here, \(d\) denotes the cumulative length of the trajectory from the first control point to the current control point; and \(^+/−\) denotes the cumulative positive/negative movement along the \(x\)- or \(y\)-axis from the first control point. Now, let \(Q\) and \(D\) be the trajectories of the query and a model in the database, respectively. We normalize the length of both trajectories into a unit length before making a comparison. This technique guarantees the requirement of scale invariance. Therefore, the parameters \(d, x^+, x^−, y^+,\) and \(y^−\) of each control point on the two trajectories must be normalized by dividing them by the length of \(Q\) and \(D\), respectively.

We align both \(Q\) and \(D\) by calculating the length \(d\) from the first control point. For each control point on \(Q\) \((D)\), we interpolate a corresponding point that has the same cumulative length onto \(D\) \((Q)\). The \(“d”\) value is used as the basis of the alignment task, because we only consider the similarity between \(Q\) and \(D\) in the spatial domain. The control points and the corresponding points are labeled by circles and triangles, respectively (Fig.2). The insertion of the corresponding points on \(Q\) and \(D\) is dependent of \(“d”\). Now, for each control point on the trajectory \(Q\) \((D)\), we can interpolate a corresponding point located on \(D\) \((Q)\). Assume the total number of control points and their corresponding points located on \(Q\) and \(D\) are both \(N\). Let \(Q’\) \(=\) \(\{Q_1’, Q_2’, \ldots, Q_N’\}\) and \(D’\) \(=\) \(\{D_1’, D_2’, \ldots, D_N’\}\) be the set of points (including the control points and the inserted corresponding points) located on \(Q\) and \(D\), respectively. The set of points \(Q’\) and \(D’\) can be called “check points”, each of which can be represented by \((x^+, x^−, y^+, y^−)\) in the spatial domain. In order to compare two arbitrary trajectories, we define a metric as follows:

\[
\text{Dis}_{ij}^{Q,D} = \left| \begin{array}{c}
10 \\
10 \\
00 \\
01 \\
01 \\
\end{array} \right|,
\]

where \(i\) and \(j\) \((i < j)\) denote the \(i\)-th and \(j\)-th check points of two partial trajectories of \(Q’\) and \(D'\), respectively; \((Q_1’, Q_2’)\) and \((D_1’, D_2’)\) represent the difference between the \(i\)-th and \(j\)-th check points of \(Q’\) and \(D’\), respectively; and \((Q_1’, Q_2’)-(D_1’, D_2’))\) is a \(1 \times 4\) vector, and its subsequent term in Eq.10 is a \(4 \times 2\) matrix. Before executing the matrix operation, the absolute value of each element of \((Q_1’, Q_2’)-(D_1’, D_2’))\) must be taken. The operation on the right-hand side of Eq.10 will result in a \(1 \times 2\) vector. In addition, \(\text{Dist}^{Q,D}_{ij}\) is basically an estimation of the distance between two partial trajectories on \(Q’\) and \(D’\), respectively. With the above distance metric, we can define the total distance between \(Q\) and \(D\) as follows:

\[
\text{TDist}(Q,D) = \sum_{i=1}^{N} \text{Dist}^{Q,D}_{ij}.
\]

The advantage of the proposed representation scheme is that we do not really need to compare the check points pair by pair. It should be noted, however, that all the elements that form the vector of a check-point on a trajectory are positive and their magnitudes are accumulated from the beginning. Therefore, if we choose an intermediate check point, \(Q_i’,\) in \(Q’\) and its corresponding check point, \(D_i’,\) in \(D’\), we can be sure that

\[
\left\| \text{Dist}^{Q,D}_{ij} \right\| \leq \text{Dist}^{Q,D}_{ij} + \text{Dist}^{Q,D}_{ij}.
\]

Eq.12 shows that a coarse-to-fine search strategy is appropriate for a trajectory-based query. In the first step of the comparison between \(Q’\) and \(D’\), we simply check the value of \(\text{Dist}^{Q,D}_{ij}\). This step only needs to consider four check points \(Q_1’, D_1’, Q_2’,\) and \(D_2’\). Since the value of \(\text{TDist}(Q,D)\) must be equal to or larger than that of \(\text{Dist}^{Q,D}_{ij}\), we can quickly determine that trajectory \(D\) is not similar to \(Q\) if the returned value of \(\text{Dist}^{Q,D}_{ij}\) is greater than a predefined threshold \(\delta\).

Once the value of \(\text{Dist}^{Q,D}_{ij} \leq \delta\), we seek the second check points on \(Q’\) and \(D’\), respectively, by checking \(Q_3’\) and \(D_3’\). If \(Q_3’\) is chosen as \(Q_2’\), we can insert \(D_1’\) into the right position between \(D_1’\) and \(D_3’\) and vice versa. Furthermore, \(Q’\) and \(D’\) can be divided into four sub-trajectories by \(Q_2’\) and \(D_2’\).
Under these circumstances, we only compute the sum of $\Delta Dist^{Q',D'}$ and $\Delta Dist^{Q',D''}$ as the distance between the sub-trajectories. If the distance between two distinct sub-trajectories is still larger than a predefined threshold $\delta$, $D$ will be filtered out. Otherwise, we insert $Q_i$ and $D_i$ to further compute $\Delta Dist^{Q',D'}$ and $\Delta Dist^{Q',D''}$. The above newly computed distances replace the value of $\Delta Dist^{Q',D'}$, and the process is executed repeatedly until the computed distance is larger than $\delta$, or there are no more intermediate check points within each sub-trajectory. Since most of the trajectories would be filtered out by checking the first few control points, our proposed algorithm is very efficient.

V. EXPERIMENT RESULTS

In order to test the effectiveness and efficiency of our real-time event detection method, we tested our algorithm on the proposed surveillance systems with monitors placed at eight different locations. These indoor/outdoor environments included a parking lot, the area in front of an elevator, and a stairway, etc. To demonstrate the performance of the proposed real-time surveillance system, we conducted a robustness test on our proposed system. The purpose of this set of experiments was to analyze the effect of false alarms. The ultimate goal of our system is to detect the movement of human beings and to “ignore” the movement of other moving objects. In order to perform real-time abnormal event detection, we pre-stored a set of trained trajectories to support real-time event detection. The pre-stored trajectories are shown in Fig.3(a). Fig.3(b) shows the trajectories detected in real-time when an intruder tried to climb the wall. Since the detected trajectories (the black curve shown in Fig.3(b)) was very close to the trained set of trajectories, the proposed surveillance system triggered an alarm to alert the human operator working in the control center. To test the robustness of the system, we conducted another experiment. Fig.3(c) shows the detected path of a moving basketball which is illustrated by the black curve. As one can see from Fig.3(c), the detected trajectory was very different from the pre-trained trajectories. Therefore, our system did not detect it as an intrusion event, because our algorithm could measure the difference between this trajectory and the database trajectories. In such a scenario, our system would not usually generate a false alarm.

The advantage of our approach is that a simple, but efficient, color-based object tracker is used following with trajectory refinement, instead of relying heavily on a complex object tracker. The response times of the above video retrieval experiments were very short because of our specially designed comparison procedure. Clearly, our system can be used to reduce the burden of monitoring many TV screens. By using a training set of anomalous events, false alarms are not be triggered; and thus, the proposed real-time surveillance system can help human operators conduct more precise and efficient surveillance tasks.

VI. CONCLUSIONS

In this work, we have proposed a new video-based surveillance system for real-time event detection. A simple, but efficient, color blob-based tracker has been developed to accomplish the multi-object tracking task. In addition, we have devised an algorithm to merge multiple trajectories into a representative one. After applying the Douglas-Peucker algorithm to approximate a trajectory, we are able to compare two arbitrary trajectories. The above mechanism enables us to conduct real-time event detection if a number of wanted trajectories are pre-stored in a video-based surveillance system.

ACKNOWLEDGEMENT

This work was supported by the Ministry of Economic Affairs under Contract No. 94-EC-17-A-02-S1-032.

REFERENCES