Topology Free Hidden Markov Models: Application to Background Modeling

B. Stenger
Department of Engineering
University of Cambridge
Cambridge CB21PZ
email: bdrs2@cam.ac.uk

V. Ramesh, N. Paragios and F.Coetzee*
Imaging & Visualization Department
Siemens Corporate Research
Princeton, NJ
email: nikos,rameshv@scr.siemens.com

J.M. Buhmann
Institute for Computer Science III
University of Bonn
Germany
email: jb@cs.bonn.edu

Abstract

Hidden Markov Models (HMMs) are increasingly being used in computer vision for applications such as: gesture analysis, action recognition from video, and illumination modeling. Their use involves an off-line learning step that is used as a basis for on-line decision making (i.e. a stationarity assumption on the model parameters). But, real-world applications are often non-stationary in nature. This leads to the need for a dynamic mechanism to learn and update the model topology as well as its parameters. This paper presents a new framework for HMM topology and parameter estimation in an online, dynamic fashion. The topology and parameter estimation is posed as a model selection problem with an MDL prior. Online modifications to the topology are made possible by incorporating a state splitting criterion. To demonstrate the potential of the algorithm, the background modeling problem is considered. Theoretical validation and real experiments are presented.

1 Introduction

Hidden Markov Models (HMM) [R89] are a powerful tool that has been shown to be of great use in speech and signal processing. These models (along with variants) are increasingly being applied by the vision community to problems such as image segmentation, face recognition, gesture interpretation, event understanding, and background modeling (e.g. [BK00], [RKJB00]). HMMs provide the framework for modeling dynamical or spatial dependencies and correlations between measurements. The dynamical dependencies are modelled implicitly by a Markov chain with a specified number of hidden states and a transition matrix, with observations that are conditionally independent given a state.

Numerous methods for estimation of the HMM model parameters exist in the literature. These methods can be classified into two major categories: batch [R89] and incremental ([DN96] [HL98]). Some examples of the batch methods include: the EM algorithm (i.e. Baum-Welch) and segmental K-means algorithm. Offline methods guarantee that the parameter estimates correspond to local maxima, but are computationally expensive. The online methods cited attempt to incrementally update HMM parameters (without topology modifications) with initial starting points determined by batch estimation. These methods can deal with slowly varying drifts in model parameters with the risk of not being able to track the real parameters under sudden changes. Once the HMM parameters are estimated, the well-known Viterbi algorithm [R89] is used to compute the likelihood of a given observation for a HMM model.

Major issues in the use of HMMs in real world applications involve two points: real-time computation, and topology modification to address non-stationarities due to dynamically varying conditions. Automatic selection of HMM topology during batch-learning phase has been addressed in the literature. These methods use state-splitting [MCB96],[OS97], or merging, [BK00], criteria to iteratively estimate the optimal number of states. More recently, a maximum aposteriori estimation scheme utilizing an entropic prior was presented wherein a redundant number of states is initially assumed and weak states satisfying a given probability threshold are eliminated iteratively [MB98]. However, dynamic update of the model topology was not addressed in the work. Besides, state-merging schemes are computationally intensive and online updates are not practical.

In video analysis applications, a topology selection scheme based on state-splitting is more applicable since the computational complexity is reasonable and online adaptation can be done. Towards this end, we explore, compare and validate a class of state-splitting algorithms with various well-known criteria including: a chi-squared goodness-of-fit test, a cross-validation criterion and, an MDL stopping criterion. The simulations point out that state-splitting with the MDL stopping criterion provides compact topologies with fair accuracy. A specific application in video surveillance is considered to illustrate the efficacy of the selected state-splitting algorithm.

Background modeling is an important task in video surveillance applications [TKBM99]. This module com-

*Frans Coetzee is now with Certus-Services Inc. Princeton, NJ

putes the "distance" between the observed image and the current background statistical model which is used to determine all non-stationary objects in the scene. Several methods for background modeling can be found in the literature. A widely used linear model for background modeling is the Kalman filter [K60],[KB90]. A significant work on background modeling and adaptation can be found in [SG99] where the distribution of each sample is approximated as a mixture of Gaussians components. Another interesting piece of work for real time background modeling based on Hidden Markov Models with fixed topology was recently introduced in [RKJB00]. In this paper, we illustrate the use of the state-splitting algorithm for background modeling using topology free HMMs. Unlike the other methods where the model is fixed and its parameters are updated, the new framework is dynamic, (*e.g.* it can change naturally the topology over time) and can deal with sudden as well as gradual illumination changes.

This paper is organized with the following fashion. In Section 2, we briefly introduce the Hidden Markov Model concept, while in section 3 the application context is descibed.

2 Hidden Markov Models

A hidden Markov model is a stochastic finite state machine, specified by a tuple $(\mathcal{S}, A, \pi)$ where

- $\mathcal{S}$ is a discrete set of hidden states with cardinality N,
- π is the probability distribution for the initial state

$$\pi(i) \;=\; P(\,s_i\,) \qquad s_i \in \mathcal{S}\ .$$

- A is the state transition matrix with probabilities:

$$a_{ij} \;=\; P(\,s_j \,|\, s_i\,) \qquad s_i, s_j \in \mathcal{S}\ ,$$

where the state transition coefficients satisfy $\sum_{s_j \in \mathcal{S}} a_{ij} \;=\; 1\,,\; s_i \in \mathcal{S}$.

The states themselves are not observable. The information accessible consists of symbols from the alphabet of observations $\mathcal{O} = (o_1, \ldots, o_T)$ where T is the number of samples in the observed sequence. For every state an output distribution is given as

$$b_i(k) \;=\; P(\,o_t = k \,|\, s_i\,) \qquad k \in \mathcal{O},\; s_i \in \mathcal{S}\ .$$

Thus, the set of HMM parameters θ consists of the initial state distribution, the state transition probabilities and the output probabilities. HMMs can be used for classification and pattern recognition by solving the following problems:

- **The Evaluation Problem:** Given the model with parameters θ, calculate the probability for an observation sequence O. Let $O \;=\; (o_1, ..., o_T)$ denote the observation sequence and $S \;=\; (s_1, ..., s_T)$ a state sequence. The probability $P(O \,|\, \theta)$ can be obtained by *Forward Algorithm*.

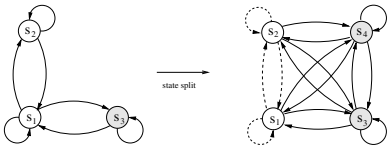


Figure 1. *Example of state-splitting.*

- **The Decoding Problem:** Find the optimal state sequence for an observation sequence

  $\mathrm{argmax}_{S \in \mathcal{S}^T}\, P(S \,|\, O, \theta)$. This can be done by the *Viterbi* algorithm.

- **The Learning Problem:** Given an observation sequence O and the HMM parameters, find the parameters $\widehat{\theta}$ which maximize $P(O \,|\, \theta)$, i.e. $\widehat{\theta} \;=\; \mathrm{argmax}_\theta\, P(O \,|\, \theta)$.

  This question corresponds to training an HMM. The state sequence is not observable. Therefore, the problem can be viewed as a missing-data problem, which can be solved via an EM-type algorithm. In the case of HMM training, this is the *Baum-Welch* algorithm.

A tutorial on HMM models, the estimation problems mentioned above, and their applications to speech modeling and recognition can be found in [R89].

3 Batch Algorithms for State Splitting

We have seen that in the applications of HMM models to vision problems we need a way to automatically select and update the topology as well as the parameters online. Various batch versions of algorithms for state splitting can be found in the literature. These algorithms are considered and presented in the next section.

The basic idea behind state splitting algorithms for HMM topology estimation is to perform a search over all hidden Markov models up to a certain maximum number of states. An appropriate model selection criterion can be used to choose the best model. The major open issues here include:

- The selection of the criterion function that is used during for model comparison,
- The initialization of the starting point for the iterative algorithm that estimates model parameters.
- The computational complexity of the model search.

3.1 State Splitting Using A Goodness-Of-Fit Test

The main idea is to perform the HMM model parameter estimation for a given number of hidden states. This is done by the Baum-Welch algorithm. Thus, given the model parameters, an optimal state labeling of the observations is obtained using the Viterbi algorithm. This in turn provides a mechanism for estimating the histograms corresponding to the conditional likelihood of the observations given state

values. A goodness of the fit test is then used to compare the histograms with their continuous counterparts provided by the HMM model parameters estimated. Discrepancies in the goodness of the fit measure correspond to bad approximations. There is a number of similarity test measures like the the chi-squared test and the Kolmogorov-Smirnov test. In this paper the chi-squared test is considered.

In our algorithm, the chi-squared test is used to test whether the data is Gaussian. If this hypothesis is rejected with high significance (95%), then the state is split. There has been prior work on state-splitting using the goodness of fit tests. Montacié et al. selects threshold values heuristically [MCB96].

3.2 State Splitting Using a Cross Validation Criterion

The idea behind cross-validation techniques is to test models for generalization performance [Z93]. The available data is split into a training set, which is used to adapt the model parameters, and a test set to assess the model performance on new data. With increasing model complexity, the error on the training set is expected to decrease. However, on the validation set the error will begin to increase beyond a certain model complexity because of overfitting to a given training set. The model complexity is measured in terms of the number of states while the model performance is evaluated using the log-likelihood.

Testing for generalization performance is the idea of the following algorithm:

1. Start with an HMM with $N = 1$ state.
2. Train the model on the training set using the Baum-Welch algorithm.
3. Compute the data likelihood of the test set, given this model.
4. If the likelihood on the test set decreases with the split, stop.
5. Select a split candidate with a goodness-of-fit test and split this state.
6. Goto step 2.

There is, however, the issue of how one should select the splitting candidate in step 5. One option is a goodness-of-fit test. This test is not efficient when higher than one-dimensional output distributions are considered. Another possibility is to use a heuristic 'variability' criterion computed for every state [TS92]. Finally, the method that is used in our experiments is to test for a likelihood increase on a constrained parameter subset [OS97].

The term for the expected log-likelihood of the observed data O and the state sequence S, given the parameters θ can be written as, (see [OS97]):

$$E\,[\log p(O,S|\theta)|O,\theta] = \sum_{i,j\in\mathcal{S}}\sum_{t=1}^{T}[p(s_t=j,s_{t-1}=i|O,\theta) \log p(s_t=j\,|\,s_{t-1}=i,\theta)] + \sum_{j\in\mathcal{S}}\sum_{t=1}^{T} p(s_t=j\,|\,O,\theta)\ \log p(s_t=j\,|\,O,\theta)\ .$$

This is basically the term which is maximized in the M step of the Baum-Welch algorithm. A computationally expensive idea is to split each state $j \in \mathcal{S}$, calculate the likelihood and determine the state that provides the largest increase in likelihood with respect to the likelihood of the current model.

In order to reduce the complexity, one can constrain the unknown parameters: all transition probabilities are retained except for the transitions from and to the split candidate [**fig.** (1)]. This method reduces the state probability parameters to estimate from $(N-1)^2$ to $3N-1$ and the number of output parameters from $N\eta(1)$ to $2\eta(1)$, where N is the number of states and $\eta(1)$ is the number of parameters of the output density.

The increase in likelihood $G(i),\ i \in \mathcal{S}$ for a split is then just the difference in the terms of the sum containing the split i candidate and the two new states $\tilde{i}_1, \tilde{i}_2$, is given by

$$G(i) = -\sum_{t=1}^{T}[p(s_t=i,s_{t-1}=i\,|\,O,\theta)\log p(s_t=i\,|\,O,\theta) \log p(s_t=i\,|\,s_{t-1}=i,\theta) + p(s_t=i\,|\,O,\theta)\,] + \sum_{t=1}^{T}\Bigg[\sum_{k,j\in\tilde{\mathcal{S}}} p(s_t=j,s_{t-1}=k\,|\,O,\theta)\ \log p(s_t=j\,|\,s_{t-1}=k,\theta) + \sum_{j\in\tilde{\mathcal{S}}} p(s_t=j|O,\theta)\ \log p(s_t=j|O,\theta)\Bigg]\ ,$$

where $\tilde{\mathcal{S}}$ denotes the set of the two states $(\tilde{i}_1, \tilde{i}_2)$ resulting from splitting state i.

However, the search space is still rather large, because we are looking for the optimal transition probabilities and output density parameters at the same time. In the speech recognition community this problem has been approached by dividing splits into two categories (see [J97]):

- In *contextual splits*, the transition probabilities are fixed and only the likelihood increase by adapting the output densities is estimated.
- In *temporal splits* also the transition probabilities are adapted.

The likelihood increase for both of these domains are computed and the better one is chosen. It is clear that the considered goodness-of-fit test corresponds to a contextual split.

3.3 State Splitting Using an MDL Criterion

In this section an algorithm is presented, which uses the minimum description length costs as a stopping criterion. The splitting procedure is performed as described in the previous section. In this version of the algorithm, the idea is to keep a copy of the current HMM with N states (model M_N) and test the likelihood increase when splitting a state (model M_{N+1}). Thus, for an HMM with data vectors of dimensionality d, Gaussian densities for the observations in each state, the state splitting criterion according to the MDL principle, [R86], is given by (selection of the M_{N+1} model),

$$\log L(\widehat{\theta}_{M_{N+1}}|x) - \log L(\widehat{\theta}_{M_N}|x) > \left(N + \frac{d^2+3d}{4}\right) \log n \ .$$

We can do the same analysis for other model selection criteria such as the AIC criterion [A74].

The state splitting scheme using the MDL-criterion can be written therefore as follows:

1. Initialize a model with N=1 states and train the model with the Baum-Welch algorithm.
2. Select the split which maximally increases the likelihood on a constrained subset of parameters.
3. Determine the likelihood increase for the complete model by training a model after state-splitting with Baum-Welch.
4. If the increased likelihood is larger than the MDL penalty difference, split and goto step 2, else stop.

4 An Online Training Algorithm

For some applications, for example in the case of a continuous data stream, the batch learning algorithm alone is not very suitable. If we would like to apply batch learning in these cases, we would have to use sections of the input stream of a certain length and train the model on this data in intervals. But several open issues have to be dealt with in this method: the selection of the length of the time intervals, the integration of the existing knowledge as well as the cases of non-stationarity.

In this section an online learning algorithm is presented, which can be used to train an HMM. Theoretical justification can be found in incremental versions of the EM algorithm [N91],[NH98],[FR97].

In the online version of the algorithm, the model is updated after each sample value. The idea is that the *forward variables* α in the forward algorithm, [R89], are updated in every step. These variables give the probability of observing $o_1, ..., o_t$ and being in state $i \in \mathcal{S}$ at time t:

$$\alpha_t(i) \ = P(s_t = i,\, o_1, ..., o_t) \qquad i \in \mathcal{S} \ .$$

They can be updated as in the batch learning version, so in the first step the forward variables are set to

$$\alpha_1(i) \ = \ \pi(i)\, p(o_1 \,|\, s_1 = i) \qquad i \in \mathcal{S} \ ,$$

where π is the initial state distribution, which may include some prior knowledge. In the experiments, a uniform distribution on the states was used. With every observation, the α values are updated by summing the probabilities over all possible paths, which end in the new state $j \in \mathcal{S}$:

$$\alpha_t(j) \ = \ \left[\sum_{i=1}^{N} \alpha_{t-1}(i)\, a_{ij}\right] p(o_t \,|\, s_t = j) \qquad j \in \mathcal{S} \ .$$

The online learning problem is addressed by setting the *backward variables* to 1, since for $t' > t$ no observations are known. Therefore, the current time instant t is assumed to be T:

$$\beta_{t+1}(j) \ = \ P(o_{t+2}...o_T \,|\, s_{t+1} = j) \ = \ 1 \qquad j \in \mathcal{S} \ .$$

Thus, the probability of being in a particular state $i \in \mathcal{S}$ at time instant t, given the observation, can be written as:

$$\gamma_t(i) \ := \ P(s_t = i \,|\, o_1, ..., o_t) = \frac{P(s_t = i,\, o_1, ..., o_t)}{P(o_1, ..., o_t)} \ = \ \frac{\alpha_t(i)}{\sum_{i=1}^{N} \alpha_t(i)} \ .$$

The probability of a certain state transition $i \to j$ given the observation is:

$$P(s_{t-1} = i, s_t = j \,|\, o_1, ..., o_t) = \frac{P(s_{t-1} = i, o_1, ..., o_{t-1}) P(s_t = j | s_{t-1} = i) P(o_t | s_t = j)}{P(o_1, ..., o_t)} = \frac{\alpha_{t-1}(i) \ a_{ij} \ p(o_t \,|\, s_t = j)}{\sum_{i=1}^{N} \alpha_t(i)} \ .$$

These formulas give us the re-estimation scheme. In every step the α variables are computed and used to update the model parameters:

- Initialize the α variables and the model parameters.
- At every time step T update the state probabilities $\gamma_t(i) \ := \ P(s_t = i \,|\, o_1, ..., o_t)$ and $P(s_{t-1} = i, s_t = j \,|\, o_1, ..., o_t)$,

and use these values to update the model parameters ($\forall i, j \in \mathcal{S}$):

$$\widehat{a}_{ij}^T = \frac{\sum_{t=1}^{T} P(s_{t-1}=i, s_t=j \,|\, o_1,...,o_t)}{\sum_{t=1}^{T} \gamma_t(i)} = \frac{\sum_{t=1}^{T-1} \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)} \quad \widehat{a}_{ij}^{T-1} + \frac{P(s_{T-1}=i,\, s_T=j \,|\, o_1,...,o_T)}{\sum_{t=1}^{T} \gamma_t(i)}$$

$$\widehat{\mu}_i^T = \frac{\sum_{t=1}^{T} \gamma_t(i)\; o_t}{\sum_{t=1}^{T} \gamma_t(i)} = \frac{\sum_{t=1}^{T-1} \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)} \quad \widehat{\mu}_i^{T-1} \;+\; \frac{\gamma_T(i)\; o_T}{\sum_{t=1}^{T} \gamma_t(i)}$$

$$\widehat{\Sigma}_i^T = \frac{\sum_{t=1}^{T} \gamma_t(i)\; (o_t - \widehat{\mu}_i^T)(o_t - \widehat{\mu}_i^T)'}{\sum_{t=1}^{T} \gamma_t(i)} = \frac{\sum_{t=1}^{T-1} \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)} \quad \widehat{\Sigma}_i^{T-1} \;+\; \frac{\gamma_T(i)\,(o_T - \widehat{\mu}_i^T)(o_T - \widehat{\mu}_i^T)'}{\sum_{t=1}^{T} \gamma_t(i)}$$

The sums in these equations are computed by storing the values and adding the new terms. This can be seen as continually updating sufficient statistics, which are used to compute the new parameters. A problem with this method is, that all values from $t = 1$ to the current time instant are used to compute the sufficient statistics. If the initial parameter settings are far away from the true values, the errors made at this stage will slow down the convergence process. Also, non-stationary data sources are not well handled. Solutions to these problems are to compute the sufficient statistics only over a finite time window, or to use *exponential forgetting*. When using a time window, the past values have to be stored, which requires a lot of memory. Therefore, a version of the algorithm using exponential forgetting was implemented. In this algorithm, estimates prior in time receive less weight.

The basic idea is to replace the sums in the above re-estimation formulas by variables which are updated recursively, for example the term $\sum_{i=1}^{T} \gamma_t(i)$ is replaced by variables $R_\gamma^T(i)$ which are updated:

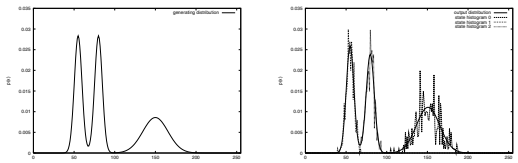
$$R_\gamma^T(i) \;=\; (1-\tau)\; R_\gamma^{T-1}(i) \;+\; \tau\; \gamma_T(i) \qquad \tau \in (0,1)\;.$$

This is done for all sum terms in the re-estimation formulas for the state probabilities and the distribution parameters.

5 Experiments

Simulations have been conducted to study the various methods described above. Within these experiments, the number of states for the generated HMMs was between three and five while training sets with 100 to 1000 samples have been considered.

The sensitivity of the schemes were studied by the mean and standard deviation of the topology (in this case the number of states of the HMM) as a function of the sample sizes. It is expected that as the sample size increases the accuracy of the estimation increases. Figure 2 illustrates the result of applying state-splitting algorithm using this test on a simulated data set with three states. In 1000 test sequences,



(a) (b)

Figure 2. *State-Splitting using the χ^2-test. A three-state HMM is generated. (a) Input distribution. (b) Output distribution using a three-states HMM model.*

Sample	1	2	3	4	5	Mean	Std Dev.
100	61	562	197	138	42	2.538	0.948
200	25	277	318	271	109	3.162	1.028
300	0	146	501	236	117	3.324	0.863
400	0	11	596	274	119	3.501	0.714
500	0	47	562	296	95	3.439	0.728
750	0	0	828	152	20	3.192	0.442
1000	0	0	842	144	14	3.172	0.413
1500	0	0	895	92	13	3.118	0.361
2000	0	0	909	81	10	3.101	0.333

Figure 3. *Cross-validation algorithm results. Column data for cols labelled 1 through 5 correspond to frequencies with which model with corresponding number of states were chosen.*

a model with 3 states was estimated in 94.1% of the cases. However, the main drawback of this algorithm is the robustness in the case of false splits. There were as many as seven states estimated for certain test sequences.

In the case of cross-validation however, we note that that topologies obtained were better. The mean number of states was 3.172 with a standard deviation of 0.413 thus showing that 100% of the time the number of states estimated was five states or less. Figure 3 shows the results obtained for samples of different sizes for the cross-validation criterion. In these tests, the first half of the sample is used as training set, the second half as test set. The table shows the result of the algorithm obtained with 1000 test sequences.

The MDL criterion had the best behavior among those tested. Figure 4 shows the behavior of the state splitting algorithm with the MDL criterion (for an HMM with four states). As one can see the standard deviation for number of states estimated is 0.045 with a sample size of 1000. This means that the state splitting method with MDL criterion produced between three to five states almost surely. To illustrate the behavior of the MDL based state-splitting algorithm and to compare the MDL and AIC criterions, we use an HMM with three states and feature vectors with dimension 2. Figure 5 shows a run of the algorithm using a sample size of 500.

The corresponding negative log-likelihood (i.e. cost with no priors), MDL costs, and AIC costs are shown in figure 6. Note how the MDL cost function has a local minimum for

Samples	1	2	3	4	5	6	Mean	Std Dev.
100	0	53	465	462	20	0	3.449	0.627
200	0	48	268	631	53	0	3.689	0.645
300	0	0	160	738	81	21	3.963	0.569
500	0	0	72	826	92	10	4.040	0.450
750	0	0	47	907	46	0	3.999	0.305
1000	0	0	0	998	2	0	4.002	0.045

Figure 4. *MDL-based state-splitting algorithm results for HMM with four states.*

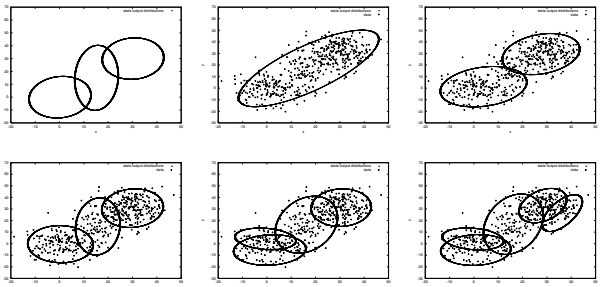


Figure 5. *Illustration of the MDL state split over time.*

the 3 state model and has higher curvature and hence has better properties than the AIC criterion.

5.1 Experiments with the Online Version

As expected, our experiments showed that the online version without exponential forgetting was sensitive to the initial values. One reason is that initial estimates may be far away from the true parameters, but are nevertheless weighted equally to all other estimates in every new estimation step. The second version of the algorithm, which uses exponential forgetting, solves the problem of adaptation to non-stationary data sources.

This version of the algorithm was tested by using a two state model. It can be seen that by using different learning parameters τ one can adapt the speed of the convergence. However, the faster this adaptation, the larger is the standard deviation of the estimates. If the learning parameter is chosen too large, the estimates of the state transitions are not very robust. It was also found that the variance estimates first increase before they decrease again to converge to the true values. This is due to the estimation errors in the means due to small sample size in the beginning.

In a second experiment, the ability to adapt to non-stationary data sources was tested. This was done by changing the parameters of the generating HMM. The transition probability matrix is switched from A_0 to A_1 and the output distribution parameters of state s_0 is changed from (μ_0, σ_0^2) to (μ_1, σ_1^2), where

$$A_0 = \begin{pmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{pmatrix}, \qquad \mu_0 = 100.0, \qquad \sigma_0^2 = 100.0,$$

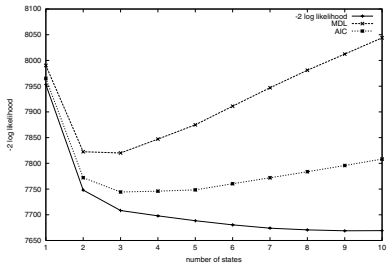


Figure 6. *Negative log-likelihood, MDL and AIC costs for HMMs with different numbers of states (corresponding to figure 5).*

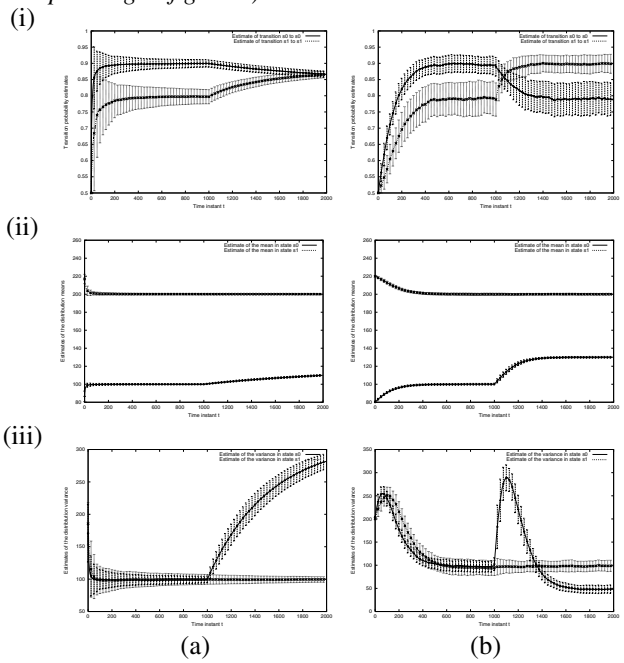


Figure 7. *Adaptation to a non-stationary data source: (a) Online Update, (b) Online Update with exponential forgeting. (i) Transition Probabilities, (ii) Means, (iii) Variances.*

$$A_1 = \begin{pmatrix} 0.2 & 0.8 \\ 0.9 & 0.1 \end{pmatrix}, \qquad \mu_1 = 130.0, \qquad \sigma_1^2 = 50.0 \ .$$

In figure 7 samples of length 2000 are generated with a two-state HMM. At data point 1000, the model parameters were changed. The plots show the means and standard deviations of parameter estimates, determined by running the algorithm 1000 times using the same initial values. For the original online update algorithm the model converges with a very slow rate.

6 Application to Background Modeling

Real-time video processing is a key feature in a number of surveillance applications like traffic surveillance, subway monitoring or building surveillance. In most of the cases

the basic module of such applications refers to *background subtraction*. This module computes the "distance" between the observed image and the current background model. This distance is used to determine all non-background objects in the scene. To perform reliable object detection a robust estimate of the background model is needed.

However, in many applications the background is non-stationary (e.g. outdoors, indoors with several configurations of the illumination sources). This task is considered to demonstrate the topology free MDL-splitting Hidden Markov Models concept. The considered application refers to two scenarios: a subway environment with typically two global states, train arrival and departure, and an indoor environment with two possible configurations of the light source. The HMM state split algorithm is used over a training sequence, and the algorithm generates the background model that consists of two states.

Figure 8 illustrates the use of global intensity means in track area to analyse train arrivals and departures in a subway setting.

The online algorithm was tested on the data. The plot of the intensity means in the track area over time is shown in [**fig.** (9)]. The peaks correspond to a train which is stopping at the station. It can be observed that the illumination intensity shifts over time and the online algorithm tracks these shifts. Initialization is done using the state-splitting algorithm (it produces 2 states) and the means are then adapted with the online algorithm.

A demonstration of this model is shown in [**fig.** (10)]. A person enters and walks in the room while a change on the light source configuration occurs. The HMM background model instantly switches to the appropriate state and provides the expected classification result. As it is clearly shown in both cases the person is detected as foreground although we have to deal with extreme light conditions. However, some errors occur due to shadows which can be dealt with by the use of illumination invariants. Inter-pixel constraints can be also used to deal with this problem.

In order to validate state splitting HMM-based background modeling algorithm, a comparison with a well known and widely used algorithm for background adaptation [SG99] is shown (See figure 10). This algorithm cannot deal with sudden global changes and one can see that nearly the whole frame is classified as foreground when a global lighting change occurs. [1] An HMM model handles sudden global state changes better as illustrated in the figure. It has to be noted here that our tests were done offline (i.e. on digitized video files) and that in the HMM state-splitting method the first light turn-off/on event after the initialization will cause the state splitting. At subsequent times, the algorithm has already adapted to the right model. The method presented in their current form is mainly suited for indoor settings with finite number of global state changes.

[1] The exponential forgetting factor in this algorithm can be adapted to deal with rapid sudden changes in illumination at the expense of possibly missing slowly moving objects with low contrast.

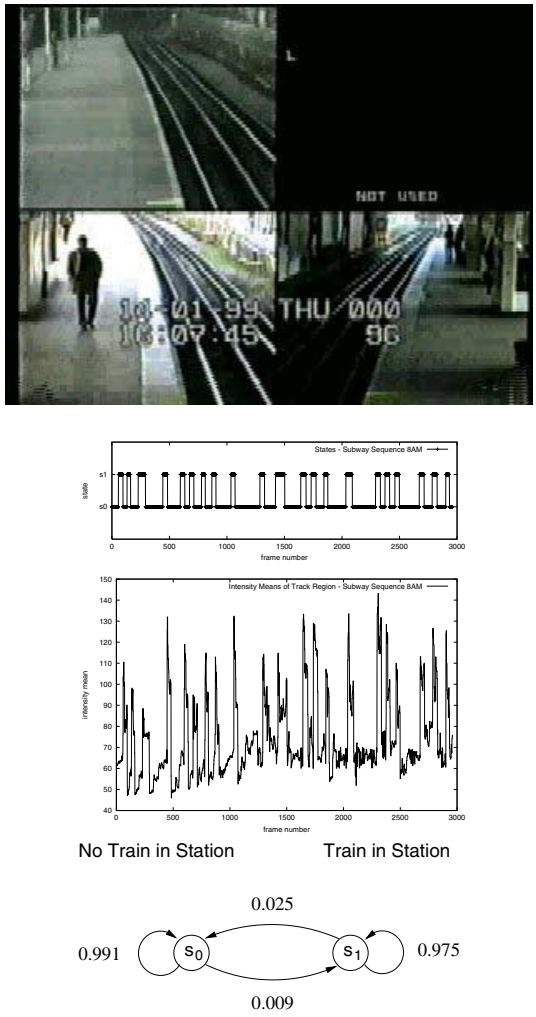


Figure 8. *Top: Example Subway data used, Middle: Plot shows 2 states, the presence (s_1) or absence (s_0) of a train in the station, Next: Intensity means of the track region over time. Bottom: A two-state model learned from annotations.*

Real-time implementation and evaluation of the online algorithm's behavior under more realistic scenarios is a subject of further research.

7 Conclusion

In this paper we described a solution to the batch and online estimation of HMM topology. For computer vision problems where the online estimation is critical, we believe that state-splitting is the most reasonable approach since it is computationally efficient. We compared several state splitting criterions such as the chi-squared goodness-of-fit test, the cross-validation criterion and the MDL and AIC criterions. It was seen that the MDL criterion performed the best in terms of minimal variance on the number of states

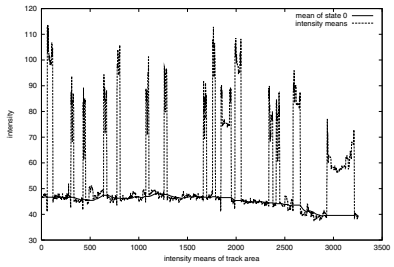


Figure 9. *Online adaptation of state parameters. The plot shows the intensity means of the track region.*

estimated. The topologies were compact and provided the right tradeoff between approximation error and model simplicity. An online version of the HMM topology estimation algorithm was also presented. Both the online versions and offline versions were tested on real data. The online version with exponential forgetting was applied to real data extracted from subway video sequences to illustrate the adaptation to change in statistics of the process. In addition, the HMM was shown to be useful at handling global illumination effects when compared to an online adaptive algorithm for background adaptation. Future work involves computational speedup of the algorithm and modification of the algorithm to address merging of states.

Figure 10. *(a) Input image, (b) Projection of our algorithm's detection map in the image, (c) Binary detection map of our algorithm, (d) Detection map using the* Stauffer-Grimson *adaptation algorithm. The rows are ordered in time.*

References

[A74] H. Akaike, A New Look At Statistical Model Identification, *IEEE Transactions on Automatic Control*, 19, pp. 716-723, 1974.

[MB98] M. Brand, "An Entropic Estimator for Structure Discovery,"l Mitsibushi Electric Research Labs, Technical Report TR-98-19, August 1998.

[BK00] M. Brand and V. Kettnaker, "Discovery and Segmentation of Activities in Video," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, pp. 844-851.

[DN96] V. V. Digalakis and L.G.Neumeyer, "Speaker Adaptation using combined transformation and Bayesian methods," *IEEE Trans. Speech and Audio Processing*, vol. 3, pp. 357-366, 1995.

[FR97] N. Friedman, S. Russel, Image Segmentation in Video Sequences: A Probabilistic Approach, *Proc. of the Thirteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, 1997.

[HL98] Q. Huo and C.H.Lee, "Online Adaptive Learning of the Correlated Continuous Density Hidden Markov Models for Speech Recognition," *IEEE Trans. Speech and Audio Processing* , vol. 6, pp. 386 - 397, 1998.

[J97] F. Jelinek, Statistical Methods for Speech Recognition, The MIT Press, Cambridge, Mass., 1997.

[K60] R.E. Kalman, A New Approach to Linear Filtering and Prediction Problems, *Transactions of the ASME*, pp. 35-45, March 1960.

[KB90] K.P. Karmann, A. von Brandt, Moving Object Recognition Using an Adaptive Background Memory, Time-varying Image Processing and Moving Object Recognition (V. Cappellini ed.), Elsevier, Amsterdam, pp. 297-307, 1990.

[MCB96] C. Montacié, M.-J. Caraty, C. Barras, Mixture Splitting Technic and Temporal Control in a HMM-Based Recognition System. *Proc. Intl. Conference on Spoken Language Processing (ICSLP)* , Vol. 2, pp. 977-980, 1996.

[N91] S. J. Nowlan, Soft Competitive Adaptation: Neural Network Learning Algorithms Based on Fitting Statistical Mixtures, Ph. D. thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, 1991.

[NH98] R. M. Neal, G. E. Hinton, A View of the EM Algorithm That Justifies Incremental, Sparse, and Other Variants, in M.I. Jordan (Ed.), Learning in Graphical Models, Kluwer, 1998.

[OS97] M. Ostendorf, H. Singer, HMM Topology Design Using Maximum Likelihood Successive State Splitting. *Computer Speech and Language*, Vol. 11 No. 1, pp. 17-41, 1997.

[R89] L.R. Rabiner, A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, *Proceedings of the IEEE* , Vol. 77, No. 2, pp. 257-286, 1989.

[RKJB00] J. Rittscher, J. Kato, S. Joga, A. Blake, A Probabilistic Background Model for Tracking, *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2000.

[R86] J. Rissanen, Stochastic Complexity and Modeling, *The Annals of Statistics*, 14, pp. 1080-1100, 1986.

[SG99] C. Stauffer, W.E.L. Grimson, Adaptive Background Mixture Models for Real-Time Tracking, *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 246-252, 1999.

[TKBM99] K. Toyama, J. Krumm, B. Brumitt, B. Meyers, Wallflower: Principles and Practice of Background Maintenance, *International Conference on Computer Vision (ICCV)*, pp. 255-261, 1999.

[TS92] J. Takami, S. Sagayama, A Successive State Splitting Algorithm for Efficient Allophone Modeling, *ICASSP 1992* , pp. 573-576, 1992.

[VEB96] Vasko R., El-Jaroudi A., Boston J., An algorithm to determine hidden Markov model topology. *ICASSP*, Vol. 6, pp. 3577-3580, 1996.

[Z93] P. Zhang, Model Selection via Multifold Cross Validation, *Annals of Statistics*, 21, pp. 299-313, 1993.