Incremental Behavior Modeling and Suspicious Activity Detection

Kan Ouivirach*, Shashi Gharti, Matthew N. Dailey

Computer Science and Information Management
School of Engineering and Technology, Asian Institute of Technology
P.O. Box 4, Klong Luang, Pathumthani 12120 Thailand

Abstract

We propose and evaluate an efficient method for automatic identification of suspicious behavior in video surveillance data that incrementally learns scene-specific statistical models of human behavior without requiring storage of large databases of training data. The approach begins by building an initial set of models explaining the behaviors occurring in a small bootstrap data set. The bootstrap procedure partitions the bootstrap set into clusters then assigns new observation sequences to clusters based on statistical tests of HMM log likelihood scores. Cluster-specific likelihood thresholds are learned rather than set arbitrarily. After bootstrapping, each new sequence is used to incrementally update the sufficient statistics of the HMM it is assigned to. In an evaluation on a real-world testbed video surveillance data set, we find that within one week of observation, the incremental method’s false alarm rate drops below that of a batch method on the same data. The incremental method obtains a false alarm rate of 2.2% at a 91% hit rate. The

*Corresponding author. Tel: +662-524-5700; Fax: +662-524-5721
Email addresses: kan@ieee.org (Kan Ouivirach), shashi.gharti@gmail.com (Shashi Gharti), mdailey@ait.ac.th (Matthew N. Dailey)

Preprint submitted to Pattern Recognition November 23, 2012
method is thus a practical and effective solution to the problem of inducing scene-specific statistical models useful for bringing suspicious behavior to the attention of human security personnel.

Keywords: hidden Markov models, incremental learning, behavior clustering, sufficient statistics, anomaly detection, bootstrapping

1. Introduction

Human monitoring of video surveillance channels is increasingly ineffective as the number of channels increases. Anomaly detection aims to alleviate this problem by filtering out typical events and only bringing suspicious events to the attention of human security personnel.

We propose a new incremental method for learning behavior models and detecting anomalies using hidden Markov model (HMM)-based clustering on a small bootstrap set of sequences labeled as normal or suspicious. After bootstrapping, we assign new observation sequences to behavior clusters using statistical tests on the log likelihood of the sequence according to the corresponding HMMs. We label a sequence as suspicious if it maps to an existing model of suspicious behavior or does not map to any existing model. After labeling, we either incrementally update the sufficient statistics for the most likely HMM’s parameters or create a new HMM for the new sequence.

In an evaluation on a real-world video surveillance situation, we find that the method is very effective at identifying suspicious behavior. A batch method achieves a false positive rate of 8.6% at a 100% hit rate. The incremental method decreases the false alarm rate to 2.2% at a 91% hit rate, and detection threshold tuning can achieve an equal error rate of 95.8% or
a false alarm rate of 3.7% at a 100% hit rate. The experimental results demonstrate that our incremental method is better able to learn new forms of normal behavior, leading in turn to more effective anomaly detection. The method is thus a practical and effective solution to the problem of inducing scene-specific statistical models useful for bringing suspicious behavior to the attention of human security personnel.

2. Related Work

In this paper, we explore methods for incremental learning of behavior models and active detection of anomalous deviations from the learned typical behavior. We build upon previous research in human behavior understanding [1, 2, 3, 4, 5, 6, 7], the use of dynamic graphical models such as hidden Markov models (HMMs) and conditional random fields (CRFs) for behavior understanding in specific scenarios [8, 9, 10, 4, 11], and classification of behavior in videos to a priori known categories [12, 13, 14, 15].

Several research groups have investigated methods for video clustering [16, 17, 18, 19, 20, 10, 21]. Especially effective are those methods using HMMs to model different classes of behavior in video surveillance [20, 10, 21].

There has been some work on anomalous time series detection outside the context of video surveillance using relatively simple statistical models [22, 23]. However, these methods learn single comprehensive models without addressing the special requirement in video surveillance to capture an extremely wide variety of typical behaviors. This requires construction of a collection of statistical models. Intrusion detection requires similar diversity in the statistical models. Some of the work uses ensembles of classifiers [24],
but most of the research has focused on anomaly detection methods using incremental clustering [25, 26, 27, 28, 29]. All of these methods generally work by comparing a new pattern against a collection of clusters representing historically typical behavior and classifying the new pattern as an anomaly if its distance from the nearest cluster is above threshold. We take a similar approach, with a specific approach to clustering, distance measurement, and threshold learning specific to the case of surveillance video.

Some incremental learning approaches have been applied to human behavior modeling. Vasquez et al. [30] incrementally model vehicle and pedestrian trajectories using growing hidden Markov models (GHMMs). The method builds a topological map and corresponding GHMM using an incremental variant of the Baum-Welch algorithm. For each new node in the map, a corresponding state and connectivity are added to the GHMM. Kulić and Nakamura [31] model movement primitives using HMMs and piece the primitives together with a higher level HMM. Similar to Vasquez et al., the method incrementally adds new hidden states when a new primitive model is built.

The existing work most similar to ours is the incremental learning approach of Xiang and Gong [32], who also model activity in a scene incrementally after initialization from a small bootstrap dataset. They build explicit probabilistic models of normal and abnormal activity patterns in the bootstrap set then classify new observations using a likelihood ratio test. The models are global joint models over the entire scene. It is not clear how the likelihood ratio test could handle completely new anomalous events with low likelihoods under both models, and it is not clear whether a global model is appropriate for detecting isolated anomalous behavior in a scene.
Our method does not require an a priori model of anomalous human behavior. It constructs an ensemble of simple models, adding to the ensemble when the existing set is insufficient to represent new cases. We associate each new observation with an individual model using separate statistical tests on the observation’s likelihood according to each individual model. Rather than globally modeling the entire scene, our method takes a more local approach, separately analyzing each individual moving object’s behavior. In an experimental comparison of the two methods, we find statistical testing of the likelihood to be superior to likelihood ratio tests, and we find the local representation to be superior to the global representation.

A preliminary report on our clustering method appeared in a conference paper [33]. In the current paper, we have extended this work to include multiple human tracking, anomaly detection, and incremental learning.

The main contribution of the work described in this paper is a new modeling method for detection of anomalous events in surveillance video based on simply-structured models and incremental learning that is demonstrably able to evolve the models over time to adapt to new behavior and also outperforms current techniques on the same data set.

3. Behavior Model Bootstrapping

Our method constructs behavior models for stationary cameras observing a specific scene. We begin with batch learning from a labeled bootstrap set comprising a video stream acquired over a short period of time such as one week. An overall block diagram of this step is shown in Figure 1. We use Poppe et al.’s [34] background modeling technique to segment fore-
Figure 1: Block diagram of behavior model bootstrapping method.

ground pixels from the background. No-motion frames are discarded based on whether a sufficient number of foreground pixels are sufficiently different between subsequent frames. We define an “event” as a contiguous change or motion detected over some period of time. Extremely short events (occasionally occurring due to noise) are automatically removed before processing.

To eliminate shadows, we apply normalized cross correlation (NCC) by computing the grayscale correlation between the foreground pixels and a background image constructed as the mean over each mixture of Gaussian distribution. Any foreground pixels whose NCC with the background are above some threshold are removed. We apply morphological opening then closing operations then obtain connected components (blobs) and filter out small components. Figure 2 shows sample results. Next we track the obtained blobs, extract a sequence of feature vectors describing each blob’s trajectory and appearance over time, and automatically bootstrap a bank of linear HMMs, each model specializing in one type of behavior.

3.1. Appearance-Based Blob Tracking

In this step, we take at time $t$ a list of blobs detected by the previous step, a set of tracks updated at time $t - 1$, and a merged track associa-
Figure 2: Sample foreground extraction and shadow removal results. (a) Original image. (b) Foreground pixels according to background model. (c) Foreground pixels after shadow removal.

tiation matrix. We output an updated track list and merged track association matrix. Algorithm 1 is a pseudocode summary of our approach. We first construct a matrix in which each element indicates the overlap area of the bounding box of a current blob with an existing track. When a blob is found to correspond to a single unique track, the track update is simple; we just associate the blob with the track. If a track is no longer visible for some period of time, it will be considered stale and deleted. Special handling is required for cases in which a new blob overlaps multiple old tracks (merges) or multiple new blobs overlap the same old track (splits). To handle these cases, our method evaluates a similarity function $S(b,t)$ for each candidate blob-track pair $(b,t)$. $S(b,t)$ is based on the color coherence vector (CCV) \cite{35}; when tracks merge, we group them, but keep their identities separate, and when tracks split, we associate the new blobs with the correct tracks or groups of tracks by comparing their CCVs, on the assumption that when an object eventually separates from the group, its appearance will be similar to its appearance before the merge. Our tracking method performs very well on typical simple cases such as those shown in Figure 3 involving clear trajectories of each moving person or object. However, it still has some problems
Algorithm 1 Appearance-Based Blob Tracking

**Input:** $B$: set of all current blobs
**Input:** $T$: set of all current tracks
**Input:** $M$: merged track association matrix
**Output:** $\tilde{T}$: set of all revised tracks
**Output:** $\tilde{M}$: revised merged track association matrix

$\tilde{T} \leftarrow \emptyset$; $\tilde{M} \leftarrow \emptyset$; $L \leftarrow \emptyset$

$A \leftarrow \text{Get-Overlap-Area-Matrix}(B,T)$

for each $t \in T$ do
    if $t$ is marked as processed then continue
    $B' \leftarrow \{b' \mid A(b',t) > 0\}$ \{$B'$ contains candidate blobs for track $t$.\}
    $T' \leftarrow \{t\} \cup \{t' \mid M(t,t') = 1\}$ \{$T'$ contains all tracks currently merged with $t$.\}
    if $|B'| \geq 1$ then
        for each $t' \in T'$ do
            Let $b = \text{argmax}_{b' \in B'} S(b',t')$
            $L \leftarrow L \cup \{(t',b)\}$
            MARK-TRACK-AS-PROCESSED($t'$)
        end for
    end if
end for

for each $(t_i,t_j) \in T \times T$ do
    If $\exists b$ s.t. $(t_i,b) \in L \land (t_j,b) \in L$, $\tilde{M}_{ij} \leftarrow 1$, otherwise $\tilde{M}_{ij} \leftarrow 0$
end for

$T^* \leftarrow \{t^* \mid \neg \exists b \in B$ s.t. $(t^*,b) \in L\}$ \{$T^*$ contains tracks for which “stale count” will be increased.\}

$\tilde{T} \leftarrow \text{Update-Or-Delete-Stale-Tracks}(T,T^*)$

$B^* \leftarrow \{b^* \mid \neg \exists t \in T$ s.t. $(t,b^*) \in L\}$ \{$B^*$ contains blobs with no tracks assigned.\}

$\tilde{T} \leftarrow \text{Add-New-Tracks-For-Not-Linked-Blobs}(\tilde{T},B^*)$

with more complex cases such as those shown in Figure 4 involving large groups of people or objects moving together and interacting for some period of time.
Once the blob-to-track association is performed, we represent each track $i$ at time $t$ by a feature vector

$$f_i^t = \begin{bmatrix} x_i^t & y_i^t & s_i^t & r_i^t & dx_i^t & dy_i^t & v_i^t \end{bmatrix},$$

where $(x_i^t, y_i^t)$ is the centroid of the blob associated with track $i$. $s_i^t$ is the area, in pixels, of the detected blob. $r_i^t$ is the aspect ratio of the blob’s bounding box, calculated by dividing the width of the bounding box by its height. $(dx_i^t, dy_i^t)$ is the unit motion vector for the blob associated with track $i$ at time $t$ compared to track $i$ at time $t - 1$. $v_i^t$ is a temporally smoothed version of the speed of the blob associated with track $i$, calculated as

$$v_i^t = r \frac{\sqrt{(x_i^t - x_{i-1}^t)^2 + (y_i^t - y_{i-1}^t)^2}}{\Delta t} + (1 - r) v_{i-1}^t,$$

where $r$ is a constant (we use $r = 0.5$ in our experiments), $\Delta t$ is the capture time difference between the frames at time $t$ and $t - 1$.

Although Kalman filtering or other methods may be more optimal for filtering blob position and velocity over time, we find that they are unnecessary in practice because we observe relatively little noise in our data, and we map the feature vectors to discrete symbols fairly coarsely in the next step.

### 3.2. Blob Feature Vector Discretization

For simplicity, we use discrete-observation HMMs in this paper. This means that each feature vector $f_i^t$ must be mapped to a discrete category (cluster ID) in the set $V = \{v_1, v_2, \ldots, v_U\}$, where $U$ is the number of categories. We use $k$-means clustering based on a training set to map feature
vectors to discrete clusters. Prior to clustering, we normalize each feature by a $z$-score. The scale factors are independent for all features except $x^t_i$ and $y^t_i$, for which we apply a single isotropic scale factor. In the training stage, we run $k$-means with $U$ cluster centers then save the cluster centers as a code-book for later use. We select $U$ based on the results of a model configuration selection procedure (to be discussed in Section 6.1). At run time, each blob feature vector is mapped to the nearest cluster center according to Euclidean distance then replaced by the corresponding cluster ID. This means that a behavior sequence is finally represented as a sequence of cluster IDs.
3.3. Behavior Clustering

Simply-structured statistical models cannot hope to capture the diversity of “normal” behavior in a given scene — multiple models are required. However, it is difficult to determine how many and what activities will occur in a scene a priori or based on manual monitoring. We therefore propose a method to automatically determine, from a small bootstrap set, the common similar behaviors occurring in a scene.

After blob tracking, we obtain, from a given video, a set of observation sequences describing the motion and appearance of every distinguishable moving object in the scene. We next use the method from our previous work [33], which first uses dynamic time warping (DTW) to obtain a distance matrix for the set of observation sequences then performs agglomerative hierarchical clustering on the distance matrix to obtain a dendrogram.

To determine where to cut off the dendrogram, we traverse the dendrogram in depth-first order from the root and attempt to model the observation sequences within the corresponding subtree using a single linear HMM. If, after training, the HMM is unable to “explain” (in the sense described below) the sequences associated with the current subtree, we discard the HMM then recursively attempt to model each of the current node’s children. Whenever the HMM is able to explain the observation sequences associated with the current node’s subtree, we retain the HMM and prune the tree.

A HMM is said to explain or model a cluster $c$ of observation sequences if there are no more than $N_c$ sequences in $c$ whose average per-observation log-likelihood is less than a threshold $p_c$. We choose $N_c$ using a model configuration selection procedure described in Section 6.1.
To determine the optimal rejection threshold \( p_c \) for cluster \( c \), we use an approach similar to that of Oates et al. [36]. We generate random sequences from the HMM and then calculate the mean \( \mu_c \) and standard deviation \( \sigma_c \) of the per-observation log likelihood over the set of generated sequences. For the lengths of the generated sequences, we simply use the average length of the sequences in the bootstrap set. After obtaining the statistics of the per-observation log likelihood, we let \( p_c \) be \( \mu_c - z\sigma_c \), where \( z \) is an experimentally tuned parameter that gives us convenient control over the probability of making Type I errors in classifying a particular sequence as having been generated by a particular HMM model.

The bootstrapping process results in a set of \( K \) different typical behavior clusters \( \mathcal{C} = \{c_1, c_2, \ldots, c_K\} \) with a set of \( K \) corresponding HMMs \( \mathcal{M} = \{M_1, M_2, \ldots, M_K\} \).

4. Anomaly Detection

In the \textit{supervised} approach, one would construct a training set consisting of anomalous and normal behaviors, build a model, then use the model to classify new behavior sequences as anomalous or normal. This is obviously not suitable when examples of anomalous behavior are sparse or nonexistent. In practical scenarios, the set of possible anomalous behaviors is infinite in its variety, making it very difficult to acquire a representative training set.

In the \textit{unsupervised} approach, one would simply construct a generative model of the normal behavior patterns, then use the model to classify new behavior sequences as abnormal when they are “too far” in some sense from typical behavior. The difficulty with this approach is that there is no clear
Algorithm 2: Anomaly Detection

Input: $O$: behavior sequence
Input: $\mathcal{M}$: set of HMMs

$\mathcal{M}_{ab} \leftarrow \{M \mid M \in \mathcal{M} \text{ and } M \text{ is marked abnormal}\}$

$(M_{ml}, L_{ml}) \leftarrow \text{Find-Most-Likely-Model}(O, \mathcal{M})$

if $M_{ml} \in \mathcal{M}_{ab}$ or $L_{ml} \leq \theta_z$ then
  Alert-Security-Personnel($O$)
end if

way to calibrate the parameters of the “too far” criterion. In practice, one would have to select a conservative initial distance threshold then fine-tune the threshold to achieve the best tradeoff between hit and false positive rates.

We propose a semi-supervised method that calibrates itself using the HMM learning and bootstrap set partitioning method introduced in the previous section. The method is simple. We acquire labels for the bootstrap patterns from the operator, then we apply our behavior clustering algorithm described in Section 3.3 to both the positive and negative sequences in the bootstrap set. We identify each cluster as a “normal” cluster if all of the sequences falling into it are labeled as normal, or identify it as an “abnormal” cluster if any of the sequences falling into it are labeled as abnormal.

Algorithm 2 is a pseudocode summary of the runtime anomaly detection method. New sequences are classified as abnormal if the most likely HMM for the input sequence is associated with one of the abnormal models, or the $z$-scored per-observation log likelihood of the sequence under that most likely model is less than a global empirically determined threshold $\theta_z$.

In principle, if we updated the log likelihood for each model every time we receive a new observation in constant time, the time complexity would be $O(nm)$, where $n$ is the number of observations and $m$ is the number of
models. However, for convenience, we rerun the forward algorithm [37] on each observation for every model.

One problem with this approach is that it never learns any new typical behavior over time. Given a small bootstrap set, the method may work well for a short period of time, but as typical behavior evolves and becomes more diverse, the false alarm rate may increase. We could retrain the system periodically, but this would require too much storage. We therefore need a more adaptive approach that learns new behaviors over time but does not require storing all of the historical data.

5. Incremental Behavior Modeling for Anomaly Detection

In this section, we describe an extension of the basic anomaly detection method described in Section 4 using the incremental maximum likelihood (IML) algorithm for HMMs [38]. The incremental expectation maximization (EM) procedure was originally proposed by Neal and Hinton [39]. The basic idea is to update the sufficient statistics for the parameters of the HMMs as new events occur. In our approach, given an initial set of observation sequence clusters and corresponding HMMs modeling the behavior in the scene, when a new observation sequence arrives, we first determine which cluster it falls into by performing statistical tests on the sequence’s likelihood according to each HMM model. Since likelihood scores for different HMMs cannot be compared directly, we select the most likely model based on the same z-scored per-observation log likelihood measure introduced in Section 3.3 for the initial behavior model clustering.

When the most likely model for a new observation sequence is a model
for a cluster of atypical behaviors, or when the new observation sequence’s likelihood according to the most likely model is not sufficiently high, we raise an alert. If the operator gives feedback that a new observation sequence marked as suspicious by the system is in fact not suspicious, we create a new typical behavior cluster and corresponding HMM for the false alarm sequence. If the likelihood according to the most likely model is sufficiently high for one of the pre-existing models and the operator confirms that the classification is correct, we incrementally update the sufficient statistics for that model; otherwise, we start a new cluster for the sequence, build a new HMM for the new cluster, and label it according to the operator feedback. This approach allows the system to raise alerts when unusual behaviors are detected while adapting to gradual changes in behavior patterns over time. The method is detailed in Algorithm 3.

When a new behavior sequence $O$ arrives, we find the cluster $k$ whose HMM $M_{ml}$ assigns $O$ the highest log likelihood $L_{ml}$. $O$ is considered consistent with $c_k$ if $L_{ml}$ is above threshold $\theta_z$ and the operator does not mark $O$ as a false positive. In this case, we use $O$ to update the sufficient statistics $S^{(k)} = \{\Gamma^{(k)}, \Xi^{(k)}\}$ for cluster $k$, allowing us to throw away the original observation sequence. $\Gamma^{(k)}$ accumulates the $\gamma^{(k)}$ probabilities (defined in Rabiner’s tutorial [37]) needed to estimate the emission and transition probabilities of $M_k$. The update rule is simply

$$\Gamma_{\text{new}}^{(k)} = \alpha \Gamma_{\text{old}}^{(k)} + \gamma^{(k)},$$

where $\alpha$ is a forgetting factor [40]. $\Xi^{(k)}$ accumulates the $\xi^{(k)}$ probabilities needed to estimate the transition probabilities. It is updated the same way
Algorithm 3 Anomaly Detection with Incremental Learning

**Input:** \( O \): behavior sequence  
**Input:** \( M \): set of HMMs  
**Input:** \( S \): set of sufficient statistics  
**Output:** \( \tilde{M} \): set of revised HMMs  
**Output:** \( \tilde{S} \): set of revised sufficient statistics  

\[
\tilde{M} \leftarrow M; \quad \tilde{S} \leftarrow S
\]

\[
\mathcal{M}_{ab} \leftarrow \{ M \mid M \in M \text{ and } M \text{ is marked abnormal} \}
\]

\[
(M_{ml}, S_{ml}, L_{ml}) \leftarrow \text{Find-Most-Likely-Model}(O, M)
\]

\[
d_{\text{feedback}} \leftarrow \emptyset
\]

if \( M_{ml} \in \mathcal{M}_{ab} \) or \( L_{ml} \leq \theta_z \) then  
\[
d_{\text{feedback}} \leftarrow \text{Alert-Security-Personnel}(O)
\]
end if  

if \( L_{ml} > \theta_z \) and \( d_{\text{feedback}} \neq \text{false positive} \) then  
\[
(M, S) \leftarrow \text{Incrementally-Update}(M_{ml}, S_{ml})
\]
\[
\mathcal{M} \leftarrow \{ \mathcal{M} \setminus M_{ml} \} \cup \{ M \}; \quad \tilde{S} \leftarrow \{ \tilde{S} \setminus S_{ml} \} \cup \{ S \}
\]
end if  

else  
\[
(M, S) \leftarrow \text{Create-New-Model}(O)
\]
if \( L_{ml} \leq \theta_z \) and \( d_{\text{feedback}} \neq \text{false positive} \) then  
Mark \( M \) as abnormal
end if
\[
\mathcal{M} \leftarrow \mathcal{M} \cup \{ M \}; \quad \tilde{S} \leftarrow \tilde{S} \cup \{ S \}
\]
end if

\( \Gamma^{(k)} \) is. In the experiments reported upon in this paper, we use \( \alpha = 0.9 \).

When an HMM is first trained, we initialize the sufficient statistics \( S^{(k)} \) over all of the observation sequences in cluster \( c_k \) using batch training. During incremental learning, in each iteration of the E-step, we calculate the sufficient statistics \( S^{(k)} \) for the single input observation sequence then update \( S^{(k)} \) accordingly to be used in the M-step. We repeat the E and M updates for a fixed number of iterations \( I \). Neal and Hinton [39] prove monotonic convergence of incremental EM under certain circumstances and find that in practice it is much faster than the batch EM algorithm. In our case, since we
Algorithm 4 Incremental EM Algorithm

Input: \( O \): behavior sequence
Input: \( M \): HMM model
Input: \( S = \{\Gamma, \Xi\} \): set of sufficient statistics
Output: \( M^* \): revised HMM model
Output: \( S^* = \{\Gamma^*, \Xi^*\} \): set of revised sufficient statistics

\[
M^* \leftarrow M
\]
for \( i = 1 \rightarrow I \) do
  \{E-step\}
  \((\gamma, \xi) \leftarrow \text{Compute-Sufficient-Statistics}(O, M^*)\)
  \(\Gamma^* \leftarrow \alpha \Gamma + \gamma; \quad \Xi^* \leftarrow \alpha \Xi + \xi\)
  \{M-step\}
  \(M^* \leftarrow \text{Reestimate-Model-Parameters}(M^*, \Gamma^*, \Xi^*)\)
end for

are incorporating a completely new observation sequence at each increment, the likelihood over all the sequences may increase or decrease. The method is summarized in Algorithm 4.

6. Experimental Results

We recorded video\(^1\) from the scene in front of a building during working hours (9:00–17:00) for one week. We started a new event whenever the number of foreground pixels exceeded 300. We used a threshold of 0.995 for shadow detection NCC. We discarded any blobs smaller than 300 pixels in area. We obtained 423 video segments containing 660 observation sequences.

Figure 5 shows examples of four common activities: people entering the building, people leaving the building, people parking bicycles, and people riding bicycles out. For the purpose of analyzing results, we manually la-

\(^1\)Freely available for others to experiment with at http://www.kanouivirach.com/#downloads.
beled each of the videos with the “normal” categories “walk-in,” “walk-out,” “cycle-in,” and “cycle-out,” or, for other activities such as walking around looking for an unlocked bicycle, with the “abnormal” category.

We performed experiments in three parts: model configuration selection (finding an optimal set of HMMs to model the bootstrap set), anomaly detection based on the HMM bootstrap set, and anomaly detection with incremental learning.

6.1. Model Configuration Selection

Towards model identification, we performed five-fold cross validation with different bootstrap parameter settings and selected the configuration with the highest accuracy in separating the normal sequences from the abnormal sequences on the bootstrap set, as measured by the false positive rate for abnormal sequences over all cross validation test folds. (Every bootstrap cluster containing an abnormal sequence is considered abnormal, so we always obtain 100% detection on the bootstrap set; the only discriminating factor is the false positive rate.) The set of parameters that we varied were 1) the number of states in each bootstrap HMM (3–8), 2) the number of symbols/k-means clusters (3–8), and 3) the number $N_c$ of deviant patterns allowed in a bootstrap cluster (1–10). To find the distribution (parameters $\mu_c$ and $\sigma_c$) of
the per-observation log likelihood for a particular HMM, we always generated 1,000 sequences of average length (the average length was computed over the bootstrap set; it was 205 observations in our experiment). For the rejection threshold \( p_c \), we used a \( z \)-threshold of 2.0 (i.e., \( p_c = \mu_c - 2\sigma_c \)), corresponding to a Type I error rate of 0.0228.

The setting of the \( z \)-score threshold is important. With a \( z \)-score of 2, on different runs, we find that the method generates 15–30 clusters from our bootstrap set, usually perfectly classifies the bootstrap set, and performs well on the test set. With a lower threshold (0.5 to 1.5), only extremely similar series are clustered together, leading to a large number of clusters and a large number of false positives on typical behaviors that vary only slightly from what has been seen in the bootstrap set. With higher thresholds (2.5 to 3), patterns that are quite dissimilar get clustered, oftentimes leading to very few clusters that do not cleanly separate the bootstrap set and have low accuracy on the test set.

To reduce the variance in the measured false positive rate due to random initialization, we ran the entire cross validation procedure three times and averaged the results.

Based on the false positive rate criterion described above, we selected the model configuration consisting of five states, seven symbols, and \( N_c = 6 \) and trained a new HMM ensemble on all 150 bootstrap sequences. We repeated the training process several times until we obtained a model that obtained perfect accuracy on the bootstrap set. Table 1 shows the distribution of bootstrap sequences across this model’s 17 behavior clusters. We used this model with the remaining 510 sequences in the anomaly detection experiments.
Table 1: Example human behavior pattern bootstrapping results. We used linear HMMs with five states and seven symbols. The model consists of 17 clusters. “W” means “walk” and “C” means “cycle.” For the six clusters containing more than one sequence, shown is the distribution of the patterns in the cluster over the activities. The last row shows the distribution of the 11 clusters containing only a single sequence over the activity categories.

6.2. Anomaly Detection (Batch Processing)

Here we evaluate the ability of our anomaly detection method to identify anomalous events in the data not included in the bootstrap set. Since the main concern in video surveillance is to detect every unusual event while minimizing the false positive rate, we calculate an ROC curve and select the detection threshold yielding the best false positive rate at a 100% hit rate. We reported preliminary results for the proposed method compared to traditional machine learning algorithms in a conference paper [41].

In this experiment, we compare the proposed method against HMM-based methods using alternative representations and scoring methods similar to those of Xiang and Gong [32]. This allows us to determine the extent to which different event representations (our local representation or Xiang and Gong’s global representation) and different scoring methods (our \( z \)-scoring method or the standard likelihood ratio test (LRT)) contribute to anomaly
Figure 6: Example anomaly detected by the proposed method (Method I). The sequence contains a person walking around looking for an unlocked bicycle.

detection performance on our test set. The four methods are as follows:

1. Method I: The proposed method.


3. Method III: An HMM-based method using a global event representation similar to that of Xiang and Gong, with our z-scored likelihood method.


For method I and II, we used the 150-sequence bootstrap sequence set from Section 6.1 as training data and tested on the remaining 510 sequences. Figure 6 shows an example of a sequence classified as abnormal. When evaluating methods III and IV, we extract global features rather than individual blob features. The local method considers the single blob motion at a time, whereas the global method simultaneously considers multiple blobs appearing at the same time. The data set for the global method comprises 401 sequences (366 negatives and 35 positives). We used the first 150 sequences (139 negatives and 11 positives) for the bootstrap set and the remaining 251
Figure 7: Anomaly detection ROC curves.

<table>
<thead>
<tr>
<th>Method</th>
<th>TP</th>
<th>FP</th>
<th>TN</th>
<th>FN</th>
<th>TPR</th>
<th>FPR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local (z-scoring)</td>
<td>24</td>
<td>42</td>
<td>444</td>
<td>0</td>
<td>1</td>
<td>0.086</td>
</tr>
<tr>
<td>Local (LRT)</td>
<td>24</td>
<td>486</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Global (z-scoring)</td>
<td>24</td>
<td>217</td>
<td>10</td>
<td>0</td>
<td>1</td>
<td>0.956</td>
</tr>
<tr>
<td>Global (LRT)</td>
<td>24</td>
<td>223</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0.982</td>
</tr>
</tbody>
</table>

Table 2: Anomaly detection results for the local method with the z-scoring method and the likelihood ratio test, and the global method with the z-scoring method and likelihood ratio test.

sequences (227 negatives and 24 positives) for the test set. In all four conditions, we performed a separate five-fold cross validation to select the best model configuration as described in Section 6.1.

Figure 7 shows ROC curves for methods I–IV. The solid red line represents the ROC curve for method I as we vary the threshold (z-score or likelihood ratio) at which a sequence is considered anomalous. Note that the ROC does not intersect the point (0, 0) because any sequence that is most likely under one of the HMMs modeling anomalous sequences in the bootstrap set is
automatically classified as anomalous regardless of the threshold. The ROC reveals that a z-score per-observation log likelihood $\theta_z$ of $-2.293$ achieves zero false negatives at a false alarm rate of 0.086. The dotted green line is the ROC curve for method II, the dashed blue line represents method III, and the dash-dot magenta line represents method IV. Table 2 shows the detailed performance of each detection method at 100% hit rate.

The ROC curves and FP rates show that our method clearly outperforms the other three methods, but there is also a surprising interaction: for the global method, z-scored likelihood and LRT are equally effective, but for our local event based method, z-scoring is much more effective than the LRT. This may reflect that the LRT’s use of a mixture model is less appropriate for the local method than the global method.

The reason that the local method is better than the global method (albeit when combined with z-scoring of the likelihood) is that many of the abnormal sequences in the test set tend to be locally similar to the abnormal sequences in the bootstrap set but globally different. The local method treats concurrent but spatially separate local events as being independent, whereas the global method attempts to construct a joint model over the entire scene. The global method might be improved with a larger bootstrap set.

The log likelihood ratio test fails to detect completely new anomalous patterns when the pattern has a very low likelihood according to the abnormal model and the normal model but has a slightly higher likelihood according to the normal model. In order for the LRT test to detect such patterns, we need to adjust the LR threshold, causing higher false positives (increasing the number of false positives to about 30–40). Since our approach creates
a new abnormal model for any low-likelihood observation, regardless of the relative likelihood, it does not suffer from this problem.

6.3. Anomaly Detection with Incremental Learning

In this experiment, we evaluate the use of incremental HMM learning for anomaly detection. We used the same settings and model configuration as in the batch method experiments described in Section 6.2. Since we incorporate a completely new observation sequence at each increment, the likelihood over all the sequences may increase or decrease. In the current work, we reestimate $\mu_c$ and $\sigma_c$ after every update by generating 1,000 new sample sequences. The update process could obviously be optimized.

Overall results of incremental learning are shown in Table 3. Compared to the batch method, the incremental version of the proposed method (local method with $z$-scoring) obtains a nearly four-fold decrease in the false positive rate (2.2% vs. 8.6%) at the cost of a 9% decrease in the hit rate. The results shown in the table are based on a fixed $z$-score threshold of $\theta_z = 2.0$. To better enable comparison of the results for batch and incremental learning, we varied $\theta_z$ to obtain false positive rates at 100% detection rates and to obtain equal error rates (EERs). The incremental method achieves a 3.7% false alarm rate at a 100% hit rate, compared to 8.6% for the batch method. It achieves an EER of 95.8%, compared to 91.8% for the batch method.

Overall, these results demonstrate that the incremental algorithm detects anomalies more effectively with lower false alarm rates. It is better able than the batch method to learn new forms of normal behavior. New behavior classes are always called anomalous at first, so completely new behaviors (suspicious or not) will always trigger a false positive and lead to the creation
Figure 8: Comparison of batch learning (dotted red line) and incremental learning (solid green line) over time. (a) Cumulative false positive rates. (b) Cumulative true positive rates. Each point is an average of the false positive or true positive rates over 5 trials.

of a new model trained on the new sequence. These one-sequence HMMs tend to generate sequences with very low variance in the log likelihood. Such models do not often get updated with new sequences, but they do from time to time. In this experiment, new sequences were assigned to existing multi-sequence HMMs 466 times, existing single-sequence HMMs 44 times, and were used to create new models 5 times. The improvement in performance of the incremental method over the batch method thus comes first from tuning the parameters of the existing behavior classes (both typical and anomalous), but it also comes from the use of additional typical labels for the false positive anomalous series.

To further understand the performance differences between the batch and incremental methods, we partitioned the test data sequentially into chunks and computed the false and true positive rates for each chunk of observations. Initially, we set the chunk size to 10. If our system detects a suspicious
pattern within a chunk, we record it and compute the false positive rate for the chunk. If the method does not detect any suspicious pattern, on the other hand, we increase the chunk’s size by 1 until a true positive is given, then we reset the next chunk’s size back to 10. This resulted in 22 chunks for our test set. Cumulative false and true positive rate data are shown in Figure 8. Initially, the incremental method has the same false alarm rate as the batch method. However, the incremental method’s false alarm rate drops quickly to achieve the eventual 2.2% false alarm rate at a 91% hit rate.

For comparison, we also tested incremental versions of the three alternative approaches previously described in Section 6.2: local method with LRT, global method with z-scoring, and global method with LRT. In every condition, we used the same model configuration and detection threshold found best in the batch experiment. For our incremental approach, both typical and atypical behavior models get updated during incremental learning. The results are shown in Table 3. Similar to the batch method results, none of the three alternative methods performed as well as the proposed method. Our experimental results also demonstrate that over 510 new patterns, our approach requires security personnel to correct the system a mere 9–10 times.

The incremental version of the global method has lower false alarm rates

<table>
<thead>
<tr>
<th>Incremental method</th>
<th>TP</th>
<th>FP</th>
<th>TN</th>
<th>FN</th>
<th>TPR</th>
<th>FPR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local (z-scoring)</td>
<td>21.8</td>
<td>9.8</td>
<td>476.2</td>
<td>2.2</td>
<td>0.91</td>
<td>0.022</td>
</tr>
<tr>
<td>Local (LRT)</td>
<td>16</td>
<td>243.4</td>
<td>212.6</td>
<td>8</td>
<td>0.6664</td>
<td>0.564</td>
</tr>
<tr>
<td>Global (z-scoring)</td>
<td>3</td>
<td>13</td>
<td>214</td>
<td>21</td>
<td>0.125</td>
<td>0.057</td>
</tr>
<tr>
<td>Global (LRT)</td>
<td>18</td>
<td>184.4</td>
<td>42.6</td>
<td>6</td>
<td>0.75</td>
<td>0.813</td>
</tr>
</tbody>
</table>

Table 3: Anomaly detection results on average for incremental HMM learning for our local method and the global method over 5 trials.
and lower detection rates than the batch version of the global method. To an extent, the fixed z-score threshold of 2.0 places the global method at an inopportune position on the ROC curve. However, generally, we find that the global method with z-scoring performs poorly regardless of the threshold. With the global representation, the abnormal models tend to be quite specific whereas the normal models tend to be a little more general, so that we often see new anomalous patterns receiving relatively good scores from some normal models but no good scores from abnormal models. The anomalies thus tend to get incorrectly classified as normal more often.

Besides improved detection, another advantage of our incremental approach is that it is more efficient than the batch method over time. To demonstrate this, we performed a brief experiment on the runtime requirements on a Core 2 Duo processor at 2.2 GHz. We first trained a model with 1,000 synthetic sequences then re-trained that model with 1,000 sequences plus 10 new sequences using batch learning or incremental learning. The batch method required 33816 ms, whereas our incremental method with only the 10 new sequences required only 1056 ms. Obviously, the incremental approach uses far less time than the batch method to integrate new data into the training set, because it does not require retraining on old data.

As a final manipulation, we performed a brief experiment to determine whether incremental learning alone without a bootstrap set is effective on these data. We found that test set performance improves with the size of the bootstrap set. With more bootstrap data we tend to discover more compact behavior classes, leading to better performance on incremental updates. Conversely, incremental updates without a good bootstrap model lead to
poor detection rates throughout the experiment. So while a no-bootstrap method could lessen the user’s up-front workload, we conclude that the risk of missed anomalies is too high.

7. Discussion and Conclusion

We have proposed and evaluated an efficient method for bootstrapping scene-specific anomalous human behavior detection systems that incrementally learns behavior models without requiring storage of large databases of training data. The method requires minimal involvement of a human operator; the only required action is to label the patterns in a small bootstrap set as normal or anomalous and then to label false positive alarms as normal when they occur.

On a testbed real-world data set, with 150 bootstrap sequences, the incremental method achieves a false positive rate of 2.2% at a 91% hit rate. With threshold tuning, the method can yield an equal error rate of 95.8% or a false positive rate of 3.7% at a 100% hit rate. The experiments show that it is possible to learn a complex set of varied behaviors occurring in a specific scene with a collection of simple HMMs while allowing evolution of the learned typical behavior model over time. This can lead in turn to more effective anomaly detection. Deploying our system on a large video sensor network would potentially lead to substantial increases in the productivity and proactivity of human monitors.

There are a few limitations to our current method. One is that the blob tracking process is not robust for scenes with dense crowds. However, it would work well for most building entrances, office building hallways, and similar
environments. Also, constructing a fixed codebook to quantize the feature space may be inappropriate for incremental approaches since the codebook would need to be revised over time to account for changing “typical” behaviors. Lastly, our current system can add new HMMs but cannot remove them. Over time, the number of HMMs would grow without bound.

In future work, we plan to address these limitations. We will also explore various means to improve the system’s performance. It may be better to take a probabilistic generative approach for the assignment of feature vectors to discrete categories rather than making hard assignments. It would be better to periodically merge similar models or remove old models that no longer represent typical behavior. Since each HMM has the same structure, it would be very straightforward to check for pairs of similar models and merge them.

Our approach can be extended to larger-scale situations as long as typical behavior can be modeled in terms of the spatio-temporal motion of foreground blobs. It could also be extended to recognize more complex events involving multiple persons — we believe that we can handle interactions between pedestrians, for example in pickpocket and assault events, within the current framework of temporal statistical models for individual humans by including observation features that characterize a person’s interaction with others while in the scene. Integrating with pedestrian detection and tracking methods that rely on body part detection and tracking rather than motion blob tracking might help capture a larger proportion of the kinds of unusual behaviors we would observe in building entrances or office hallways.

We have shown that local event processing is more effective than global event processing on our data set. It would be interesting to compare to
a hybrid approach in which we apply the global method to blocks in an image rather than the whole image. Finally, we will explore integrating the algorithms into a complete video surveillance system such as ZoneMinder [42].

Acknowledgments

This work was partly supported by a grant from the Royal Thai Government to MND and by graduate fellowships from the Royal Thai Government and AIT to KO and SG. We thank the AIT Vision and Graphics Lab for valuable discussions and comments. We also thank our peer reviewers, who provided extensive constructive feedback on previous versions of the paper.

References


