Submitted to the *IEEE Conference on Computer Vision and Pattern Recognition*, 2000.

Transformed Hidden Markov Models: Estimating Mixture Models of Images and Inferring Spatial Transformations in Video Sequences

Nebojsa Jojic, Nemanja Petrovic, Brendan J. Frey and Thomas S. Huang

Beckman Institute for Advanced Science and Technology
University of Illinois at Urbana-Champaign

Abstract

In this paper we describe a novel generative model for video analysis called the transformed hidden Markov model (THMM). The video sequence is modeled as a set of frames generated by transforming a small number of class images that summarize the sequence. For each frame, the transformation and the class are discrete latent variables that depend on the previous class and transformation in the sequence. The set of possible transformations is defined in advance, and it can include a variety of transformation such as translation, rotation and shearing. In each stage of such a Markov model, a new frame is generated from a transformed Gaussian distribution based on the class/transformation combination generated by the Markov chain. This model can be viewed as an extension of a transformed mixture of Gaussians [1] through time. We use this model to cluster unlabeled video segments and form a video summary in an unsupervised fashion. We also use the trained models to perform tracking, image stabilization and filtering. We demonstrate that the THMM is capable of combining long term dependencies in video sequences (repeating similar frames in remote parts of the sequence) with short term dependencies (such as short term image frame similarities and motion patterns) to better summarize and process a video sequence even in the presence of high levels of white or structured noise (such as foreground occlusion).

1 Introduction

Classical video analysis algorithms are based on short term processing, such as video summarization based on shot cut detection, image stabilization based on optical flow computation and image denoising by filtering out high frequencies in time and space. In specialized applications, especially in HCI and recently in image databases, researchers have often used more elaborate models that combine high level concepts with low-level observations, such as Bayesian nets and Hidden Markov Models (HMM) (e.g., [2] and [3]). We believe that general video analysis algorithms should use similar learning techniques in order to gain global knowledge about a video sequence before performing analysis and processing, including even local stabilization techniques and filtering.

We are interested in developing algorithms that can learn models of different types of object from unlabeled frames in a video sequence that include background clutter, occlusion and spatial transformations, such as translation, rotation and shearing. For example, Fig. 1 shows some 44×28 grayscale frames from a video sequence (Movie 1) of a person walking across a cluttered background while changing his pose. Traditional video summary algorithms based on scene cut or large motion detection would fail to extract interesting frames from the sequence as the amount of motion is roughly equal everywhere in the sequence.

We would like to form a different type of video summary that consist of models for different head poses and the characteristics of the global motion of the face. Instead of using solely short term changes as the criterion for summarizing the video, we choose to define video summarization as the task of parameter estimation in a generative model that is capable of randomly generating a sequence of similar characteristics. Generative models also allow estimation of the likelihood of a new sequence and thus they can be used to recognize new sequences similar in appearance or in motion pattern. The structure of the generative model should be chosen in a way that minimizes the number of parameters to be estimated and reduces chances of sticking into a local maximum during training. It is likely that such a structure would parallel the interesting structure in real scenes, and thus the inference in different stages of a generative model would result in performing some of the typical image processing tasks.

In this paper, we illustrate this approach and develop a general video analysis tool that extracts long and short term similarities in video using a novel gen-

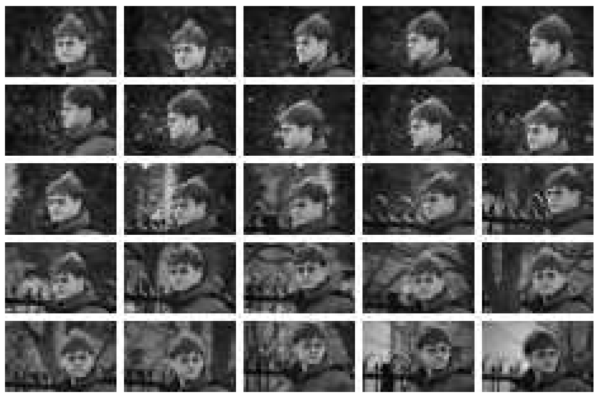

Figure 1: A subset of frames from a video sequence of a head which appears at different positions and has different poses.

erative model, called the transformed hidden Markov model (THMM). In the next section we describe the parameters and hidden variables in this model, and describe how to infer the hidden variables when the parameters are known. Then, in Section 3, we show how to learn the optimal parameters of the model for a given sequence using the EM algorithm [4]. We compare THMM with other clustering approaches on a tough toy problem in Section 4 and illustrate joint learning of object appearances, tracking, stabilization and filtering on natural outdoor sequences in Section 5.

2 The Transformed Hidden Markov Model

We assume the following generative model (Fig. 2) for a video sequence. Each frame corresponds to one of the finite number of clusters ($c = 1, ..., C$) which are characterized by a Gaussian distribution with a diagonal covariance matrix (Section 2.1). It is assumed that when forming an image in a sequence, a latent image is first drawn from this distribution and then it is transformed by one of the finite number of possible transformations $\mathbf{\Gamma}_\ell$, $\ell = 1, ..., L$.

Although the true transformation variables are real-valued, real-valued latent variables introduce complicated integrals (instead of summations) into the inference problem. So, we will assume there is a fixed set of possible transformations and that this set is specified beforehand. Also, the algorithm is simplified by assuming that the vector of pixel values for the transformed image is obtained by multiplying the vector of pixel values for the latent image by a matrix ($\mathbf{\Gamma}_\ell$). For most transformations, each pixel in the observed image will depend on only a small number (say, 4)

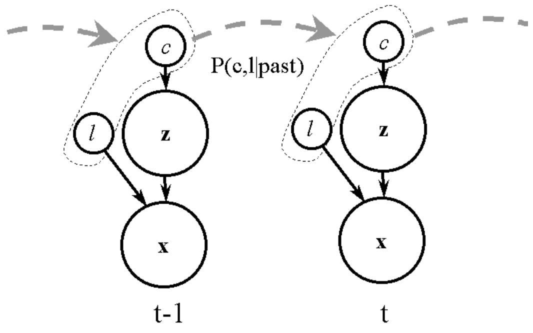

Figure 2: The THMM model

of pixels in the latent image. So, $\mathbf{\Gamma}_\ell$ is a sparse matrix with a number of columns equal to the number of pixels in the latent image and a number of rows equal to the number of pixels in the observed image. This permits a broad class of transformations, including translation, scale, in-plane rotation, and shearing.

In the sequence in Fig. 1, for example, the classes should represent different head poses, and the set of transformations $\{\mathbf{\Gamma}_\ell\}$ can be the set of possible translations defining where the face appears in the image.

Each frame t can be described by a state $s_t = (c_t, l_t)$ containing the class index and the transformation index. The actual frame x_t is generated randomly from the distribution $p(x_t|s_t) = p(x_t|c_t, l_t)$ associated with the state. For this distribution, we use the transformed mixture of Gaussians (TMG) model [1] which we describe next.

2.1 Observation Likelihood

The probability density of the vector of pixel values $\mathbf{z}$ for the latent image corresponding to cluster c is

$$p(\mathbf{z}|c) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_c, \mathbf{\Phi}_c). \tag{1}$$

$\boldsymbol{\mu}_c$ is the mean of the latent image and $\mathbf{\Phi}_c$ is a diagonal covariance matrix that specifies the variability of each pixel in the latent image. The data we are interested in here is high dimensional, so we use a diagonal covariance matrix instead of a full covariance matrix. In general, different clusters will represent different types of latent image and the corresponding noise variance maps will specify which regions are not well modeled — *e.g.*, background clutter.

The probability density of the vector of pixel values $\mathbf{x}$ for the image corresponding to transformation ℓ and latent image $\mathbf{z}$ is

$$p(\mathbf{x}|\ell, \mathbf{z}) = \mathcal{N}(\mathbf{x}; \mathbf{\Gamma}_\ell \mathbf{z}, \mathbf{\Psi}), \tag{2}$$

where $\mathbf{\Psi}$ is a diagonal covariance matrix that specifies the noise on the observed pixels.

The variances $\mathbf{\Phi}_c$ of the pixels in the latent image are quite different from the variances $\mathbf{\Psi}$ of the pixels in the observed image. $\mathbf{\Phi}_c$ models the noise in the pixel values for cluster c and this coheres to the latent image under transformations. In contrast, $\mathbf{\Psi}$ models noise in the observed pixels and this noise does not depend on the transformations.

The joint distribution of the observed and latent image, given the state of the Markov Chain at the time t is then:

$$p(\mathbf{x_t}, \mathbf{z_t}|\ell_t, c_t) = \mathcal{N}(\mathbf{x_t}; \mathbf{\Gamma}_{\ell_t}\mathbf{z_t}, \mathbf{\Psi})\mathcal{N}(\mathbf{z_t}; \boldsymbol{\mu}_{c_t}, \mathbf{\Phi}_{c_t}). \quad (3)$$

The latent image $\mathbf{z}$ can be integrated over in closed form [1]:

$$p(\mathbf{x}_t|\ell_t, c_t) = \mathcal{N}(\mathbf{x}_t; \mathbf{\Gamma}_{\ell_t}\boldsymbol{\mu}_{c_t}, \mathbf{\Gamma}_{\ell_t}\mathbf{\Phi}_{c_t}\mathbf{\Gamma}'_{\ell_t} + \mathbf{\Psi}), \quad (4)$$

where " $'$ " indicates transpose.

2.2 Transition probabilities

The frames in a video sequence should not be treated as independent. For example, if several head poses are used to represent all frames in the video sequence, then consecutive frames are likely to have the same head pose, and when the pose changes in the next frame, not all the poses are equally likely. Also, the transformations in two consecutive frames are likely to be similar.

There are several types of temporal dependencies that can be incorporated into the THMM. To reduce the number of parameters, we can assume that the state at time $t-1$ contains all the necessary information about the past, i.e. $p(s_t|s_{t-1}, s_{t-2}, ...) = p(s_t|s_{t-1})$. Also, in most cases it is reasonable to assume that while the motion (transformation transition) may or may not depend on the image class, the class transition is independent from the current transformation index, i.e., $p(s_t|s_{t-1}) = p(c_t|c_{t-1})p(\ell_t|\ell_{t-1}, c_{t-1})$. Such a model would have $C^2 + L^2C$ transition probabilities to be estimated in the training procedure. The number of parameters is dominated by the term L^2, since there is typically many more possible transformations than classes. To further reduce the number of parameters, we limit the probabilistic model to relative motion defined by a mapping $m(\ell_t, \ell_{t-1})$.

For example, in the case of image translations, this mapping would simply correspond to the relative shift between two global translations. If a total of M vertical and M horizontal shifts are possible, then the total number of transformations is $L = M^2$ and the transformations can be sorted so that $\ell = iM + j$, where i, j denote indices of the appropriate vertical and horizontal shifts. Then, we can define the mapping as $m(\ell_t, \ell_{t-1}) = \sqrt{(i_t - i_{t-1})^2 + (j_t - j_{t-1})^2}$, if we are interested only in the intensity of the relative motion, or as the vector $m(\ell_t, \ell_{t-1}) = (i_t - i_{t-1}, j_t - j_{t-1})$, if the direction of the motion is also important. In a similar fashion the distance measure for other types of transformations can be defined. Then, we can define:

$$\begin{aligned} p(s_t|s_{t-1}) &= p(c_t|c_{t-1})p(m(\ell_t, \ell_{t-1})), \quad or, \quad (5) \\ p(s_t|s_{t-1}) &= p(c_t|c_{t-1})p(m(\ell_t, \ell_{t-1})|c_{t-1}), \quad (6) \end{aligned}$$

depending on whether or not we want to allow different motion characteristics for different image classes. By assuming small motion between consecutive frames and setting $p(m) = 0$ for $|m| > thr$, the number of parameters and the computational load of the inference procedure can be drastically reduced.

Another useful variation to defining the transformation transition is the use of an autoregressive model to capture higher order dynamics in a sequence.

The full Transformed Hidden Markov Model has the following parameters: $\boldsymbol{\mu}_c$, for $c = 1, ..., C$ - the mean images for C classes; Φ_c, defining the levels of uncertainty for different pixels for each class; Ψ, the diagonal covariance matrix describing the sensor noise; π_s where $s = (c, \ell)$, the prior probabilities of different states; and finally the transition probabilities $a_{s,s'} = p(s_t = s'|s_{t-1} = s)$, that can be factorized as shown above. The hidden variables in the model are the states s_t and the latent images z_t.

In this generative model, given the previous state, the cluster index c_t and the transformation index ℓ_t are drawn randomly from $a_{s_{t-1},s_t} = p(c_t, \ell_t|c_{t-1}, \ell_{t-1})$. Then, a latent image is drawn from $p(\mathbf{z}_t|c_t)$, and then the final frame is drawn from $p(\mathbf{x}_t|\ell_t, \mathbf{z}_t)$. The process is repeated until the end of the sequence.

If we have the optimal THMM parameters for a given sequence (and they can be learned as we will show later), several interesting image processing tasks can be defined simply as inference of the hidden variables in THMM (Fig. 3) Inferring the mean of the Gaussian in Eq. 2 corresponds to the removal of sensor noise. Inference of the transformation index is equivalent to object tracking. Since the model assumes that the latent image $\mathbf{z}$ was produced before applying the transformation, inference of $\mathbf{z}$ (most likely image given in Eq. 26) results in image stabilization (a frame can also be stabilized by applying an inverse of the most likely transformation). As a probabilistic model, the trained THMM can also be used for video coding and recognition of similar video sequences.

In the next two sections we describe how to infer the states in this model and how to train the model using the EM algorithm.

2.3 Inferring states and computing the likelihood of the observed sequence

The joint log likelihood of a video sequence $\mathbf{X} = \{\mathbf{x}_t\}_{t=1,\dots,T}$ and the state sequence $S = \{s_t\}_{t=1,\dots,T}$ is:

$$\log p(\mathbf{X}, S) = \log \pi_{s_0} \sum_{t=0}^{T} (\log p(x_t|s_t) + \log a_{s_t,s_{t+1}}). \tag{7}$$

The likelihood of the video sequence $\mathbf{X}$ is computed by summing over all possible state sequences S:

$$p(\mathbf{X}) = \sum_S p(\mathbf{X}, S). \tag{8}$$

When the model has C clusters and L transformations, the number of possible state sequences S is equal to $(CL)^T$ and thus computing the likelihood according to the previous equation is impractical. However, it is possible to infer the states and compute the sequence likelihood for given parameters in a much more efficient way using the forward-backward algorithm.

The "forward probability distribution" is defined as:

$$\alpha_t(s) = p(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t, s_t = s), \tag{9}$$

which can be computed efficiently using the recursion,

$$\alpha_1(s) = \pi_s p(\mathbf{x}_1|s), \tag{10}$$

$$\alpha_{t+1}(u) = \Big[\sum_s \alpha_t(s) a_{s,u}\Big] p(\mathbf{x}_{t+1}|u). \tag{11}$$

The observation probability density $p(\mathbf{x}_{t+1}|u)$, where u denotes a particular combination (c, ℓ) is given in Eq. 4.

Similarly, the "backward probability distribution" defined as,

$$\beta_t(s) = p(\mathbf{x}_{t+1}, \mathbf{x}_{t+2}, ..., \mathbf{x}_T, s_t = s), \tag{12}$$

can be computed using the recursion:

$$\beta_T(s) = 1. \tag{13}$$

$$\beta_t(s) = \sum_u a_{s,u} p(\mathbf{x}_{t+1}|u) \beta_{t+1}(u) \tag{14}$$

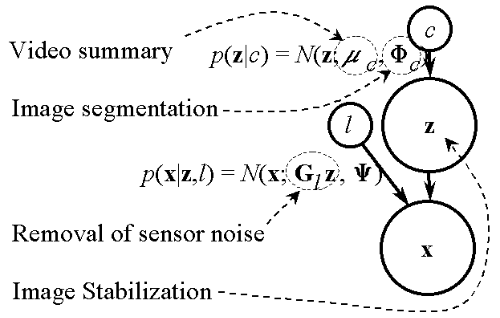
Video summary
$p(\mathbf{z}|c) = N(\mathbf{z}; \mu_c, \Phi_c)$
Image segmentation
$p(\mathbf{x}|\mathbf{z},l) = N(\mathbf{x}; \mathbf{G}_l \mathbf{z}, \Psi)$
Removal of sensor noise
Image Stabilization
c
l
z
x

Figure 3: Video analysis as learning and inference in a THMM model

The uniform initialization in the backward recursion makes no assumptions about the unobserved images beyond the end of the sequence.

It can be readily seen that:

$$p(\mathbf{X}, s_t = s) = \alpha_t(s)\beta_t(s). \tag{15}$$

Thus, after computing forward and backward probability variables, the likelihood of a sequence can be computed as,

$$p(X) = \sum_s \alpha_T(s), \tag{16}$$

or compute the probability of each state at time t:

$$P(s_t = s|\mathbf{X}) = \frac{\alpha_t(s)\beta_t(s)}{\sum_u \alpha_t(u)\beta_t(u)}. \tag{17}$$

We can also compute the distribution over two consecutive states given the observations as:

$$P(s_t = s, s_{t+1} = u, \mathbf{X}) = \frac{\alpha_t(s) a_{s,u} p(\mathbf{x}_{t+1}|u) \beta_{t+1}(u)}{p(\mathbf{X})}. \tag{18}$$

Computing the conditional likelihood $p(\mathbf{x}_t|s_t = s)$ for all possible states s takes $O(CLN)$ time where N is the number of pixels in the frame [1], and the computation of forward and backward variables takes $O((CL)^2T)$ computations for the full transition model which can be further reduced by assuming small relative motion as discussed in the previous section. Summing over the states in the forward algorithm can also be replaced by maximizations (Viterbi algorithm) if the goal of the inference is the sequence that maximizes the total likelihood, as opposed to the most likely state for a given time frame.

3 The EM Algorithm for a Transformed Hidden Markov Model

In this section, we describe an iterative expectation maximization (EM) algorithm for estimating the maximum likelihood parameters of a THMM. In the E step the expectations conditioned on the current parameters and the observed data are computed (for example, the distributions over hidden states s_t and latent images $\mathbf{z}_t$ are inferred from the given data). While the E step keeps the model parameters constant, the M step uses only the sufficient statistics from the E step to find the parameters that maximize the log likelihood of the data. The change of model parameters requires re-computation of the statistics in the E step, and thus the need for iterating. This procedure consistently increases the likelihood of the training data.

In speech processing literature, the EM algorithm for HMMs with mixture of Gaussians as observation distributions was proven to consist of computing relative frequencies of state transitions (just as for the fully discrete HMMs in [5], [4]) and re-estimating the observation mixture parameters using the state responsibilities similar to Eq. 18 (see [6], for example). Our observation distribution is also a Gaussian for each state s (Eq. 4), but the parameters for different states are tied. For example, for a fixed class c, all observation means are the transformed versions of each other. The way to handle this was described in [1], and for THMM we just need to compute state responsibilities using the forward-backward algorithm, instead of treating the frames independently.

3.1 M-Step: The Maximizations

$\langle \cdot \rangle = \frac{1}{T}\sum_{t=1}^{T}(\cdot)$ indicates a statistic sufficient for the M-Step, which is computed by averaging over the training set as described in the next section; $\text{diag}(\mathbf{A})$ gives a vector containing the diagonal elements of matrix $\mathbf{A}$; $\text{diag}(\mathbf{a})$ gives a diagonal matrix whose diagonal contains the elements of vector $\mathbf{a}$; and $\mathbf{a}\circ\mathbf{b}$ computes the element-wise product of vectors $\mathbf{a}$ and $\mathbf{b}$. We denote the updated parameters by "$\tilde{\ }$".

$$\tilde{a}_{s,u} = \frac{\langle P(s_t = s, s_{t+1} = u, \mathbf{X})\rangle}{\langle P(s_t = s, \mathbf{X})\rangle}, \qquad (19)$$

$$\tilde{\boldsymbol{\mu}}_k = \frac{\langle P(c_t = k|\mathbf{X})\text{E}[\mathbf{z}|c_t = k, \mathbf{X}]\rangle}{\langle P(c = k|\mathbf{x}_t)\rangle}, \qquad (20)$$

$$\tilde{\boldsymbol{\Phi}}_k = \text{diag}\Big(\frac{\langle P(c_t = k|\mathbf{X})\text{E}[(\mathbf{z}-\boldsymbol{\mu}_k)\circ(\mathbf{z}-\boldsymbol{\mu}_k)|c_t = k, \mathbf{X}]\rangle}{\langle P(c_t = k|\mathbf{X})\rangle}\Big), \qquad (21)$$

$$\tilde{\boldsymbol{\Psi}} = \text{diag}\big(\langle \text{E}[(\mathbf{x}_t - \boldsymbol{\Gamma}_\ell\mathbf{z})\circ(\mathbf{x}_t - \boldsymbol{\Gamma}_\ell\mathbf{z})\big|\mathbf{X}]\rangle\big). \qquad (22)$$

The prior distribution of the states for the first frame can also be estimated if the training is performed on multiple sequences. Otherwise, we can just assume a uniform prior. In order to avoid over fitting the noise variances, it is sometimes useful to set the diagonal elements of $\boldsymbol{\Phi}_k$ and $\boldsymbol{\Psi}$ that are below some ϵ equal to ϵ. When the transition probability distribution is factorized as suggested in Section 2.2, the re-estimation formula in Eq. 19 can be simply adjusted to do appropriate averaging of the evidences of the transitions that are treated as the same in the model. For example, to estimate the relative motion distribution in Eq. 5, we compute:

$$\tilde{p}(m) = \frac{\langle \sum_{s_t, s_{t+1}} P(s_t, s_{t+1}, \mathbf{X})\delta(m(l_t, l_{t+1}) - m)\rangle}{\langle \sum_{s_t} P(s_t, \mathbf{X})\rangle}, \qquad (23)$$

where $\delta(0) = 1$ and $\delta(m) = 0$ if $m <> 0$.

3.2 E-Step: The Expectations

The forward-backward algorithm is used to compute for each time instant t the state responsibilities $P(s_t|\mathbf{X})$. These are then marginalized for each value of c and normalized for each value of c to obtain $P(c_t = c|\mathbf{X})$ and $P(\ell_t = \ell|c_t = c, \mathbf{X})$. Then, the sufficient statistics are computed from [1]:

$$\boldsymbol{\Omega}_{c\ell} = \text{COV}(\mathbf{z}|c, \ell, \mathbf{x}) = (\boldsymbol{\Phi}_c^{-1} + \boldsymbol{\Gamma}'_\ell\Psi^{-1}\boldsymbol{\Gamma}_\ell)^{-1}. \qquad (24)$$

$$\text{E}[\mathbf{z}|c, \ell, \mathbf{x}_t] = \boldsymbol{\Omega}_{c\ell}\boldsymbol{\Gamma}'_\ell\boldsymbol{\Psi}^{-1}\mathbf{x}_t + \boldsymbol{\Omega}_{c\ell}\boldsymbol{\Phi}_c^{-1}\boldsymbol{\mu}_c, \qquad (25)$$

$$\text{E}[\mathbf{z}|c_t, \mathbf{X}] = \sum_{\ell=1}^{L} P(\ell|c_t, \mathbf{X})\text{E}[\mathbf{z}|c_t, \ell, \mathbf{x}_t]. \qquad (26)$$

$$\begin{aligned}\text{E}[(\mathbf{z}-\boldsymbol{\mu}_c)\circ(\mathbf{z}-\boldsymbol{\mu}_c)|c_t, \mathbf{X}] &= \sum_{\ell=1}^{L} P(\ell|c_t, \mathbf{X}) \\ &\cdot \big\{(\text{E}[\mathbf{z}|c_t, \ell, \mathbf{x}_t] - \boldsymbol{\mu}_c)\circ(\text{E}[\mathbf{z}|c_t, \ell, \mathbf{x}_t] - \boldsymbol{\mu}_c) \\ &\quad + \text{diag}(\boldsymbol{\Omega}_{c_t\ell})\big\}\end{aligned} \qquad (27)$$

$$\begin{aligned}\text{E}[(\mathbf{x}_t - \boldsymbol{\Gamma}_\ell\mathbf{z})\circ(\mathbf{x}_t - \boldsymbol{\Gamma}_\ell\mathbf{z})|\mathbf{X}] &= \sum_{c=1}^{C}\sum_{\ell=1}^{L} P(c_t = c, \ell_t = \ell|\mathbf{X}) \\ &\cdot \big\{(\mathbf{x}_t - \boldsymbol{\Gamma}_\ell\text{E}[\mathbf{z}_t|c, \ell, \mathbf{X}])\circ(\mathbf{x}_t - \boldsymbol{\Gamma}_\ell\text{E}[\mathbf{z}_t|c, \ell, \mathbf{X}]) \\ &\quad + \text{diag}(\boldsymbol{\Gamma}_\ell\boldsymbol{\Omega}_{c\ell}\boldsymbol{\Gamma}'_\ell)\big\}.\end{aligned} \qquad (28)$$

4 Extracting Shapes and Learning Motion Pattern From Synthetic Data

Fig. 4 shows 30 11x11 frames from a 200 frames long 'pac-man sequence'. Each frame contains one of

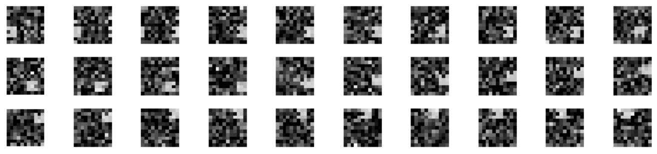

Figure 4: 30 frames from a 200-frame sequence of a pac-man game on an 11x11 grid

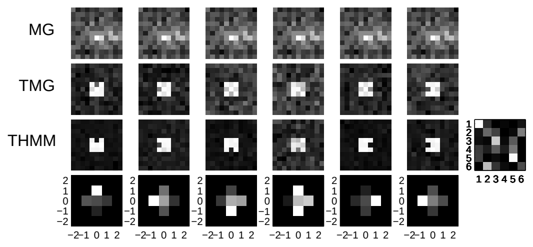

Figure 5: Cluster means resulting form MG, TMG and THMM training; the relative motion patterns for different clusters in the trained THMM (last row) and the class transition matrix of the THMM (right)

the four basic 3x3 pac-man shapes discernable by the orientation of the mouth. The levels of background noise and sensor noise make it difficult to recognize the shapes when the frames are studied individually. However, the pac-man's motion is highly regimented: it always moves by at most one pixel in the direction of his mouth (probability of staying put is 0.2), and occasionally (with probability 0.75) it makes a left turn by appropriately changing the mouth orientation.

A model that does not allow image transformations, such as an ordinary mixture of 6 Gaussians shown in the first row of Fig. 5 will fail to cluster the data. Even using a tranformation invaraint clustering algorithm that treats frames as independent, it is difficult to tell the classes apart, as the basic shapes differ in less than 2% of the number of observed pixels. For example, in the second row of Fig. 5, we show the six classes learned by training a TMG (until convergence) assuming a uniform prior over all 11x11=121 possible translations. While the classes are no longer blurred over the whole image, the background clutter is still not sufficiently suppressed and the shapes are not completely separated. In fact one shape is missing entirely (mouth looking down).

We refined the learned classes by training a THMM to see if it can detect the regimented class/transformation transition and use it to better cluster the data and estimate the cluster means. We initialized the THMM model with learned clusters from TMG training and assumed the transition model of Eq. 6. In the second row of Fig. 5 we show the learned cluster means after 14 iterations of EM. The background clutter is further suppressed, and one of the ambiguous TMG classes was refined into the missing 'down shape.' In the third row of the figure we show the motion distribution for each cluster as a gray level image in which the intensity is proportional to the probability of the corresponding relative shift. These motion likelihood images show the correlation between the direction of the mouth in the shape and the direction of the motion. The class transition probability matrix (shown as a gray level image) captures the left turns in the data as the transition between appropriate appearances. Since 6 classes were used in the model and there are only 4 distinct shapes in the data, 2nd and 6th mean are similar and there is a high probability of switching between them; and the 4th class is estimated to be very unlikely, as the 4 column of the class transition matrix idicates that none of the classes are likley to change into the 4th shape.

5 Extracting Structure from Outdoor Video Sequences with Cluttered Background and Foreground Obstructions

Fig. 1 shows 25 examples from a 400-frame video sequence (video 1) where the individual changed his pose. We trained a THMM containing $C = 5$ clusters and $L = 625$ translation transformations (25 horizontal shifts and 25 vertical shifts) using 30 iterations of the EM algorithm. We assumed the transition model of Eq. 5. Fig. 6 shows how the THMM model summarized the sequence. In the first row of the figure we show the 5 cluster means, and the second row contains the pixel variances for the clusters, while the last image illustrates the probability distribution of the translational part of the head motion (the larger shifts than 3 pixels turned out to be highly unlikely). For or all but one cluster, the learned cluster noise map illustrates that the model learned to segment the object from the background by realizing that the background is much more varying (in cases when the background is uniform, or when it moves together with the object, this would not be the case). The cluster variance map shows how the model scales the contributions of different pixels when matching a class and tracking the object.

The motion characteristics in this summary are the statistics of the object motion as opposed to an optical flow map, for example, that gives local motion

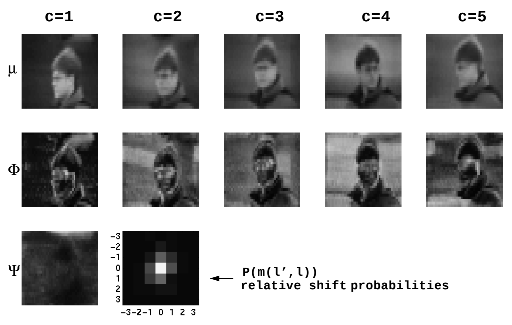

Figure 6: Summarizing the sequence (Movie 1) by the learned THMM parameters

estimates. This richness and compactness of knowledge about the sequence is the consequence of summarizing the video with the global goal of maximizing the likelihood of the data with respect to a structured generative model, rather than with local motion characteristics or pixel histograms in mind.

As illustrated in Fig. 3, the trained model can be used for a variety of image procssing tasks. In Movie 2, we show the result of image stabilization performed on the sequence in Movie 1 by infering the most likely latent image $\mathbf{z}_t$, for each frame (Eq. 26). To illustrate THMM as a generative model, in Movie 3, we continue the original sequence with generated frames from the THMM model with neglected noise (the frames contain transformed means in Fig. 6 and the class/transformation transition is governed by the model estimated from the original sequence).

Next, we illustrate noise removal using THMM inference. In Movie 4, we show three sequences running synchronously. The first sequence was produced by adding very high level of white noise to the sequence in Movie 1. The second sequence contains the frames with removed estimated sensor noise $\mathbf{\Psi}$. The third sequence contains the result of removing also the estimated cluster nose $\mathbf{\Phi}$, which means that each frame is replaced by the transformed mean of the most likely cluster. Finally, the last sequence shows for each frame the most likely latent image $\mathbf{z}$ which results in joint stabilizeation and sensor noise suppresion. The model used for processing was trained on the noisy sequence itself, *not* on the clean sequence in Movie 1. THMM was able to suppress the noise and learn clean data means (as can be seen in the third sequence in Movie 3). As can be seen in the third sequence, stabilization still works most of the time, and as inherent to the framework, the stabilization error in one frame does not ruin the stabilization in the rest of the sequence. Note that estimation of local motion in presence of this level of noise would be too unreliable to perform image stabilization using traditional techniques.

In the second example of noise suppression, we degraded the original sequence by placing a static dark bar in front of both the background and the face (the first row in Figure 7 and the first sequence in Movie 4). We then performed 30 iterations of THMM learning assuming a single class and the set of transformations as before. In Fig. 7 we show the resulting cluster mean and variances and the sensor noise estimate. The cluster mean and variances show no presence of the black bar and the observation noise $\mathbf{\Psi}$ was estimated to be very high in the bar region even though these pixels had almost constant intensities in the image! This happened because the model fixated on the face which was the largest object of relatively constant appearance, even though it was partially occluded in many frames. During the training the missing data was filled in with average appearance of the face in the rest of the sequence (which was possible as the bar did not always occlude the same part of the face). As the face and the background moved with respect to the bar, THMM could not properly predict the observed pixels in the bar using its single class template and cluster pixel variances, and thus increased the sensor noise in that region. Removing the sensor noise (second sequence in Movie 4 and Fig 7), and stabilizing the sequence (the third sequence and Fig. 7) using THMM inference resulted in suppressing the bar, i.e., when inferring the most likely latent image $\mathbf{z}$ given the observed frame, the observed pixels in the bar where weighted less than the expected values from the cluster mean, which in the denoised sequence looks as if the black bar was made a transparent.

In the final experiment, we set up a mock video surveillance system assuming that for the best angle of view the camera had to look at the street through the bars and circular ornaments of a metal gate. A short video sequence (some of the frames shown in the upper row Fig. 8) shot in this way contained a truck that drove from right to left. We first trained a regular 1-class TMG model (assuming that frames are independent of each other), and the resulting mean and variances are shown in Fig. 9 (left). The set of transformations contained all possible horizontal shifts in the 60x29 frames, and each translation matrix was defined with 'wrap around'. Even though TMG has the capability of normalizing for the transformations in the data, it focused on the static background and the gate, simply because the foreground bars and circular ornaments occupied too many pixels. In fact,

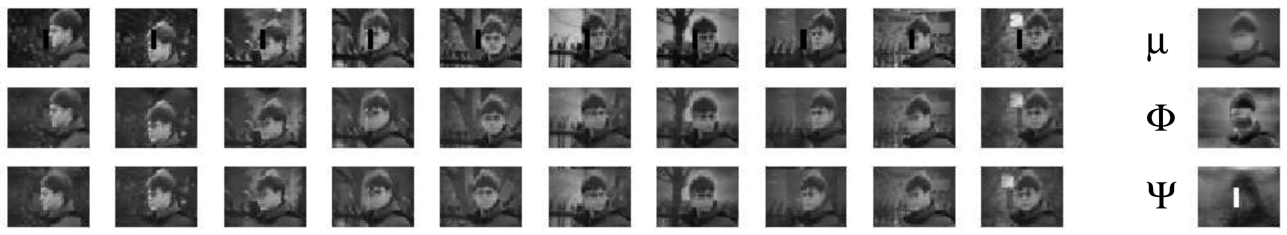

Figure 7: Several frames from a sequence degraded with a black static obstruction occluding the object of interest, and the result of suppressing the distraction (second row) and image stabilization (last row) using a THMM model (right) learned from the degraded sequence

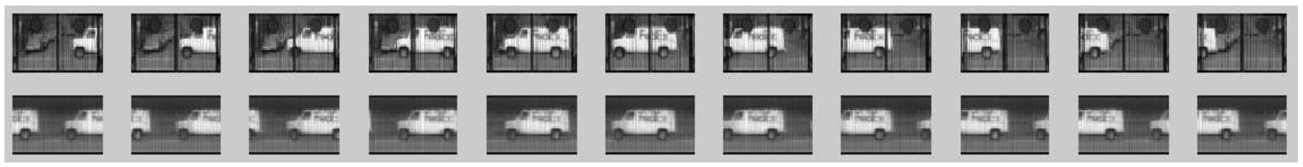

Figure 8: Learning the model and tracking an object moving between a static background and an obstructing static foreground object (the bar gate)

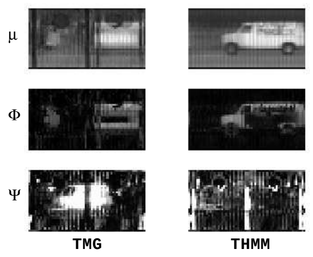

Figure 9: Learned means and variances for a TMG model and a THMM model with the relative motion distribution favoring motion to the right

even applying THMM blindly to this problem produces a similar result, as THMM can also decide that there is no global motion in the video. So, we defined a prior distribution over the relative motion $m(l', l)$ that disallows rest and motion to the right and used it in THMM training. The prior gave equal weights to the motion to the left regardless of the intensity. In this way we told the learning algorithm to look for the objects moving to the left. The learned mean and variances are shown in Fig. 9 (right). Even though parts of the truck were occluded in *each* frame, THMM was able to integrate the truck's appearance over all frames and suppress the gate parts. In the second row of Fig. 8, we demonstrate tracking in this sequence - at each time frame the image contains the cluster mean shifted to the most likely position for that time, given the *whole* observed sequence. As the truck disappears from the screen, the tracking slows down, as the forward-backward estimation of the state sequence forces the motion to continue, but the likelihood of the data reduces (since the assumed transformations wrap around the vertical edges of the frame, the data does not offer evidence for the appropriate shift, as THMM expects to see the missing part of the truck at the opposite side of the frame).

6 Summary

We presented a new generative model for video analysis. Using learning and inference in this model we were able to jointly suppres the backgund clutter and foregorund occlusions, learn object appearances and track the objects in the sequences. The model can be naturally extended to include multiple objects in different layers and more complex appearance models such as factor analysis. Our future research will also focus on fast variational techniques and multiresolution approaches to reduce the computational expense of the exact EM algorithm that we used in our experiments (for example, the training for the last experiment took 6 minutes per frame per iteration using our loop rich Matlab code on an older SGI computer).

References

[1] B. Frey and N. Jojic, "Learning mixture models of images and inferring spatial transformations using the

em algorithm," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR '99), Ft. Collins, Colorado*, June, 1999.

[2] N. Vasconcelos and A. Lippman, "Bayesian modeling of video editing and structure: Semantic features for video summarization and browsing," in *Proceedings of the IEEE Intl. Conference on Image Processing*, Chicago, IL, Oct. 1998, vol. 2, pp. 550–555.

[3] M. Naphade, T. Kristjansson, B. Frey, and T. S. Huang, "Probabilistic multimedia objects (multijects): A novel approach to indexing and retrieval in multimedia systems," in *Proceedings of the fifth IEEE International Conference on Image Processing*, Chicago, IL, Oct 1998, vol. 3, pp. 536–540.

[4] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Proceedings of the Royal Statistical Society*, vol. B-39, pp. 1–38, 1977.

[5] L. E. Baum, "An inequality and associated maximization technique in statistical estimation for probabilistic functions of markov processes," *Inequalities*, vol. 3, pp. 1–8, 1972.

[6] B. H. Juang, S. E. Levinson, and M. M. Sondhi, "Maximum likelihood estimation for multivariate mixture observations of markov chains," *IEEE Trans. Information Theory*, vol. IT-32(2), pp. 307–309, 1982.