Automated Intrusion Detection for Video Surveillance Using Conditional Random Fields

Dierck Matern, Alexandru Condurache and Alfred Mertins
Institute for Signal Processing
University of Luebeck
{matern, condurache, mertins}@isip.uni-luebeck.de

Abstract

In this paper, we propose a method for intrusion detection in a video surveillance scenario. For this purpose, we train a conditional random field (CRF) on features extracted from a video stream. CRFs estimate a state sequence, given a feature sequence. To detect intrusions, we analyze this state sequence. CRFs are usually trained in a supervised manner. Here, we especially propose a new training algorithm for CRFs based on expectation maximization, which can be used with unlabeled data. We apply the resulting trained CRF to separate normal activities from suspicious behavior. We have successfully tested our algorithm on 169 sequences.

1 Introduction

Automated video surveillance is a fundamental task for security applications. Assume following scenario. A camera monitors an enclosed areal. A perpetrator attempts to conduct an unallowed activity. We are supposed to design a system that detects such unallowed activities. The perpetrator attempts to deceive the system by conducting this activity in such a way that it appears similar to an allowed one. The only difference is that the concealed unallowed activity takes longer (or shorter) than the allowed activity which it imitates. Therefore the system would detect the allowed activity for a longer (or shorter) than usual period of time. This unallowed activity is embedded in a sequence of allowed activities, hence it is not trivial to detect the unallowed one.

With motion detectors, we cannot separate an allowed activity from an unallowed one. Hence, we need to use better automated methods to separate the activities.

In the automated surveillance, we have the problem that the activities we want to detect, most importantly intrusions, are not well defined. Also a potential intruder will act "normal". However, he will usually do so for a protracted time. For example, if he uses a special tool to open the door, it will take considerably longer time to do so than using a normal key. In this paper, we therefore concentrate on the description of the normal behavior rather than the direct modeling of the suspicious behavior, and detect differences from the normal behavior. We call this task "event detection" and the situations when we want to react are named "events". Everything else is called the "normal case".

An example algorithm can be found in [4]. However, that method is based on linear predictors. Using linear predictors can be inappropriate for our problem. In principle, an event that we can detect can either be a different measurement (that is, we can detect the event instantaneously using a single frame by some arbitrary measure) or a context based difference where an event is visible by considering sequences of frames, or a combination of both. The first case is often called salience detection [1, 9]. In this paper, we concentrate on the context based differences. The context based differences are more useful if we assume that the observed person tries to obfuscate his actions and therefore imitates a person who acts normal. The algorithm in [4] and also the one in [5] concentrate on the differences in the measurements, hence they are no alternatives to our algorithm.

We train a conditional random field (CRF) [3] on features we extract from videos which include normal case activities only. We use CRFs because they are data-driven models which are based on the principal of maximum entropy [3, 6] and therefore apply minimal assumptions on the features. Further, they have proven to be superior in a great variation of tasks [3, 2], so they are an interesting algorithm for our problem.

A trained CRF generates a state sequence that is conditioned on the given data. This is a difference to many other models based on the Markov theory like hidden Markov models [8], which assume the measurements to be conditionally dependent on the state sequence. This difference allows us to avoid modeling the distribution of the features in a direct manner.

A problem with the usual training of CRFs is that it needs a labeled training dataset. That is, for each activity in the training data, we should have a corresponding state [3, 2]. During training, the parameters of the CRF can then be adapted such that the state sequence the CRF generates is equal to the given training state sequence. Because prelabeling of the training data can be expensive, we have developed a new unsupervised training algorithm based on Expectation Maximization (EM) [7]. Events are detected as a result of a statistical analysis of the state sequence generated by the CRF.

The rest of the paper is structured as follows. In Section 2, we first discuss the features we extract from the video streams. We then discuss the EM CRF training algorithm and the statistical analysis of the state sequences for the purpose of event detection. In Section 3, we demonstrate our algorithm on an exemplary video surveillance scenario. In Section 4, we give a summary and outlook of the proposed method.

2 Conditional Random Fields for Intrusion Detection

CRFs are feature dependent Markov models, linear chain CRFs are therefore Markov chains [3]. Given a feature sequence, linear chain CRFs give the probabilities of a corresponding state sequence. It is also possible to determine the probability of each state at each time step.

In the following, we discuss the features we extract from the video stream. We train a linear chain CRF on these features in an unsupervised manner, as discussed afterwards. We further apply a statistical analysis of the states sequence. This analysis we discuss in Section 2.3.

2.1 Feature Extraction

To extract features from the video streams, we use an adaptive yet simple foreground extraction algorithm [4]. Assume the frames $\mathbf{F}^n$, $n = 0, 1, 2, \ldots$ are images with $\mathbf{F}^n : \{(i,j)|1 \leq i \leq M_1, 1 \leq j \leq M_2\} \to [0,1]^3$, where $M_1 \times M_2$ is the resolution, and F_{ij}^n is a pixel of $\mathbf{F}^n$ at the coordinates (i,j). From the first n_0 frames $\mathbf{F}^0, \mathbf{F}^1, \mathbf{F}^2, \ldots, \mathbf{F}^{n_0}$ where no person is located in the visible area, we initialize a background model $\mathbf{B}^{n_0}$ as the mean of the initial frames. Further, we initialize a threshold with $T_{ij}^{n_0} = 1$ for each pixel with coordinates (i,j). We then compute the *foreground* $\mathbf{I}^n$ for each $n > n_0$ by

$$I_{ij}^n = [\![\|F_{ij}^n - B_{ij}^{n-1}\|_2 > T_{ij}^{n-1}]\!],$$

where $[\![P]\!]$ is 1 if the predicate P is true and 0 else. The background image and the thresholds are then adapted by

$$B_{ij}^n = \begin{cases} B_{ij}^{n-1} & \text{if } I_{ij}^n = 1 \\ \alpha F_{ij}^n + (1-\alpha) B_{ij}^n & \text{else,} \end{cases}$$

$$T_{ij}^n = \begin{cases} T_{ij}^{n-1} & \text{if } I_{ij}^n = 1 \\ \alpha(\|F_{ij}^n - B_{ij}^{n-1}\|_2 + T_{off}) & \\ \quad +(1-\alpha) T_{ij}^{n-1} & \text{else,} \end{cases}$$

where $T_{off} > 0$ is a parameter to suppress noise (in our experiments, $T_{off} = 0.25$). The features we use are

$$\mathbf{x}(n) = \left[\left[I_{ij}^n \right]_{i=1}^{M_1} \right]_{j=1}^{M_2},$$

the vector representation of the foreground images. We use this algorithm rather than a more sophisticated tracking algorithm due to its simplicity and few assumptions about the form of the observed object. Further, in our experiments in Section 3, this algorithm has shown to be sufficient. An example of the video stream and the extracted foreground can be found in Figure 1.

2.2 Unsupervised Training of CRFs

CRFs are usually trained in an supervised manner [3, 2]. However, we do not want to pre-label the feature vectors which is necessary if we apply an observed training algorithm, because we believe that a blind and unsupervised training algorithm is more practical for our surveillance scenario.

A linear chain CRF computes the probability of a state sequence $\mathbf{S} = s(1), s(2), \ldots, s(N)$, given a sequence of features vectors $\mathbf{X} = \mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(N)$. We have discussed the feature vectors in Section 2.1. We assume K different states, that is, for each n with $1 \leq n \leq N$, $s(n) \in \{\zeta_1, \zeta_2, \ldots, \zeta_K\}$ where ζ_i is a possible state. Our training algorithm determines the differences of the possible states, their properties and the transition probabilities.

The definition of a linear chain CRF is as follows. Given the feature sequence $\mathbf{X}$, the probability of a state sequence $\mathbf{S}$ (see [3]) is given by

$$p(\mathbf{S}|\mathbf{X};\lambda) = \frac{1}{Z_\lambda(\mathbf{X})} \exp\left(\sum_{n=1}^{N} \sum_{k=1}^{K} \lambda_k^\top \varphi_k\left(s(n), \mathbf{x}(n)\right) \right) \times \exp\left(\sum_{n=1}^{N} \sum_{j,k=1}^{K} \lambda_{jk} \varphi_{jk}(s(n-1), s(n)) \right) \quad (1)$$

where $\lambda = \{\lambda_i, \lambda_{jk}\}$ with $i, j, k = 1, 2, \ldots, K$ is the set of the parameters for the CRF, $Z_\lambda(\mathbf{X})$ is a normalization constant such that $p(\mathbf{S}|\mathbf{X};\lambda)$ is a probability, and

$$\varphi_k(s(n), \mathbf{x}(n)) = [\![s(n)]\!] \cdot \mathbf{x}(n),$$
$$\varphi_{jk}(s(n-1), s(n)) = [\![s(n-1) = \zeta_j]\!] \cdot [\![s(n) = \zeta_k]\!].$$

Note that λ_k is actually a vector of the length of the feature vectors, and λ_{jk} is a scalar value. λ is calculated in the training phase. We can train the CRF using gradient based optimization applied to the log-likelihood of the CRF.

We train the CRF in an EM-like algorithm. The EM algorithm we use here consists of two steps, the E step where we estimate a state sequence, and the M step where we optimize the parameters to the estimated sequence. Doing this several times iteratively leads to a locally optimal solution [7]. We set up the EM algorithm such that the CRF does not degenerate, that is, all possible states can be generated by the CRF if we provide the respective feature sequences, and the training data provides data for all possible states. We therefore control the state entropy, which is the entropy over the occurrence frequency of the possible states in a state sequence.

In the initialization of the EM algorithm, we set λ at random. In the E step, a state sequence is estimated by

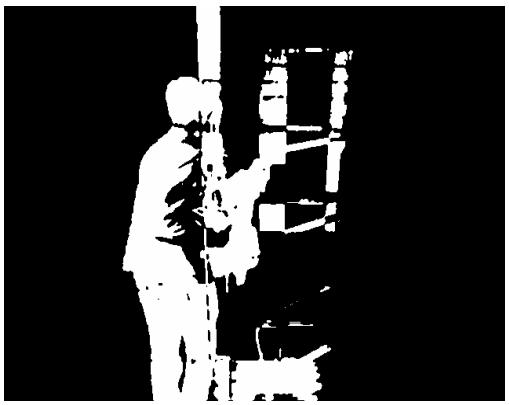
$$\tilde{s}(n) = \arg \max_{s(n)} \tilde{p}(s(n)|\mathbf{X}, s(n-1), s(n+1); \lambda) \quad (2)$$

with $\tilde{p}$ as the probabilities of one state in the state sequence divided by the sum of the probability of this very state in the whole sequence,

$$\tilde{p}(s(n) = \zeta_k|\mathbf{X}, s(n-1), s(n+1); \lambda) = \frac{p(s(n) = \zeta_k|\mathbf{X}, s(n-1), s(n+1); \lambda)}{\sum_{n=1}^{N} p(s(n) = \zeta_k|\mathbf{X}, s(n-1), s(n+1); \lambda)}.$$

After the E step, we perform the M step. In the M step, we maximize the log-likelihood of the state sequence $\tilde{\mathbf{S}} = \tilde{s}(1), \tilde{s}(2), \ldots$ using gradient ascend. We do not use optimization algorithms with higher order because due to the new state sequence in each iteration, the location of the unique optimum for the CRF [6, 3, 2] changes, and we do not want to over-adapt

(a) Example Image



(b) Foreground Features

Figure 1. An example image from the experiments (a), and the extracted features (b).

to a temporary state sequence. The gradient of the log-likelihood of a state sequence, given the feature sequence, is

$$\nabla_{\lambda_k} \log(p(\tilde{\mathbf{S}}|\mathbf{X};\lambda)) = -\nabla_{\lambda_k} \log(Z_\lambda(\mathbf{X})) + \sum_{n=1}^{N} \varphi_k(\tilde{s}(n), \mathbf{x}(n)) \tag{3}$$

$$\nabla_{\lambda_{jk}} \log(p(\tilde{\mathbf{S}}|\mathbf{X};\lambda)) = -\nabla_{\lambda_{jk}} \log(Z_\lambda(\mathbf{X})) + \sum_{n=1}^{N} \varphi_{jk}(\tilde{s}(n-1), \tilde{s}(n)), \tag{4}$$

where $\nabla_{\lambda_k} \log(Z_\lambda(\mathbf{X})) = \sum_{n=1}^{N} E[\varphi_k(s, \mathbf{x}(n)); \lambda]$ and $\nabla_{\lambda_{jk}} \log(Z_\lambda(\mathbf{X})) = \sum_{n=1}^{N} E[\varphi_{jk}(s_1, s_2); \lambda]$. Here E is the expected value, given the actual parameters λ. s, s_1, s_2 are the random variables whose distribution is given by the CRF. For details and derivation, see [2].

We use constrained optimization to avoid overfitting and to increase the state entropy. The constraints are

$$\sum_{i=1}^{M} \lambda_j(i) = 0, \qquad j = 1, 2, \ldots, K, \tag{5}$$

$$\sum_{i=1}^{M} (\lambda_j(i))^2 = 1, \qquad j = 1, 2, \ldots, K, \tag{6}$$

$$\sum_{k=1}^{K} \lambda_{jk} = 0, \qquad j = 1, 2, \ldots, K, \tag{7}$$

$$\sum_{k=1}^{K} (\lambda_{jk})^2 = 1, \qquad j = 1, 2, \ldots, K., \tag{8}$$

where M is the number of features. The constraints (5) and (6) control the weighting vectors λ_k which are related to the feature vectors, see (1). Constraint (5) decreases the iterations needed in the EM algorithm. It can be interpreted as a specialization of each state to a single action. The constraint (6) is set to avoid overfitting. The constraints (7) and (8) have similar effects on the transition probabilities. This is a different regularization than in most CRF algorithms and keeps the state entropy as high as possible while adapting the CRF to a temporary state sequence.

With these constraints, we perform one step of the gradient ascend optimization. After this, we start again with the E step, using the new λ to generate a new state sequence. After several iterations, we get a locally optimal solution $\hat{\lambda}$ which is sufficient for our surveillance problem.

Using the trained model, we calculate a state sequence for the feature sequence $\mathbf{X}$ with $\hat{\mathbf{S}} = \arg\max_{\mathbf{S}} p(\mathbf{S}|\mathbf{X};\hat{\lambda})$. This is a sequence of automatically learned states and describes the activities in the visible area.

2.3 Statistical Analysis of the State Sequence

In the last section, we discussed an EM algorithm to train a CRF on features extracted from video streams. However, because we discuss video streams, actions are observable for several consecutive frames. Hence, in a state sequence, the same state appears repeatedly. To detect the events, we therefore measure the length of each time window defined over these repetitions (segments). If the probability of the actual length is low, assuming a normal action, we have detected an event. Thus, we are able to detect events, even though the observed person tries to obfuscate his unallowed activity.

To calculate the likelihood of the length of the actual segment, we train a Gaussian Mixture Model (GMM) on the lengths of the segments in the training data. In the inference, if the likelihood of the length of a segment is low, we say we have detected an event.

3 Experiments and Discussion

We have set up a camera in a floor, monitoring a door to an adjacent room, and we measure 30 frames per second. For speedup, we reduced the native resolution of 640×480 to $M_1 \times M_2 = 32 \times 32$.

The scenario we use is as follows. Normally, a person is passing the door, locking and unlocking the door, entering and leaving the room or looking into the room through a window, see Figure 1. This scenario we have set up to simulate usual activities of a watchman. For the event, we assume an intruder who breaks into the

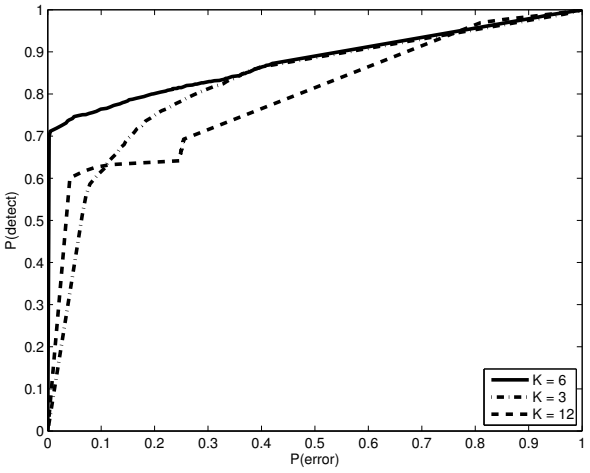


Figure 2. Receiver Operation Characteristic (ROC) of the proposed method. The ROC is the plot of the probability to correctly detect an event against the probability of a false alert, using a varying threshold.

observed room. The CRF was trained on the normal-case data only. The activities of the intruder we assume in our scenario are performed by the same person. Hence, we are sure that detected suspicious behavior is based on his activities only, not on personal differences to the watchman.

We have 153 normal case video sequences where the observed person behaves like a watchman. Further, we have 16 event-case video sequences. We have chosen this difference between the number of normal-case and event videos because of two reasons. First, the events are by default rare actions but have, in real scenarios, many variations. More event videos would increase the redundancy of the events only, which is not a realistic scenario. Second, the more difficult part is to avoid false alerts, so we need a higher number of the normal case videos for our evaluation of the proposed method. The original data is available for download.[1]

We have trained on 60% of the normal case videos. We have tested the number of states K between 3 and 20. A high value of K results in very short segments, a low value decreases the possible complexity of the normal case. $K = 6$ is a trade-off between the lengths of the usual actions and their complexity. For the lengths of the segments, we train GMMs with ten modes. The exemplary Receiver Operating Characteristics (ROCs) can be seen in Figure 2. Due to the high detection rate of the events, considering a low false alert rate, we rate our experiments as a success. Note that even in the event sequences, several segments can be, by themselves, considered be be normal (for example, if a person enters the visible area, we cannot decide if he will act normal or abnormal in the beginning). Hence, a part of the event videos consists of normal activities, and higher detection rates cannot be expected.

4 Summary and Outlook

We have proposed an automated system for video surveillance, based on CRFs. The method has shown to be capable of detection of unprecedented activities, like intrusions.

Our system learns the actions without prelabeling, which is an advantage in this setting, because the application is simpler. Further, we do not make assumptions on the activities performed in the visible area. Therefore, we can apply this system for many other observation problems.

The test for the lengths of the segments is a simple yet effective method for the detection of the events in this scenario. Differences in measurements of single frames are less likely to be detected, because a intruder would try to avoid unlikely behavior.

In the future, we plan to adapt the algorithm to online training. That is, we even do not record a training sequence, but the model is trained while recording the features.

Further, we can include the capability to detect strange measurements that are related to unprecedented actions. An example of this event detection can be found in [5]. That method can be incorporated in our method using more sophisticated features. However, a more complex scenario has to be set up for the necessity of this adaption. For the surveillance scenario, our method has shown to be sufficient.

[1] http://www.isip.uni-luebeck.de/downloads

References

[1] O. Boiman and M. Irani: "Detecting Irregularities in Images and in Video", *Proceedings of the International Conference on Computer Vision*, pp. 462-469, 2005.

[2] R. Gupta: "Conditional Random Fields," technical report, IIT Bombay, 2006.

[3] J. Lafferty, A. McCallum and F. Pereira: "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data," *Proceedings of the Eighteenth International Conference on Machine Learning*, pp. 282-289, 2001.

[4] D. Matern, A. Condurache and A. Mertins: "Linear Prediction based Mixture Models for Event Detection in Video Sequences", *Proceedings of the Iberian Conference on Pattern Recognition and Image Analysis*, pp. 25-32, 2011.

[5] D. Matern, A. Condurache and A. Mertins: "Event Detection using Log-Linear Models for Coronary Contrast Agent Injections" *Proceedings of the International Conference on Pattern Recognition Applications and Methods*, pp. 172-179, 2012.

[6] A. McCallum, D. Freitag and F. Pereira: "Maximum Entropy Markov Models for Information Extraction and Segmentation," *Proceedings of the Seventeenth International Conference on Machine Learning*, pp. 591-598, 2000.

[7] G. McLachlan and T. Krishnan: " The EM Algorithm and Extensions", Second Edition, Wiley Series in Probability and Statistics, 2008.

[8] L. Rabiner: "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*, pp. 257-286, 1989.

[9] A. Sodemann, M. Ross and B. Borghetti: "A Review of Anomaly Detection in Automated Surveillance", *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 42, vo. 6, pp. 1267-1272, 2012.