AUTOMATIC SUSPICIOUS BEHAVIOR DETECTION FROM A SMALL BOOTSTRAP SET

Kan Ouivirach, Shashi Gharti, and Matthew N. Dailey
School of Engineering and Technology, Asian Institute of Technology, Thailand
kan@ieee.org, shashi.gharti@gmail.com, mdailey@ait.ac.th

Keywords: Hidden Markov Models, Behavior Clustering, Anomaly Detection, Bootstrapping

Abstract: We propose and evaluate a new method for automatic identification of suspicious behavior in video surveillance data. The approach works by constructing scene-specific statistical models explaining the behaviors occurring in a small bootstrap data set. It 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. In an evaluation on a real-world testbed video surveillance data set, the method proves extremely effective, with a false alarm rate of 7.4% at a 100% hit rate. 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.

1 INTRODUCTION

Video surveillance is ubiquitous in our lives, but the ongoing proliferation of surveillance cameras makes it increasingly difficult to monitor all channels continuously. As the amount of surveillance video increases, monitoring becomes more expensive and less effective. Security would be enhanced if it were possible to perform intelligent filtering of typical events and to automatically bring suspicious events to the attention of human security personnel.

The goal of automated suspicious event detection, however, requires some level of human behavior understanding. Many researchers have attempted to build intelligent systems able to interpret and understand human behaviors (Cao et al., 2004; Davis et al., 1998; Du et al., 2006; Li et al., 2006; Masoud and Papanikolopoulos, 2003; Smichisescu et al., 2005; Wang et al., 2006). Pre-trained hidden Markov models (HMMs) and other dynamic Bayesian networks such as conditional random fields (CRFs) have been widely used in this area, ranging from visual activity recognition (Smichisescu et al., 2005; Vail and Guerstein, 2007; Yamato et al., 1992), gesture recognition (Gao et al., 2004; Wilson and Bobick, 2000), and unusual activity detection (Andrade et al., 2006; Chan et al., 2004; Nair and Clark, 2002; Snoek et al., 2006; Zhang et al., 2005). However, the problem is difficult and remains unsolved, due to the large amount of variability within any particular activity. In the context of video surveillance, the problem is even more challenging because behavior considered normal in one scene might be considered unusual in another scene.

One limitation of much of the existing work is that it manually creates separate models for each distinct a priori known class of normal behavior. One example is the work of Nair and Clark (2002), which performs automated video surveillance using HMMs, each modeling a common, pre-defined activity in a scene. They classify a sequence as anomalous if the log likelihood of the sequence is below a pre-defined threshold for all trained HMMs. Other work (Arsić et al., 2007; Lee et al., 2003; Wu et al., 2005) uses support vector machines (SVMs) to model and classify behaviors into pre-defined classes.

In more recent years, research has started to focus on the problem of unsupervised analysis and clustering of behaviors in a particular scene for a variety of purposes including anomaly detection, surveillance, and classification. Zhong et al. (2004) treat video segments as documents and cluster the documents based on co-occurrence information. Li et al. (2006) cluster human gestures by constructing an affinity matrix using dynamic time warping (DTW) (Sakoe, 1978), then they apply the normalized-cut algorithm to cluster the gestures. Hautamäki et al. (2008) apply DTW and use the resulting pairwise DTW distances as input to a hierarchical clustering process in which k-
means is used to fine-tune the output. Swears et al. (2008) propose hierarchical HMM-based clustering to find and cluster motion trajectories and velocities for surveillance video in a highway interchange scene. Xiang and Gong (2005) model the distribution of activity data in a scene using a Gaussian mixture model (GMM) and employ the Bayesian information criterion (BIC) to select the optimal number of behavior classes prior to HMM training.

In this paper, we propose a method for automatic suspicious behavior detection that utilizes a small bootstrap set in which observation sequences are manually labeled as normal or suspicious (warranting an operator’s attention). We partition the bootstrap set into clusters of similar sequences, model each cluster with a simple HMM, and then label each behavior cluster as normal or possibly suspicious based on the labels of the individual sequences mapped to the cluster. After bootstrapping is complete, we assign new observation sequences to behavior clusters using statistical tests on the log likelihood of the sequence according to the corresponding HMMs. A sequence is considered suspicious if the most likely cluster’s HMM is too low. The cluster-specific likelihood threshold is learned rather than set arbitrarily. In an evaluation on a real-world video surveillance situation, we find that, based on a bootstrap set of 150 human motion sequences, our method is extremely effective at identifying suspicious behavior, with a false positive rate of 7.4% at a hit rate of 100%.

Our 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.

In the rest of this paper, we provide the details of our human behavior modeling and bootstrapping algorithm in Section 2 and our anomaly detection method in Section 3. We demonstrate the feasibility of the algorithm with an experimental evaluation in Section 4, and then conclude and point to future work in Section 5.

2 BEHAVIOR MODEL BOOTSTRAPPING

Our method for bootstrapping a model of the specific behaviors in a scene is a batch procedure based on a training video stream acquired over a short period of time such as one week. We first perform moving blob tracking on the training video on the assumption that moving blobs of sufficient size are people or groups of people. For each blob, we extract a sequence of feature vectors describing the blob’s trajectory and appearance over time. From these data, we automatically bootstrap a bank of linear HMMs, each model specializing in one type of behavior. In the following sections, we describe each of these steps in more detail.

2.1 Global Motion Detection

To decrease processing and storage time, we discard any video frames in which no motion occurs. We simply subtract each frame from the previous frame and then determine if the difference image has the number of pixels whose intensity differences are above a threshold. We also buffer the no-motion frames for a period of time and include some number of frames before and after the motion in the event. This method avoids oversegmentation of events when moving objects stop moving briefly.

2.2 Blob Extraction

After discarding the no-motion video segments, we use the background modeling technique proposed by Poppe et al. (2007) to separate the moving foreground pixels from the background. This method extends the standard mixture of Gaussians background model (Stauffer and Grimson, 1999) to handle gradual illumination changes. In addition, we use normalized cross correlation (NCC) to eliminate shadows cast by moving objects. Sample results from the foreground extraction and shadow removal procedures are shown in Figure 1.

To remove noise and connect foreground regions, we apply morphological opening then closing operations to obtain the connected components. We then filter out any components whose size is below threshold. We finally represent each blob (connected fore-

![Figure 1: Sample foreground extraction and shadow removal results. (a) Original image. (b) Foreground pixels according to background model. (c) Foreground pixels after shadow removal.](image-url)
ground component) \( i \) at time \( t \) by the feature vector
\[
\vec{f}_t^i = [x'_t, y'_t, s'_t, r'_t, dx'_t, dy'_t, v'_t],
\]
where \([x'_t, y'_t]\) is the centroid of the blob, \( s'_t \) is the size of the blob in pixels, \( r'_t \) is the aspect ratio of the blob’s bounding box, \([dx'_t, dy'_t]\) is the unit-normalized motion vector for the blob compared to the previous frame, and \( v'_t \) is the blob’s speed compared to the previous frame, measured as
\[
v'_t = \sqrt{(x'_t - x'_{t-1})^2 + (y'_t - y'_{t-1})^2} / \Delta t,
\]
where \( \Delta t \) is the capture time difference between the frames at time \( t \) and \( t-1 \). We increase the stability by using an average velocity, measured as
\[
v'_t = rv'_t + (1-r)v'_{t-1},
\]
where \( r \) is a constant. We use \( r = 0.5 \) in our experiments.

### 2.3 Blob Feature Vector Discretization

For simplicity, we use discrete-observation HMMs in this paper. This means that each feature vector \( \vec{f}_t^i \) 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.

In the training stage, we run \( k \)-means with \( U \) cluster centers then save the cluster centers as a codebook for later use. 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. Therefore, a behavior sequence is finally represented as a sequence of cluster IDs.

To prevent differing numeric scales of the features from affecting the distance metric, we first normalize each feature independently by \( z \)-scaling to a mean of 0 and standard deviation of 1 over the training set. For the vectors \((x'_t, y'_t)\), rather than normalizing the \( x \) and \( y \) components independently, we use a common isotropic scale factor for the two dimensions to avoid overemphasizing small deviations from typical trajectories in directions without much deviation in the training data.

Currently, we empirically tune the free parameters (frame buffer length, thresholds, and number of \( k \)-means clusters) to the training data. We hope to automate the parameter selection process in future work.

In the rest of this paper, when we mention the term “observation sequence,” we mean a sequence of cluster IDs. The outputs to the next step are a list of blob feature vectors with the corresponding cluster IDs and bounding boxes, and the current frame and foreground mask.

### 2.4 Appearance-Based Blob Tracking

In this step, we link blobs in successive images to create distinct tracks, i.e., sequences of blob descriptors, for each moving object in the scene. The process is simple when there is only one moving object in the scene at any given time, but when multiple moving objects are present, they can occlude each other so that a single motion blob may contain multiple moving people or objects, and over time, blobs may merge and split. Here we describe our approach to maintaining the identity of moving objects through blob splits and merges.

The inputs to our tracker at time \( t \) are the set of tracks from time \( t-1 \), a list of blob feature descriptors with corresponding cluster IDs and bounding boxes, and the current image and foreground mask. The tracker’s goal is to update the track list by creating new tracks for newly appeared objects, updating old tracks with new blob information, and deleting tracks for objects that have left the scene.

We use a forward and backward overlap method to associate blobs in the current frame with the tracks from the previous frame. We first find the amount of bounding box overlap between each new blob and each existing track and build a correspondence matrix indicating the size of the area of overlap between each blob and each track. Blobs corresponding to isolated moving objects are associated with unique tracks. Under normal circumstances, when no merge or split occurs, each blob will either match one of the existing tracks or be classified as new, in which case a new track will be created. We create new tracks for newly appearing isolated blobs and destroy tracks that are inactive for some number of frames. This approach is similar to the work of Senior et al. (2006). 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). We use the color coherence vector (CCV) (Pass et al., 1996) as an appearance model to handle these cases. The CCV combines color histogramming with spatial information and helps to distinguish different textures with similar colors. When tracks merge, we group them, but keep their identities separate, and when tracks split, we attempt to associate the new blobs with the correct tracks or groups of tracks by comparing their CCVs. When blob \( i \) is found to correspond to a single unique track, we use \( \vec{f}_t^i \) and the blob’s CCV to update that track. For each track set representing a series of blob merges, however, we only update the grouped tracks’ centroids to be the center of mass of the set of blobs associated with the track set. For the other features (size, aspect ratio, and so on) and the
Figure 2: Sample blob tracking results for typical simple cases. In frame 94, track 0 and track 1 have merged. In frame 103, our method has correctly associated the each motion blob with the correct tracks after splitting. Similarly, frame 110 shows the result of another merge, and frame 116 shows the (correct) result when the merged motion blob splits.

Figure 3: Sample blob tracking results for a complex case. In frame 131, track 1 and track 2 have merged. In frame 133, our method has correctly associated the current motion blobs with tracks after the merged blob splits. However, when the merged blob shown in frames 141 and 151 splits (in frame 160), we observe a track ID switch error in associating blobs with tracks.

CCV, we preserve the track’s historical values, on the assumption that when a merged object eventually separates from its group, its appearance will be similar to its appearance before the merge. This approach performs very well on typical simple cases such as those as shown in Figure 2, but it can make mistakes with more complex cases such as the merged blob split at frame 160 shown in Figure 3. In future work we plan to further improve the split and merge processing method to handle more complex cases.

2.5 Behavior Clustering

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 partition the observation sequences into clusters of similar behaviors then model the sequences within each cluster using a simple linear HMM. We use the method from our previous work (Ouivirach and Dailey, 2010), 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 dendogram (a binary tree expressing the similarity structure within the set of observation sequences). To determine where to cut off the dendogram, we traverse the DTW dendogram in depth-first order from the root and attempts 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 cluster $c$ whose per-observation log-likelihood is less than a threshold $p_c$. We use $N_c = 10$ in our experiments. The per-observation log likelihood of a sequence $\hat{O}_i = \{O_{i1}, O_{i2}, \ldots, O_{iT_i}\}$ is

$$L_i = \log P(\hat{O}_i | M_c),$$

where $M_c$ is the HMM that models the sequences in cluster $c$, $T_i$ is the number of observations in sequence $i$, and $P(\hat{O}_i | M_c)$ is calculated using the forward algorithm (Rabiner, 1989).

The clustering 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 $\mathcal{M} = \{M_1, M_2, \ldots, M_K\}$.

3 ANOMALY DETECTION

Here we describe our method for anomalous or suspicious behavior detection. In the 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.

The pure supervised approach is obviously not suitable, however, 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 unsupervised approach, on the other hand, we 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. For example, we could use the algorithm described in Section 2 to construct a bank of HMMs explaining the normal sequences in a training set of patterns, then classify new sequences as abnormal or suspicious whenever the likelihood of the sequence according to the most likely HMM is below some specific threshold.

The difficulty with the pure unsupervised approach, however, is that there is no clear 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. An inappropriate initial cutoff could lead to disastrous misses of suspicious behavior or inundation with false positives.

Since both approaches have limitations, rather than the pure supervised method or the pure unsupervised method, we instead propose a semi-supervised method that self-calibrates itself from a small bootstrap set in which each bootstrap sequence is manually labeled as normal or suspicious by a human operator. Our method is simple. We acquire labels for the bootstrap patterns from the operator, then we apply the algorithm of Section 2 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. New sequences are classified as normal if the most likely HMM for the input sequence is associated with a cluster of normal sequences and the z-scaled per-observation log likelihood of the sequence under that most likely model is greater than a global empirically determined threshold $\zeta$.

4 EXPERIMENTAL RESULTS

We created a testbed data set using a CCTV camera with a view of the front of a building, as seen in Figure 4(a). Videos were recorded at 25 frames per second with a resolution of $320 \times 240$ during working hours (9:00–17:00) for one week. We used the previously-described techniques to segment the raw video stream into separate videos containing motion, track blobs, extract features, discretize the feature vectors, and create observation sequences. We obtained 423 video segments containing 660 observation sequences.

We observed that there are four common activities in this scene: people walking into the building, people walking out of the building, people parking bicycles, and people riding bicycles out. Figure 4 shows examples of each of these behaviors. In our experiments, we considered all other behaviors, such as walking around looking for an unlocked bicycle, to be suspicious or abnormal.

To evaluate the effectiveness of our method, we divide the experiments into two parts: model configuration selection (finding an optimal set of HMMs to model the bootstrap set) and anomaly detection based on the HMM bootstrap set. In all of the experiments, we used linear HMMs. We chose this model structure based on our previous empirical experience (Ouivirach, 2006).

4.1 Model Configuration Selection

We first manually labeled each of the videos with the categories “normal” or “abnormal.” For purposes of analyzing the results, we further subdivided the normal patterns into categories “walk-in,” “walk-out,” “cycle-in,” and “cycle-out,” but we did not use these labels for model selection or learning.

Towards model identification, we performed a series of experiments 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 sequence set, as measured by the false positive rate for abnormal sequences. 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 (4–8), 2) the number of sequences used for bootstrapping (50–200), and the number of tokens used (4–8). To find the distribution (parameters $\mu_c$ and $\sigma_c$) of the per-observation log likelihood for a particular HMM, we always generated 1000 sequences of 150 observations then used a $z$-threshold of 2.0. We fixed the parameter $N_c$ (the number of deviant patterns allowed in a cluster) to 10.

A subset of the results of the model configuration selection experiments are shown in Figure 5.

Based on the false positive rate criterion described above, we selected the model configuration consisting of HMMs with five states and seven tokens trained on 150 bootstrap sequences.

We arbitrarily chose the model from one of the 10 trials with this configuration. Table 1 shows an example of the distribution of bootstrap sequences across behavior clusters with the selected model configuration. In this run, our method obtained perfect accuracy with 20 behavior clusters. We used this model with the remaining 510 sequences in the anomaly detection experiments described next.

Table 1: Example human behavior pattern bootstrapping results. We used linear HMMs with five states and seven tokens. The model consists of 20 clusters. “W” means “walk” and “C” means “cycle.” For the seven 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 13 clusters containing only a single sequence over the activity categories.

<table>
<thead>
<tr>
<th>Cluster #</th>
<th>W-in</th>
<th>W-out</th>
<th>C-in</th>
<th>C-out</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>44</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>37</td>
<td>0</td>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>One-seq clusters</td>
<td>0</td>
<td>2</td>
<td>6</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Figure 6: Anomaly detection ROC curves. Red, green and blue lines represent ROCs for the proposed method in Experiment I, PCA-based anomaly detection in Experiment III, and SVM-based anomaly detection in Experiment IV, respectively.

4.2 Anomaly Detection

Here we describe four experiments to evaluate our anomaly detection method. In Experiment I, we applied our proposed method, as previously described, to detect anomalous events. In Experiments II, III and IV, for comparison, we applied traditional machine learning algorithms to the same problem. We use $k$-nearest neighbors ($k$-NN), principal components analysis (PCA), and support vector machines (SVMs). For each method, we use the 150-sequence bootstrap sequence set from Section 4.1 as training data and test on the remaining 510 sequences. Since the main concern in video surveillance is to detect every unusual event while minimizing the false positive rate, in every experiment, we calculate an ROC curve and select the detection threshold yielding the best false positive rate at a 100% hit rate. Our experimental hypothesis was that the proposed method for modeling scene-specific behavior patterns should obtain better false positive rates than the traditional methods.

4.2.1 Experiment I: Proposed method

The red line in Figure 6 represents the ROC curve for our method as we vary the likelihood threshold 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 threshold of $-3.259$ achieves zero false negatives at a false alarm rate of 0.074. Table 2 shows the detailed performance of this model, and Figure 7 shows an example of a sequence classified as abnormal (a person walking around looking for an unlocked bicycle).

4.2.2 Experiment II: $k$-NN

In this experiment, we tested the ability of a more traditional machine learning algorithm, $k$-nearest neighbors, to detect the anomalies in our testbed data set.
Figure 5: Subset of model configuration selection results. Model configuration with (a) five tokens, (b) seven tokens and (c) nine tokens. Red, green and blue lines represent models with five, six and seven states, respectively. Each point is an average over 10 trials.

Figure 7: Example anomaly detected by the proposed method in Experiment I. The sequence contains a person walking around looking for an unlocked bicycle.

Table 2: Anomaly detection results for the proposed method, \(k\)-NN, PCA, and SVM in Experiment I, II, III, and IV, respectively. For PCA, we include 11 abnormal sequences from the bootstrap set in the test set, so the total number of positives is 35.

<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>Ours</td>
<td>24</td>
<td>36</td>
<td>450</td>
<td>0</td>
<td>1</td>
<td>0.074</td>
</tr>
<tr>
<td>1-NN</td>
<td>19</td>
<td>1</td>
<td>485</td>
<td>5</td>
<td>0.792</td>
<td>0.002</td>
</tr>
<tr>
<td>2-NN</td>
<td>19</td>
<td>2</td>
<td>484</td>
<td>5</td>
<td>0.792</td>
<td>0.004</td>
</tr>
<tr>
<td>3-NN</td>
<td>16</td>
<td>0</td>
<td>486</td>
<td>8</td>
<td>0.667</td>
<td>0</td>
</tr>
<tr>
<td>4-NN</td>
<td>16</td>
<td>1</td>
<td>485</td>
<td>8</td>
<td>0.667</td>
<td>0.002</td>
</tr>
<tr>
<td>5-NN</td>
<td>14</td>
<td>0</td>
<td>486</td>
<td>10</td>
<td>0.583</td>
<td>0</td>
</tr>
<tr>
<td>PCA</td>
<td>35</td>
<td>421</td>
<td>65</td>
<td>0</td>
<td>1</td>
<td>0.87</td>
</tr>
<tr>
<td>SVM</td>
<td>24</td>
<td>228</td>
<td>258</td>
<td>0</td>
<td>1</td>
<td>0.469</td>
</tr>
</tbody>
</table>

using the same division of sequences into training and testing as in Experiment I. As the distance measure, we used the same DTW measure we used for hierarchical clustering of the bootstrap patterns in our method. We varied \(k\) from 1 to 5. The results are shown in Table 2. While the false positive rates are much lower than those obtained in our method, the hit rates are unacceptable. For \(k\)-NN to be a practical anomaly detection method, we would have to adjust it. For example, we could impose a distance threshold beyond which a pattern is considered anomalous even if a majority of the nearest neighbors are normal.

4.2.3 Experiment III: PCA

In this experiment, we classified sequences as normal or anomalous using a Gaussian density estimator derived from principal components analysis (PCA). Since PCA requires a fixed-length input vector, we calculated, for each sequence in the testbed data set, a summary vector consisting of the means and standard deviations of each observation vector element over the entire sequence. With seven features in the observation vector, we obtained a 14-element vector summarizing each the sequence. After feature summarization, we normalized each component of the summary vector by z-scaling. Then, since we are performing probability density estimation for the normal patterns, we applied PCA to the 139 normal sequences in the bootstrap set. We chose the number of principal components accounting for 80% of the variance in the bootstrap data. Finally, we classified the remaining 521 test sequences using the PCA model to calculate the Mahalanobis distance of each sequence’s summary vector to the mean of the normal bootstrap patterns’ summary vectors. The green line in Figure 6 is the ROC curve obtained by varying the Mahalanobis distance threshold, and Table 2 shows detailed results for anomaly detection at a 100% hit rate. The high false positive rate at this threshold and the overall poor performance in the ROC analysis show that PCA is clearly inferior to our proposed method.
4.2.4 Experiment IV: SVMs

Here we used the same summary vector technique used in Experiment III but performed supervised classification using support vector machines. We used the radial basis function kernel implementation in LIBSVM (Chang and Lin, 2001) with grid search for the optimal hyperparameters using five-fold cross validation on the training set (150 sequences). The blue line in Figure 6 is the ROC curve obtained by varying the threshold on the signed distance to the separating hyperplane used for classification as normal or abnormal. Table 2 shows the detailed anomaly detection results for SVMs at a 100% hit rate. Although the results are clearly better than those obtained from k-NN or PCA, they are also clearly inferior to those obtained in Experiment I.

5 DISCUSSION AND CONCLUSION

In this paper, we have proposed and evaluated a new method for bootstrapping scene-specific anomalous human behavior detection systems. 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. On a testbed data set acquired in a real-world video surveillance situation, with a bootstrap set of 150 sequences, the method achieves a false positive rate of merely 7.4% at a hit rate of 100%. The experiments demonstrate that with a collection of simple HMMs, it is possible to learn a complex set of varied behaviors occurring in a specific scene. Deploying our system on a large video sensor network would potentially lead to substantial increases in the productivity of human monitors.

The main limitation of our current method is that the blob tracking process is not robust for complex events involving multiple people. The method also does not allow evolution of the learned bootstrap model over time. In future work, we plan to address these limitations.

ACKNOWLEDGEMENTS

This work was partly supported by a grant from the Royal Thai Government to MND and by graduate fellowships from the Royal Thai Government to KO. We thank the AIT Computer Vision Group for valuable discussions and comments on this work.

REFERENCES


