Modeling, Clustering, and Segmenting Video with Mixtures of Dynamic Textures

Antoni B. Chan, Member, IEEE, Nuno Vasconcelos, Member, IEEE

Abstract

A dynamic texture is a spatio-temporal generative model for video, which represents video sequences as observations from a linear dynamical system. This work introduces the mixture of dynamic textures, a statistical model for an ensemble of video sequences that is sampled from a finite collection of visual processes, each of which is a dynamic texture. An expectation-maximization (EM) algorithm is derived for learning the parameters of a mixture of dynamic textures, and the model is related to previous works in linear systems, machine learning, time-series clustering, and computer vision. Through experimentation, it is shown that the mixture of dynamic textures is a suitable representation for both the appearance and dynamics of a variety of visual processes (e.g. fire, steam, water, clouds, trees, vehicle and pedestrian traffic, etc.) that have traditionally been challenging for computer vision. When compared with traditional representations based on optical flow or other localized motion representations, the dynamic texture mixture achieves substantially superior performance in the problems of clustering and segmenting video of such processes.

I. INTRODUCTION

One family of visual processes that has relevance for various applications of computer vision is that of, what could be loosely described as, visual processes composed of \textit{ensembles of particles subject to stochastic motion}. The particles can be microscopic, e.g plumes of smoke, macroscopic, e.g. leaves and vegetation blowing in the wind, or even objects, e.g. a human crowd, a flock of birds, a traffic jam, or a bee-hive. The applications range from remote monitoring for the prevention of natural disasters, e.g. forest fires, to background subtraction in challenging environments, e.g. outdoors scenes with vegetation, and various type of surveillance, e.g. traffic monitoring, homeland security applications, or scientific studies of animal behavior.

Despite their practical significance, and the ease with which they are perceived by biological vision systems, the visual processes in this family still pose tremendous challenges for computer
vision. In particular, the *stochastic nature* of the associated motion fields tends to be *highly challenging for traditional motion representations* such as optical flow [1]–[4], which requires some degree of motion smoothness, parametric motion models [5]–[7], which assume a piece-wise planar world [8], or object tracking [9]–[11], which tends to be impractical when the number of subjects to track is large and these objects interact in a complex manner.

The main limitation of all these representations is that they are inherently *local*, aiming to achieve understanding of the whole by modeling the motion of the individual particles. This is *contrary to how these visual processes are perceived by biological vision*: smoke is usually perceived as a whole, a tree is normally perceived as a single object, and the detection of traffic jams rarely requires tracking individual vehicles. Recently, there has been an effort to advance towards this type of *holistic modeling*, by viewing video sequences derived from these processes as *dynamic textures* or, more precisely, samples from stochastic processes defined over space and time [12]–[16]. In fact, the dynamic texture framework has been shown to have great potential for video synthesis [12], motion segmentation [13], [14], [17], and video classification [15], [16]. This is, in significant part, due to the fact that the underlying generative probabilistic framework is capable of 1) abstracting a wide variety of complex motion patterns into a *simple* spatio-temporal process, and 2) synthesizing samples of the associated time-varying texture.

One major current limitation of the dynamic texture framework is its inability to account for visual processes consisting of *multiple, co-occurring, dynamic textures*. For example, a flock of birds flying in front of a water fountain, highway traffic moving in opposite directions, video containing both smoke and fire, and so forth. While, in such cases, existing dynamic texture models are inherently incorrect, the underlying generative framework is not. In fact, co-occurring textures can be accounted for by augmenting the probabilistic generative model with a discrete *hidden* variable, that has a number of states equal to the number of textures, and encodes which of them is responsible for a given piece of the spatio-temporal video volume. Conditioned on the state of this hidden variable, the video is then modeled as a simple dynamic texture.

This leads to an extension of the dynamic texture model, a *mixture of dynamic textures* [17], that we study in this work. In addition to introducing the model itself, we report on three main contributions. First, an expectation maximization (EM) algorithm is derived for maximum likelihood estimation of the parameters of a mixture of dynamic textures. Second, the relationships between this mixture model and various other models previously proposed in the
machine learning and computer vision literatures, including mixtures of factor analyzers, linear dynamical systems, and switched linear dynamic models, are analyzed. Finally, we demonstrate the applicability of the new model to the solution of traditionally difficult vision problems that range from clustering traffic video sequences to segmentation of sequences containing multiple dynamic textures.

The paper is organized as follows. In Section II, we formalize the dynamic texture mixture model. In Section III we present the EM algorithm for learning its parameters from training data. In Section IV and Section V, we relate it to previous models and discuss its application to video clustering and segmentation. Finally, in Section VI we present an experimental evaluation in the context of these applications.

II. MIXTURES OF DYNAMIC TEXTURES

In this section, we introduce the dynamic texture mixture model. For completeness, we start with a brief review of the dynamic texture model.

A. Dynamic texture

A dynamic texture [12] is a generative model for both the appearance and the dynamics of video sequences. It consists of a random process containing an observed variable \( y_t \), which encodes the appearance component (video frame at time \( t \)), and a hidden state variable \( x_t \), which encodes the dynamics (evolution of the video over time). The state and observed variables are related through the linear dynamical system (LDS) defined by

\[
\begin{align*}
    x_{t+1} &= Ax_t + v_t \\
    y_t &= Cx_t + w_t
\end{align*}
\]

where \( x_t \in \mathbb{R}^n \) and \( y_t \in \mathbb{R}^m \) (typically \( n \ll m \)). The parameter \( A \in \mathbb{R}^{n \times n} \) is a state transition matrix and \( C \in \mathbb{R}^{m \times n} \) is an observation matrix containing the principal components of the video sequence. The driving noise process \( v_t \) is normally distributed with zero mean and covariance \( Q \), i.e. \( v_t \sim \mathcal{N}(0, Q) \) where \( Q \in \mathbb{R}^{n \times n} \). The observed noise \( w_t \) is also zero mean and Gaussian, with covariance \( R \), i.e. \( w_t \sim \mathcal{N}(0, R) \) where \( R \in \mathbb{R}^{m \times m} \). We extend the definition of [12] by allowing the initial state \( x_1 \) to have arbitrary mean and covariance, i.e. \( x_1 \sim \mathcal{N}(\mu, S) \). The

\[1\] In [12], the initial condition is specified by \( x_1 \sim \mathcal{N}(Ax_0, Q) \) where \( x_0 \in \mathbb{R}^n \)
A number of methods are available to learn the parameters of the dynamic texture from a training video sequence, including methods that are asymptotically optimal, in the maximum likelihood sense, such as N4SID [19] or expectation-maximization [25], and a suboptimal (but computationally efficient) solution [12].

**B. Mixture of dynamic textures**

Under the dynamic texture mixture model, the observed video sequence \( y_1^\tau \) is sampled from one of \( K \) dynamic textures, each having some non-zero probability of occurrence. This is a useful extension for two classes of applications. The first class involves video which is homogeneous...
at each time instant, but has varying statistics over time. For example, the problem of clustering a set of video sequences taken from a stationary highway traffic camera. While each video will depict traffic moving at homogeneous speed, the exact appearance of each sequence is controlled by the amount of traffic congestion. Different levels of traffic congestion can be represented by \( K \) dynamic textures. The second involves inhomogeneous video, i.e. video composed of multiple process that can be individually modeled as dynamic textures of different parameters. For example, the problem of segmenting video from a forest scene where trees blow in the wind while clouds move in the sky. A random video patch (e.g. \( 5 \times 5 \times \tau \) pixels) extracted from the video will either contain waving trees or moving clouds, and a collection of video patches can be represented as a sample from a mixture of two dynamic textures.

Formally, given mixture component probabilities \( \{ \alpha_1, \ldots, \alpha_K \} \) with \( \sum_{j=1}^{K} \alpha_j = 1 \) and dynamic texture components of parameters \( \{ \Theta_1, \ldots, \Theta_K \} \), a video sequence is drawn by:

1) Sampling a component index \( z \) from the multinomial distribution parameterized by \( \{ \alpha_1, \ldots, \alpha_K \} \).

2) Sampling an observation \( y_1^\tau \) from the dynamic texture component of parameters \( \Theta_z \).

The probability of a sequence \( y_1^\tau \) under this model is

\[
p(y_1^\tau) = \sum_{j=1}^{K} \alpha_j p(y_1^\tau | z = j)
\]

where \( p(y_1^\tau | z = j) \) is the class conditional probability of the \( j^{th} \) dynamic texture, i.e. the dynamic texture component parameterized by \( \Theta_j = \{ A_j, Q_j, C_j, R_j, \mu_j, S_j \} \). The system of equations that define the mixture of dynamic textures is

\[
\begin{align*}
x_{t+1} &= A_z x_t + v_t \\
y_t &= C_z x_t + w_t
\end{align*}
\]

where the random variable \( z \), which signals the mixture component from which the observations are drawn (the mixture component index), the initial conditions, and the noise processes are given by

\[
\begin{align*}
z &\sim \text{multinomial}(\alpha_1, \ldots, \alpha_K), \\
x_1 &\sim \mathcal{N}(\mu_z, S_z), \\
v_t &\sim \mathcal{N}(0, Q_z), \\
w_t &\sim \mathcal{N}(0, R_z).
\end{align*}
\]
The conditional distributions of the states and observations, given the component index $z$, is

$$p(x_1|z) = G(x_1, \mu_z, S_z)$$

(9)

$$p(x_t|x_{t-1}, z) = G(x_t, A_z x_{t-1}, Q_z)$$

(10)

$$p(y_t|x_t, z) = G(y_t, C_z x_t, R_z),$$

(11)

and the overall joint distribution is

$$p(y_1^\tau, x_1^\tau, z) = p(z) p(x_1|z) \prod_{t=2}^\tau p(x_t|x_{t-1}, z) \prod_{t=1}^\tau p(y_t|x_t, z).$$

(12)

The graphical model for the dynamic texture mixture is presented in Figure 1b. Note that, although the addition of the random variable $z$ introduces loops in the graph, exact inference is still tractable because $z$ is connected to all other nodes. Hence, the graph is already moralized and triangulated [22], and the junction tree of Figure 1b is equivalent to that of the basic dynamic texture, with the variable $z$ added to each clique. This implies that the complexity of exact inference for a mixture of $K$ dynamic textures is $K$ times that of the underlying dynamic texture.

III. PARAMETER ESTIMATION USING EM

The EM algorithm [23] is a procedure for estimating the parameters of a probability distribution that depends on hidden variables (i.e. the output of the system is observed, but its state unknown). For the dynamic texture mixture, the observed information is a set of video sequences $\{y^{(i)}\}_{i=1}^N$, and the missing data consists of: 1) the assignment $z^{(i)}$ of each sequence to a mixture component, and 2) the hidden state sequence $x^{(i)}$ that produces $y^{(i)}$. The EM algorithm is an iterative procedure that alternates between estimating the missing information with the current parameters, and computing new parameters given the estimate of the missing information. In particular, each iteration consists of

\textbf{E – Step}: \quad Q(\Theta; \hat{\Theta}) = E_{X,Z|Y:\hat{\Theta}}(\log p(X, Y, Z; \Theta)) \quad (13)

\textbf{M – Step}: \quad \hat{\Theta}^* = \arg\max_{\Theta} Q(\Theta; \hat{\Theta}) \quad (14)

where $p(X, Y, Z; \Theta)$ is the complete data likelihood of the observations, hidden states, and hidden assignment variables, parameterized by $\Theta$. To maximize clarity, we only present here the
equations of the E and M steps for the estimation of dynamic texture mixture parameters. Their detailed derivation is given in Appendix I. The only assumptions are that the observations are independent and zero-mean, but the algorithm could be trivially extended to the case of non-zero means. All equations follow the notation of Table I.

Observations are denoted by \( \{y^{(i)}\}_{i=1}^{N} \), the hidden state variables by \( \{x^{(i)}\}_{i=1}^{N} \), and the hidden assignment variables by \( \{z^{(i)}\}_{i=1}^{N} \). As is usual in the EM literature [23], we introduce a vector \( z_{i} \in \{0, 1\}^{K} \), such that \( z_{i,j} = 1 \) if and only if \( z^{(i)} = j \). The complete data log-likelihood is (up to a constant) given by

\[
\ell(X, Y, Z) = \sum_{i,j} z_{i,j} \log \alpha_{j} \\
- \frac{1}{2} \sum_{i,j} z_{i,j} \sum_{t=1}^{T} \text{tr} \left[ R_{j}^{-1} \left( y_{t}^{(i)} y_{t}^{(i)^{T}} - y_{t}^{(i)} x_{t}^{(i)^{T}} C_{j}^{T} - C_{j} x_{t}^{(i)} y_{t}^{(i)^{T}} + C_{j} P_{t,t}^{(i)} C_{j}^{T} \right) \right] \\
- \frac{1}{2} \sum_{i,j} z_{i,j} \sum_{t=2}^{T} \text{tr} \left[ Q_{j}^{-1} \left( P_{t,t}^{(i)} - P_{t,t-1}^{(i)} A_{j}^{T} - A_{j} P_{t,t-1}^{(i)} A_{j}^{T} + A_{j} P_{t,t-1}^{(i)} A_{j}^{T} \right) \right] \\
- \frac{1}{2} \sum_{i,j} z_{i,j} \text{tr} \left[ S_{j}^{-1} \left( P_{1,1}^{(i)} - x_{1}^{(i)^{T}} \mu_{j}^{T} - \mu_{j} x_{1}^{(i)^{T}} + \mu_{j} \mu_{j}^{T} \right) \right]
\]
where  

\[ P_{t,t}^{(i)} = x_t^{(i)} (x_t^{(i)})^T \quad \text{and} \quad P_{t,t-1}^{(i)} = x_t^{(i)} (x_{t-1}^{(i)})^T. \]

Applying the expectation of (13) to (15) yields the \( Q \) function

\[
Q(\Theta; \hat{\Theta}) = \sum_j \hat{N}_j \log \alpha_j \tag{16}
\]

where

\[
\hat{N}_j = \sum_{i=1}^N \hat{z}_{i,j} \\
\phi_j = \sum_{i=1}^N \hat{z}_{i,j} \sum_{t=1}^T \hat{P}_{t,t|j}^{(i)} \\
\psi_j = \sum_{i=1}^N \hat{z}_{i,j} \sum_{t=2}^T \hat{P}_{t,t-1|j}^{(i)} \\
\Lambda_j = \sum_{i=1}^N \hat{z}_{i,j} \sum_{t=1}^T y_t^{(i)} (y_t^{(i)})^T \\
\eta_j = \sum_{i=1}^N \hat{z}_{i,j} \hat{P}_{1,1|j}^{(i)}
\]

and

\[
\hat{x}_{t|j}^{(i)} = E_{x^{(i)|y^{(i)},z^{(i)}}=j} \left( x_t^{(i)} \right) \tag{18}
\]

\[
\hat{P}_{t,t|j}^{(i)} = E_{x^{(i)|y^{(i)},z^{(i)}}=j} \left( P_{t,t|j}^{(i)} \right) = E_{x^{(i)|y^{(i)},z^{(i)}}=j} \left( x_t^{(i)} (x_t^{(i)})^T \right) \tag{19}
\]

\[
\hat{P}_{t,t-1|j}^{(i)} = E_{x^{(i)|y^{(i)},z^{(i)}}=j} \left( P_{t,t-1|j}^{(i)} \right) = E_{x^{(i)|y^{(i)},z^{(i)}}=j} \left( x_t^{(i)} (x_{t-1}^{(i)})^T \right) \tag{20}
\]

\[
\hat{z}_{i,j} = p(z^{(i)} = j| y^{(i)}) = \frac{\alpha_j p(y^{(i)}|z^{(i)} = j)}{\sum_{k=1}^K \alpha_k p(y^{(i)}|z^{(i)} = k)}. \tag{21}
\]

Hence, the E-step consists of computing the conditional expectations (18)-(21), and can be computed efficiently with the Kalman smoothing filter [25] (see also Appendix II), which estimates the mean and covariance of the state variable \( x^{(i)} \) conditioned on the observation.
$y^{(i)}$ and $z^{(i)} = j$,

\begin{align}
\hat{x}_{t|j}^{(i)} &= E_{x^{(i)}|y^{(i)},z^{(i)}=j}(x_t^{(i)}) \\
\hat{V}_{t|j}^{(i)} &= \text{cov}_{x^{(i)}|y^{(i)},z^{(i)}=j}(x_t^{(i)}, x_t^{(i)}) \\
\hat{V}_{t,t-1|j}^{(i)} &= \text{cov}_{x^{(i)}|y^{(i)},z^{(i)}=j}(x_t^{(i)}, x_{t-1}^{(i)}).
\end{align}

The second-order moments of (19) and (20) are then calculated as $\hat{P}_{t|j}^{(i)} = \hat{V}_{t|j}^{(i)} + \hat{x}_{t|j}^{(i)}(\hat{x}_{t|j}^{(i)})^T$ and $\hat{P}_{t,t-1|j}^{(i)} = \hat{V}_{t,t-1|j}^{(i)} + \hat{x}_{t|j}^{(i)}(\hat{x}_{t-1|j}^{(i)})^T$. Finally, the probabilities $p(y^{(i)}|z^{(i)} = j)$ are computed using the “innovations” form of the log-likelihood (again, see [25] or Appendix II).

In the M-step, the dynamic texture parameters are updated according to (14), resulting in the following update step for each mixture component $j$,

\begin{align}
C_j^* &= \Gamma_j(\Phi_j)^{-1}, & R_j^* &= \frac{1}{\tau N_j} (\Lambda_j - C_j^* \Gamma_j), & \alpha_j^* = \frac{N_j}{N}, \\
A_j^* &= \Psi_j(\phi_j)^{-1}, & Q_j^* &= \frac{1}{(\tau-1)N_j} (\varphi_j - A_j^* \Psi_j^T), \\
\mu_j^* &= \frac{1}{N_j} \xi_j, & S_j^* &= \frac{1}{N_j} \eta_j - \mu_j^*(\mu_j^*)^T.
\end{align}

A summary of EM for the mixture of dynamic textures is presented in Algorithm 1. The E-step relies on the Kalman smoothing filter to compute 1) the expectations of the hidden state variables $x_t$, given the observed sequence $y^{(i)}$ and the assignment $z^{(i)}$ to a mixture component; and 2) the probability of observation $y^{(i)}$ given the assignment $z^{(i)}$. The M-step then computes the maximum likelihood parameter values for each dynamic texture component $j$, by averaging over all sequences $\{y^{(i)}\}_{i=1}^N$, weighted by the posterior probability of assigning $z^{(i)} = j$.

IV. CONNECTIONS TO THE LITERATURE

The mixture of dynamic textures, as well as the proposed EM algorithm, are related to several previous works. For a single component ($K = 1$) and a single observation ($N = 1$), the EM algorithm for the mixture of dynamic textures reduces to the classical EM algorithm for learning a linear dynamical system [25]–[27]. The linear dynamical system (1) is a generalization of the factor analysis model [18], a statistical model which explains an observed vector as a combination of measurements which are driven by independent factors. In the LDS framework, the time index $t$ becomes the index of the independent observations $y_t$. The factors $x_t$ (the hidden state) are independent and distributed as $\mathcal{N}(0, I_n)$ (hence $S = Q = I_n$, $\mu = 0$, and $A = 0$).
Algorithm 1 EM for a Mixture of Dynamic Textures

**Input:** $N$ sequences $\{y^{(i)}\}_{i=1}^{N}$, number of components $K$.

Initialize $\Theta_j = \{\alpha_j, A_j, Q_j, R_j, C_j, \mu_j, S_j\}$ for $j = 1$ to $K$.

repeat

{Expectation Step}

for $i = 1$ to $N$ do

for $j = 1$ to $K$ do

Compute the conditional expectations (18-21) using the Kalman smoothing filter (Appendix II) with $y_i$ and $\Theta_j$.

end for

end for

{Maximization Step}

for $j = 1$ to $K$ do

Compute aggregated expectations using (17).


end for

until convergence

**Output:** $\Theta = \{\Theta_j\}_{j=1}^{K}$

observation $y_t$ is then a function of the factors $x_t$, the factor loading matrix $C$ (which explains how each factor influences the observation vector), and the observation noise $\mathcal{N}(0, R)$ where $R$ is a diagonal matrix. With the appropriate restrictions on the mixture parameters, the EM algorithm for a dynamic texture mixture reduces to the EM algorithm used for learning a mixture of factor analyzers [28]. In particular, this requires setting $S_j = Q_j = I$ and $A_j = 0$ for each factor analysis component, and $\tau = 1$ since there are no temporal dynamics.

The dynamic texture mixture is also related to “switching” linear dynamical models, where the system parameters of the LDS are selected via a separate Markovian switching variable as the time series progresses. Variations of these models include [29], [30] where only the observation matrix $C$ switches, [31]–[33] where the state parameters switch ($A$ and $Q$), and [34], [35] where the observation and state parameters switch ($C$, $R$, $A$, and $Q$). These switching models are not mixtures of linear dynamical systems because they have one state variable that evolves according to the active system parameters at each time step.

In contrast to switching models with a single state variable, the switching state-space model proposed in [36] switches the observed variable between the output of different linear dynamic systems at each time step. Each LDS has its own observation matrix and state variable, which
evolves according to its own system parameters. The difference between the switching state-space model and the mixture of dynamic textures is that the switching state-space model can switch between LDS outputs at each time step, whereas the mixture of dynamic textures selects an LDS only once at time $t = 1$, and never switches from it. Hence, the mixture of dynamic textures is similar to a special case of the switching state-space model, where the initial probabilities of the switching variable are the mixture component probabilities $\alpha_j$, and the Markovian transition matrix of the switching variable is equal to the identity matrix.

This key difference has consequences of significant practical importance. In particular, the ability to switch at each time step, in the switching state-space model, results in a posterior distribution that is a Gaussian mixture with a number of terms that increases exponentially with time [36]. Thus, exact inference on this model is intractable, and EM-style of learning requires approximate methods (e.g. variational approximations). In contrast, because the mixture of dynamic textures selects only one LDS for an observed sequence, the posterior is a mixture with a constant number of Gaussians and exact inference in the dynamic texture mixture model is tractable, allowing an exact EM algorithm as derived above.

Applications of switching linear model are numerous, including tracking of multiple objects with sensor data [29], contour tracking in clutter [30], tracking with multiple motion models [33], recognition of computer mouse gestures [32], recognition of honey bee dances [34], human motion modeling [31], economic growth modeling [35], and respiration modeling of people with sleep apnea [36]. Although we apply the mixture of dynamic textures to video, the model is general and can be used to cluster any type of time-series [37]. When compared to the literature in this field, the mixture of dynamic textures can be categorized as a model-based method for clustering multivariate continuous-valued time-series data. Two alternative methods are available for clustering this type of data, both based on the K-means algorithm with distance (or similarity) measures suitable for time-series. The first [38] measures the distance between two time-series with the KL divergence or the Chernoff measure, which are estimated non-parametrically in the spectral domain. The second [39] measures the similarity between two time-series as a weighted average between: 1) the similarity between the two PCA spaces of the time series (the weighted sum of the cosines between the PCA basis vectors), and 2) the similarity of the means of the time-series (the probability that the true mean of the second time-series is further than the Mahalanobis distance between the two empirical means). Note that the similarity
measure of [39] completely ignores the dynamics of the time-series, using only the distribution of the time-series observations. The well known connection between EM and K-means makes these algorithms somewhat related to the mixture of dynamic textures, but they lack a precise probabilistic interpretation. Finally, the mixture of dynamic textures is related to the ARMA mixture proposed in [40]. The main differences are that the ARMA mixture 1) only models univariate data, and 2) does not utilize a hidden state model. On the other hand, the ARMA mixture supports high-order Markov models, while the mixture of dynamic texture is based on a first-order Markovian assumption.

V. Applications

Like any other probabilistic model, the dynamic texture has a large number of potential application domains, many of which extend well beyond the field of computer vision (e.g. modeling of high-dimensional time series for financial applications, weather forecasting, etc.). In this work, we concentrate on vision applications, where mixture models are frequently used to solve problems such as clustering [41], [42], background modeling [43], image segmentation and layering [6], [44]–[48], or retrieval [48], [49]. The dynamic texture mixture extends this class of methods to problems involving video of particle ensembles subject to stochastic motion, i.e. scenes composed of particles that move stochastically but, as a whole, exhibit coherent patterns. We consider, in particular, the problems of clustering and segmentation.

Video clustering can be a useful tool to uncover high-level patterns of structure in a video stream, e.g. recurring events, events of high and low probability, outlying events, etc. These operations are of great practical interest for some classes of particle-ensemble video, such as those which involve understanding video acquired in crowded environments. In this context, video clustering has application to problems such as surveillance, novelty detection, event summarization, or remote monitoring of various types of environments. It can also be applied to the entries of a video database in order to automatically create a taxonomy of video classes that can then be used for database organization or video retrieval, e.g. for the purpose of law-enforcement training. Under the mixture of dynamic textures representation, a set of video sequences is clustered by first learning the mixture that best fits the entire collection of sequences, and then assigning each sequence to the mixture component with largest posterior probability of having
generated it, i.e. by labeling sequence \( y^{(i)} \) with

\[
\ell_i = \arg \max_j \log p(y^{(i)} | z^{(i)} = j) + \log \alpha_j. 
\]  

(26)

Video segmentation addresses the problem of decomposing a video sequence into a collection of homogeneous regions. While this has long been known to be solvable with mixture models and the EM algorithm [6], [7], [42], [44], [45], the success of the segmentation operation depends on the ability of the mixture model to capture the dimensions along which the video is statistically homogeneous. In the context of particle ensembles, homogeneity rarely holds in terms of low-level image measurements, such as pixel colors or optical flow. Instead, it tends to be a global property of these measurements, e.g. that while the optical flow of a crowd entering a sports arena is compressive, the same optical flow becomes expansive when the crowd leaves that space. With traditional mixture representations, segmenting video that exhibits this type of global patterns of homogeneity requires the design of specific models for concepts such as “expansive” and “compressive” motion. This is usually not easy to do. The ability of the dynamic texture mixture to capture homogeneity at the level of both the video appearance and dynamics makes it a more suitable representation to address such problems. In fact, as illustrated in the next section, this type of segmentation can be achieved by simply dividing the sequence into a set of localized spatio-temporal patches and then clustering these patches with a mixture of dynamic textures.

VI. EXPERIMENTAL EVALUATION

The performance of the mixture of dynamic textures was evaluated with respect to various applications: 1) clustering of time-series data, 2) clustering of highway traffic video, and 3) motion segmentation of both synthetic and real video sequences. In all cases, performance was compared to a representative of the state-of-the-art for these application domains. In all experiments, each dynamic texture component \( \Theta_j \) was initialized by using the suboptimal learning method of [12] on a random sequence from the training set. The component probabilities were initialized to a uniform distribution, \( \alpha_j = 1/K \). Since the EM algorithm can converge to a local minimum, multiple runs were performed with different initialization seeds. The parameters which best fit the training data (in the maximum likelihood sense) were then selected. Finally, the observation noise was assumed to be iid (i.e. \( R = \sigma^2 I_m \)), and the covariance matrices \( Q, S \),
and \( R \) were regularized by enforcing a lower bound on their eigenvalues. Videos of the results from all experiments are available from a companion website, accessible at [50].

A. Time-series clustering

We start by presenting results of a comparison of the dynamic texture mixture with several multivariate time-series clustering algorithms. To enable an evaluation based on clustering ground truth, this comparison was performed on a synthetic time-series dataset, generated as follows. First, the parameters of \( K \) LDSs, with state-space dimension \( n = 2 \) and observation-space dimension \( m = 10 \), were randomly generated according to

\[
\mu \sim \mathcal{U}_n(-5, 5), \quad S \sim \mathcal{W}(I_n, n), \quad C \sim \mathcal{N}_{m,n}(0, 1), \quad \sigma^2 \sim \mathcal{W}(1, 2), \quad R = \sigma^2 I_m, \\
Q \sim \mathcal{W}(I_n, n), \quad A_0 \sim \mathcal{N}_{n,n}(0, 1), \quad \lambda_0 \sim \mathcal{U}_1(0.1, 1), \quad A = \lambda_0 A_0 / \lambda_{\text{max}}(A_0),
\]

(27)

where \( \mathcal{N}_{m,n}(\mu, \sigma^2) \) is a distribution on \( \mathbb{R}^{m \times n} \) matrices with each entry distributed as \( \mathcal{N}(\mu, \sigma^2) \), \( \mathcal{W}(\Sigma, d) \) is a Wishart distribution with covariance \( \Sigma \) and \( d \) degrees of freedom, \( \mathcal{U}_d(a, b) \) is a distribution on \( \mathbb{R}^d \) vectors with each coordinate distributed uniformly between \( a \) and \( b \), and \( \lambda_{\text{max}}(A_0) \) is the magnitude of the largest eigenvalue of \( A_0 \). Note that \( A \) is a random scaling of \( A_0 \) such that the system is stable (i.e. the poles of \( A \) are within the unit circle). A time-series dataset was generated by sampling 20 time-series of length 50 from each of the \( K \) LDSs. Finally, each time-series sample was normalized to have zero temporal mean.

The data was clustered using (DytexMix) the mixture of dynamic textures, and three multivariate-series clustering algorithms from the time-series literature. The latter are based on variations of K-means for various similarity measures: (Singhal) PCA subspace similarity [39]; (KakKL) the KL divergence [38]; and (KakCh) the Chernoff measure [38]. For completeness, the data was also clustered with two baseline methods that represented the time-series as a single feature vector (formed by concatenating all observations). These feature vectors were then clustered using K-means [42] based on two measures: (K-means) the Euclidean distance, and (K-means-c) the cosine similarity. In all cases, a clustering error was computed for each value of \( K = \{2, 3, \ldots, 8\} \), by averaging the errors over 100 random trials of the clustering experiment. We refer to this synthetic experiment setup as “synthetic A”. The clustering algorithms were also tested on two variations of the experiment based on time-series which were more difficult to cluster. In the first (“synthetic B”), the \( K \) random LDSs were forced to share a random observation matrix (C...
Fig. 2. Clustering results on a synthetic problem generated from (a) $K$ random LDS (synthetic A); (b) $K$ random LDS with the same observation matrix (synthetic B); (c) $K$ random LDS with large observation noise (synthetic C). Plots show the cluster error versus number of clusters ($K$) for six time-series clustering methods: (DytexMix) mixture of dynamic textures; (Singhal) [39]; (K-means) K-means on concatenated vectors; (K-means-c) K-means using cosine distance; (KakKL) [38] using KL divergence; and (KakCh) [38] using Chernoff distance.

<table>
<thead>
<tr>
<th></th>
<th>DytexMix</th>
<th>Singhal</th>
<th>KakKL</th>
<th>KakCh</th>
<th>K-means</th>
<th>K-means-c</th>
</tr>
</thead>
<tbody>
<tr>
<td>synthetic A</td>
<td>0.034</td>
<td>0.021</td>
<td>0.140</td>
<td>0.029</td>
<td>0.544</td>
<td>0.284</td>
</tr>
<tr>
<td>synthetic B</td>
<td>0.018</td>
<td>0.247</td>
<td>0.067</td>
<td>0.042</td>
<td>0.531</td>
<td>0.403</td>
</tr>
<tr>
<td>synthetic C</td>
<td>0.020</td>
<td>0.220</td>
<td>0.100</td>
<td>0.097</td>
<td>0.625</td>
<td>0.463</td>
</tr>
</tbody>
</table>

**Table II**

Overall clustering error on the three synthetic experiments for six clustering algorithms.

parameter), therefore forcing all time-series to be defined in similar subspaces. In the second ("synthetic C"), the LDSs were designed with large observation noise, i.e. $\sigma^2 \sim \mathcal{W}(16, 2)$. Note that these variations are typically to be expected in video clustering problems. For example, in applications where the appearance component does not change significantly between clusters (e.g. highway video with varying levels of traffic), all time-series will span similar image subspaces.

Figure 2 presents the results obtained with the six clustering algorithms on the three experiments, and Table II shows the overall error ($E$), computed by averaging over $K$. In synthetic A (Figure 2a), Singhal, KakCh, and DytexMix achieved comparable performance ($E = 0.021$, $E = 0.029$, and $E = 0.034$) with Singhal performing slightly better. On the other hand, it is clear that the two baseline algorithms are not suitable for clustering time-series data, albeit there is significant improvement when using K-means-c ($E = 0.284$) rather than K-means ($E = 0.544$). In synthetic B (Figure 2b), the results were different: the dynamic texture mixture performed best ($E = 0.018$), closely followed by KakCh and KakKL ($E = 0.042$ and $E = 0.067$). On the other hand, Singhal did not perform well ($E = 0.247$). This is due to the fact that, in this experiment, all time-series had similar PCA subspaces, therefore causing substantial difficulties to subspace-similarity clustering. Finally, in synthetic C (Figure 2c), the dynamic texture mixture
repeated the best performance \( E = 0.020 \), followed by KakCh and KakKL \( E = 0.097 \) and \( E = 0.100 \), with Singhal performing the worst \( E = 0.220 \). In this case, the difference between the performance of the mixture of dynamic textures and those of the immediate followers was significant. This can be explained by the robustness of the former to observation noise, a property not shared by the latter due to the fragility of non-parametric estimation of spectral matrices.

In summary, these results show that the mixture of dynamic textures performs similarly to state-of-the-art methods, such as [39] and [38], when clusters are well separated. It is, however, more robust against occurrences that increase the amount of cluster overlap, which proved difficult for the other methods. Such occurrences include 1) time-series defined in similar subspaces, and 2) time-series with significant observation noise, and are common in video clustering applications.

### B. Video clustering

To evaluate its performance in problems of practical significance, the dynamic texture mixture was used to cluster video of vehicle highway traffic. In particular, clustering was performed on 253 video sequences collected by the Washington Department of Transportation (WSDOT) on interstate I-5 in Seattle, Washington [20]. Each video clip is 5 seconds long and the collection spanned about 20 hours over two days. The video sequences were converted to grayscale, normalized to \( 48 \times 48 \) pixels, the mean value was removed from each sequence, and the pixel amplitudes normalized to unit variance.

The mixture of dynamic textures was used to organize this dataset into 5 clusters. Figure 3a shows six typical sequences for each of the five clusters. These examples, and further analysis of the sequences in each cluster, reveal that the clusters are in agreement with classes frequently used in the perceptual categorization of traffic: light traffic (spanning 2 clusters), medium traffic, slow traffic, and stopped traffic (“traffic jam”). Figures 3b, 3c, and 3d show a comparison between the temporal evolution of the cluster index and the average traffic speed. The latter was measured by the WSDOT with an electromagnetic sensor (commonly known as a loop sensor) embedded on the highway asphalt, near the camera. The speed measurements are shown in Figure 3b and the temporal evolution of the cluster index is shown for \( K = 2 \) (Figure 3c) and \( K = 5 \) (Figure 3d). Unfortunately, a precise comparison between the speed measurements and the video is not possible because the data originate from two separate systems, and the video data is not time-stamped with fine enough precision. Nonetheless, it is still possible to
examine the correspondence between the speed data and the video clustering. With two clusters, the algorithm groups the video sequences into fast-moving and slow-moving traffic. When the number of clusters is increased to 5, the fast moving traffic is split into two clusters and the slow-moving traffic into 3 clusters, which correspond to different levels of slowed traffic (i.e. medium, slow, and stopped).

C. Motion segmentation

While the idea of using EM for clustering or motion segmentation is not novel, the mixture of dynamic textures representation enables its application to a class of visual processes that has traditionally been quite challenging for clustering and motion segmentation algorithms. To evaluate the performance of the dynamic texture mixture in this context, the following procedure was adopted. A collection of video patches was extracted from the sequence to segment using a $p \times p$ spatially-sliding window. The temporal extent of this window was set to either the length of the entire sequence or a smaller value, in which case the window had size $p \times p \times q$ and slid both spatially and temporally. These patches were then clustered into $K$ classes. The segmented sequence was produced by a voting scheme, where each pixel in a patch receives a vote for the class of that patch, as given by the clustering. The pixels were then assigned to the class with the most votes. Finally, a $3 \times 3$ maximum vote filter was used to smooth the segmented video regions.

The segmentation produced by the mixture of dynamic textures was compared with two other segmentation algorithms, which are representative of the state-of-the-art for traditional represen-
tations. The first method is based on normalized cuts and the “motion profile” representation, which captures the movement of image patches, as presented in [51], [52]. Each pixel in the video is represented with a “motion profile” [51], which is the likelihood of a pixel moving within a neighborhood around the pixel. The computation of the likelihood is based on the summed-squared differences between image patches. The similarity between two motion profiles, and hence two pixels, is measured using cross-correlation [51]. Finally, the pixels are grouped by applying the normalized cuts algorithm [52] to the similarity matrix. For these experiments, a patch window of size $15 \times 15$ and a motion profile neighborhood of $5 \times 5$ were used. The second segmentation method uses optical flow and the mean-shift algorithm [53]. Each pixel was represented as a feature vector containing the average optical flow over a $5 \times 5$ window centered on the pixel. The feature vectors were clustered using the mean-shift algorithm, and feature vectors assigned to the same mode were considered to be from the same segment.

We start by presenting segmentation results on various synthetic sequences resulting from the composition of different dynamic textures, and then present segmentations of (more challenging) real video sequences. In all experiments the video was grayscale, and segmentation was based on patches of either $5 \times 5$, $7 \times 7$, or $15 \times 15$ pixels, depending on the image size. The patches were normalized to have zero temporal mean, and all patches with average temporal pixel variance of less than 50 were marked as static background. Finally, the dimension of the state space was set to $n = 10$. Details of the experimental setup and video can be found in Table III.

<table>
<thead>
<tr>
<th>Video</th>
<th>video size</th>
<th>video patch size</th>
<th>$N$</th>
<th>$K$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ocean-steam</td>
<td>$220 \times 220 \times 140$</td>
<td>$7 \times 7 \times 140$</td>
<td>1849</td>
<td>2</td>
</tr>
<tr>
<td>ocean-appearance</td>
<td>$161 \times 111 \times 120$</td>
<td>$5 \times 5 \times 120$</td>
<td>4266</td>
<td>2</td>
</tr>
<tr>
<td>ocean-dynamics</td>
<td>$161 \times 111 \times 120$</td>
<td>$5 \times 5 \times 120$</td>
<td>4266</td>
<td>2</td>
</tr>
<tr>
<td>ocean-fire</td>
<td>$161 \times 111 \times 120$</td>
<td>$5 \times 5 \times 10$</td>
<td>46926</td>
<td>2</td>
</tr>
<tr>
<td>water-steam-fire</td>
<td>$170 \times 115 \times 70$</td>
<td>$5 \times 5 \times 70$</td>
<td>4648</td>
<td>3</td>
</tr>
<tr>
<td>highway traffic</td>
<td>$160 \times 120 \times 51$</td>
<td>$5 \times 5 \times 51$</td>
<td>1594</td>
<td>4</td>
</tr>
<tr>
<td>bridge traffic</td>
<td>$160 \times 120 \times 51$</td>
<td>$5 \times 5 \times 51$</td>
<td>1686</td>
<td>5</td>
</tr>
<tr>
<td>waterfall</td>
<td>$320 \times 240 \times 50$</td>
<td>$15 \times 15 \times 50$</td>
<td>1967</td>
<td>4</td>
</tr>
<tr>
<td>fountain</td>
<td>$160 \times 110 \times 75$</td>
<td>$5 \times 5 \times 75$</td>
<td>4134</td>
<td>3</td>
</tr>
<tr>
<td>trees</td>
<td>$170 \times 115 \times 120$</td>
<td>$7 \times 7 \times 120$</td>
<td>4510</td>
<td>3</td>
</tr>
<tr>
<td>pedestrian 1</td>
<td>$238 \times 158 \times 200$</td>
<td>$7 \times 7 \times 20$</td>
<td>39672</td>
<td>4</td>
</tr>
<tr>
<td>pedestrian 2</td>
<td>$238 \times 158 \times 200$</td>
<td>$7 \times 7 \times 20$</td>
<td>39672</td>
<td>4</td>
</tr>
<tr>
<td>flag</td>
<td>$170 \times 115 \times 120$</td>
<td>$7 \times 7 \times 10$</td>
<td>49610</td>
<td>4</td>
</tr>
</tbody>
</table>

TABLE III

Motion Segmentation Experiments: video size, video patch size, number of video patches ($N$), and the number of clusters ($K$).
### Fig. 4. Segmentation of synthetic video: (a) ocean and steam; (b) ocean with two patches that are rotated; (c) ocean with two patches of different dynamics; (d) water-steam-fire. The left column shows a frame from the original videos, while the remaining columns show the segmentation using: (DytexMix) mixture of dynamic textures; (NormCuts) normalized-cuts and motion profile [51]; and (OpFlow) mean-shift algorithm with optical flow.

1) **Segmentation of synthetic video:** We present segmentation results on five synthetic sequences. The first three are test sequences from [13] and contain different textures with stationary borders: (Figure 4a) steam superimposed over an ocean background; (Figure 4b) ocean with two regions rotated by ninety degrees, i.e. the regions have the same dynamics, but different appearance; and (Figure 4c) ocean with two regions of slower dynamics, i.e. the regions have the same appearance component, but different dynamics. A frame from each of the original videos is shown in the left column of Figure 4, and the segmentations produced by the mixture of dynamic textures are correct (see second column of Figure 4). The fourth synthetic sequence, shown in Figure 4d, is a composite of three video textures: water, steam, and fire. The segmentation using the mixture of dynamic textures is shown in the second column, and again the three

<table>
<thead>
<tr>
<th>Original</th>
<th>DytexMix</th>
<th>NormCuts</th>
<th>OpFlow</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
</tbody>
</table>
segmented regions are correct. The segmentation of these four synthetic videos using normalized-cuts (NormCuts) and optical flow (OpFlow) are shown in the third and fourth columns of Figure 4, respectively. Except for the ocean-appearance sequence, neither of the segmentation algorithms performs well on these test sequences.

The final synthetic sequence (also from [13]) consists of fire superimposed on an ocean background. Segmenting the video is challenging because the boundary of the fire region is not stationary, i.e. the region changes over time as the flame evolves. This can be seen in Figure 5a, which shows six frames from the original sequence. The video was segmented by clustering patches generated by a $5 \times 5 \times 10$ window that slides both spatially and temporally. Figure 5b shows the segmentation obtained with the mixture of dynamic textures. Note that this algorithm is able to generate a moving boundary, that tracks the fire as it moves. For comparison, the segmentation obtained with a pure spatially-sliding window (i.e. a window occupying the entire temporal volume) is shown in Figure 5c. In this case, the segmented fire region is the area which was occupied by fire in a majority of the frames. Figures 5d and 5e show the segmentations
produced by normalized cuts and optical flow, respectively. The results using normalized cuts are comparable to those of the mixture of dynamic textures. On the other hand, mean-shift segmentation performs poorly, failing to segment parts of the flame (e.g. frames 2, 3, and 5).

2) Segmentation of real video: We finish with segmentation results on eight real video sequences, which depict highway traffic, natural scenes, pedestrian crowds, and a flag moving in the wind. All of these sequences are more challenging than the synthetic ones considered above, but it is not as straightforward to perform a precise comparison of the different techniques because there is no absolute ground truth. Nevertheless, the segmentations are sufficiently different to support a qualitative ranking of the performance of the various methods.

The segmentation of a highway traffic scene using $K = 4$ clusters is shown in Figure 6a. The mixture of dynamic textures segmented the video into regions of traffic that move away from the camera (the two large regions on the right) and regions that move towards the camera (the regions on the left). The only error is the split of the incoming traffic into two regions, and can be explained by the strong perspective effects inherent to car motion towards the camera. Figure 6b shows another example of segmentation of vehicle motion, this time on a bridge. Traffic lanes of opposite direction are correctly segmented near the camera, but merged further down the bridge. The algorithm also segments the water in the bottom-right of the image, but assigns it to the same cluster as the distant traffic.

While not perfect, these segmentations are significantly better than those produced by the traditional representations. For both normalized cuts and mean shift, segmented regions tend to extend over multiple lanes, incoming and outgoing traffic are merged with greater ease than desirable, and the same lane is frequently broken into a number of sub-regions. Nevertheless, the combination of mean-shift and optical flow outperforms that of normalized cuts and motion profiles. These sequences illustrate some of the properties of video of particle ensembles that make it very challenging to traditional representations: 1) the motion information is quite sparse (there are significant gaps between cars), 2) the perspective effects can be extreme (cars at a distance occupy little more than a single pixel), and 3) some of the segmentation errors could really only be avoided with recourse to high level scene interpretation (e.g. the bridge example where the visual appearance of the distant traffic is indeed more similar to that of the water near the bridge than to that of the nearby traffic). Because it is a global model, the dynamic texture model is much less susceptible to the sparsity problem. It could also be explicitly extended to...
Fig. 6. Segmentation of real video: (a) highway traffic; (b) vehicle traffic on bridge; (c) water fountain; (d) waterfall; (e) trees in the wind. The left column shows a frame from the original videos, while the remaining columns show the segmentation using mixtures of dynamic textures (DytexMix), normalized-cuts (NormCuts), and optical flow with mean-shift (OpFlow).

overcome the other problems, e.g. by explicitly accounting for the drastic perspective deformation to which the dynamic texture components are subject. We intend to consider such extensions in the future.

The next two examples involve video of moving water. Figure 6c shows a segmentation of water in a fountain. The segmentation using DytexMix is close to perfect, and the different regions corresponding to regions of different water dynamics, i.e. water flowing down a wall, water falling, and the turbulent water in the pool. Figure 6d shows the segmentation of a waterfall
scene where the segmentation has a similar interpretation (i.e. fast moving water, turbulent water, and slow moving water). The segmentation of a tree scene is shown in Figure 6e. In this case, the video contains tree branches waving in the foreground, and clouds moving slowly in the background. The segments produced with the dynamic texture mixture correspond to the clouds in the background, moving branches that are aligned diagonally to the right, and moving branches aligned diagonally to the left. For all these sequences, the segmentations produced by the traditional methods are much more difficult to explain than those of the dynamic texture mixture. For example, in Figure 6c, the perceptually very distinct regions of flowing and falling water are, in both cases, merged. Overall, Figure 6 supports the conclusion that, for most real scenes, the segmentations produced by the dynamic texture mixture are vastly superior to those achievable with the conventional representations.

Figure 7. Segmentation of a pedestrian scene: (a) frames from the original video; (b) segmentation of pedestrian motion.

Figure 8. Segmentation of a crowded pedestrian scene: (a) frames from the original video; Segmentation using: (b) dynamic texture mixture (c) normalized cuts and motion profile; (d) optical flow and mean-shift.
Figures 7 and 8 address the segmentation of pedestrian scenes. The first scene, shown in Figure 7a, contains sparse pedestrian traffic, i.e. with large gaps between pedestrians. The scene was segmented with $K = 4$ clusters, and the segmentation is shown in Figure 7b. The dynamic texture mixture segmented people moving up the walkway from people moving down the walkway. The remaining two segments account for patches which are only half-covered by moving objects, i.e. they model the boundary between regions with motion and regions without. The second scene contains a large crowd that is moving up the walkway, while only a few people move in the opposite direction. Again, the dynamic texture mixture segmented the groups moving in different directions, even in instances where only one person is surrounded by the crowd and is moving in the opposite direction. Figures 8c and 8d show the segmentations produced by the two traditional methods. The segmentation produced by normalized cuts contains gross errors, e.g. frame 2 at the far end of the walkway. While the segmentation achieved with mean-shift is more comparable to that of the mixture of dynamic textures, it tends to over-segment the people moving down the walkway.

This example also illustrates one major advantage of the dynamic texture mixture, which we have not yet discussed: that it can be used to efficiently segment long video sequences. This is achieved by first learning the mixture model on a short training sequence (e.g. a clip from the long sequence), and then segmenting the long sequence by classifying its video patches using the learned mixture model. Since classification only requires computing the patch likelihoods under each mixture component, the computation necessary to segment the long sequence is greatly reduced. To illustrate this, a continuous hour of video was segmented using the mixture model learned from the second scene (Figure 8a), and is available at the companion website [50]. Note that segmentation of long sequences with the two alternative methods is not as straightforward or as fast. Segmentation using normalized-cuts requires generalizing the graph partitioning problem to out-of-sample points (e.g. by computing a data-dependent kernel [54]). Segmentation with the mean-shift algorithm is straightforward since the patches from the training sequence define the non-parametric distribution used for clustering. However, unlike the dynamic texture mixture, there is no reduction in computation (relative to the training phase) because all the training data must be used for classification of the patches in the long sequence.

The final video (Figure 9a) contains a flag moving chaotically in the wind, and clouds moving in the background. The segmentation produced by the dynamic texture mixture for $K = 4$ clusters
Fig. 9. Segmentation of flag moving in the wind: (a) frames from the original video; (b) segmentation of flag motion.

is shown in Figure 9b. The first segment corresponds to the moving cloud background, and the second to the oscillation of the flag pole and flag rope in the wind. The third segment models the motion of the flag when the stripes are horizontal (see the last three frames), and the fourth segment models when the flag is moving chaotically, e.g. the trailing edge of the flag, and when the flag is folded over itself. Finally, the “stars” region of the flag is segmented from the stripes, but it is labeled as the moving clouds segment. The stars are fairly stationary, and they are very small and have similar gray-scale intensity to the flag background. Hence, the perceptible motion within the region is more similar to the moving clouds than the other regions of motion.

In summary, the mixture of dynamic textures produces sensible segmentations over a significant diversity of scenes involving particle ensembles subject to stochastic motion. This is in stark contrast to the performance of the two representatives of the state-of-the-art for traditional representations. The latter seem to work well only on a few sequences that are suited for the particular representation (either image patch differences or optical flow), while performing poorly on all others.

VII. CONCLUSIONS

In this work, we have introduced the mixture of dynamic textures, a principled probabilistic extension of the dynamic texture model. Whereas a dynamic texture models a single video sequence as a sample from a linear dynamic system, a mixture of dynamic textures models a collection of sequences as samples from a set of linear dynamic systems. We derived an exact EM algorithm for learning the parameters of the model from a set of training video, and explored the connections between the model and other linear system models, such as factor analysis, mixtures of factor analyzers, and switching linear systems. Through extensive video clustering and segmentation experiments, we have also demonstrated the efficacy of the
mixture of dynamic textures for modeling video, both holistically and locally (patch-based representations). In particular, it has been shown that the mixture of dynamic textures is a suitable model for simultaneously representing the localized motion and appearance of a variety of visual processes (e.g. fire, water, steam, clouds, trees, cars, people), and that the model provides a natural framework for clustering such processes. For the purposes of motion segmentation, the mixture of dynamic textures is capable of segmenting video where traditional representations, such as motion profiles or optical flow, perform poorly. As a final remark, it is worth mentioning that some of the results above, e.g. the segmentation of pedestrian scenes, suggest that the dynamic texture mixture could be the basis for the design of computer vision systems capable of successfully tackling problems, e.g. monitoring and surveillance of crowded environments, which currently have great societal interest.

APPENDIX I

EM ALGORITHM FOR THE MIXTURE OF DYNAMIC TEXTURES

This appendix presents the derivation of the EM algorithm for the mixture of dynamic textures. In particular, the complete-data log-likelihood function, the E-step, and the M-step are derived in the remainder of this appendix.

A. Log-Likelihood Functions

We start by obtaining the log-likelihood of the complete data, i.e. the observations \( \{y^{(i)}\}_{i=1}^{N} \), the hidden state variables \( \{x^{(i)}\}_{i=1}^{N} \), and the hidden assignment variables \( \{z^{(i)}\}_{i=1}^{N} \). As is usual in the EM literature [23], we introduce a vector \( z_i \in \{0, 1\}^K \), such that \( z_{i,j} = 1 \) if and only if \( z^{(i)} = j \). Using (12), the complete data log-likelihood can then be written as

\[
\ell(X, Y, Z) = \sum_{i=1}^{N} \log p(x^{(i)}, y^{(i)}, z^{(i)}) 
\]

\[
= \sum_{i=1}^{N} \log \prod_{j=1}^{K} \left[ p(x^{(i)}, y^{(i)}, z^{(i)} = j) \right]^{\hat{z}_{i,j}}
\]

\[
= \sum_{i,j} \hat{z}_{i,j} \log \left[ \alpha_j p(x_1^{(i)} | z^{(i)} = j) \prod_{t=2}^{\tau} p(x_t^{(i)} | x_{t-1}^{(i)}, z^{(i)} = j) \prod_{t=1}^{\tau} p(y_t^{(i)} | x_t^{(i)}, z^{(i)} = j) \right]
\]

\[
= \sum_{i,j} \hat{z}_{i,j} \left[ \log \alpha_j + \sum_{t=1}^{\tau} \log p(y_t^{(i)} | x_t^{(i)}, z^{(i)} = j) + \log p(x_1^{(i)} | z^{(i)} = j) \right]
\]

September 7, 2006 DRAFT
\[ + \sum_{t=2}^{T} \log p(x_{t}^{(i)}|x_{t-1}^{(i)}, z^{(i)} = j) \] .

Note that, from (9)-(11), the sums of the log-conditional probability terms are of the form

\[ \sum_{i,j} a_{i,j} \sum_{t=t_{0}}^{t_{1}} \log G(b_{t}, c_{j,t}, M_{j}) = \]

\[ -\frac{n}{2}(t_{1} - t_{0} + 1) \log 2\pi \sum_{i,j} a_{i,j} - \frac{1}{2} \sum_{i,j} \sum_{t=t_{0}}^{t_{1}} ||b_{t} - c_{j,t}||_{M_{j}} - \frac{t_{1} - t_{0} + 1}{2} \sum_{i,j} a_{i,j} \log |M_{j}|. \]

Since the first term on the right-hand side of this equation does not depend on the parameters of the dynamic texture mixture, it does not affect the maximization performed in the M-step and can, therefore, be dropped. Substituting the appropriate parameters for \( b_{t}, c_{j,t} \) and \( M_{j} \), we have:

\[ \ell(X, Y, Z) = \sum_{i,j} z_{i,j} \log \alpha_{j} - \frac{1}{2} \sum_{i,j} z_{i,j} \sum_{t=1}^{\tau} y_{t}^{(i)} - C_{j}x_{t}^{(i)} \right|^{2}_{R_{j}} - \frac{\tau}{2} \sum_{i,j} z_{i,j} \log |R_{j}| \]

\[ - \frac{1}{2} \sum_{i,j} z_{i,j} \sum_{t=2}^{\tau} ||x_{t}^{(i)} - A_{j}x_{t-1}^{(i)}||^{2}_{Q_{j}} - \frac{\tau - 1}{2} \sum_{i,j} z_{i,j} \log |Q_{j}| \]

\[ - \frac{1}{2} \sum_{i,j} z_{i,j} ||x_{1}^{(i)} - \mu_{j}||^{2}_{S_{j}} - \frac{1}{2} \sum_{i,j} z_{i,j} \log |S_{j}| \].

Defining the random variables \( P_{t,t}^{(i)} = x_{t}^{(i)}(x_{t}^{(i)})^{T} \) and \( P_{t,t-1}^{(i)} = x_{t}^{(i)}(x_{t-1}^{(i)})^{T} \) and expanding the Mahalanobis distance terms, the log-likelihood becomes (15).

**B. E-Step**

The expectation step of the EM algorithm is to take the expectation of (15) conditioned on the observed data and the current parameter estimates \( \hat{\Theta} \), as in (13). We note that each term of \( \ell(X, Y, Z) \) is of the form \( z_{i,j}f(x^{(i)}, y^{(i)}) \), for some functions \( f \) of \( x^{(i)} \) and \( y^{(i)} \). Taking the expectation we have

\[ E_{X,Y,Z}[z_{i,j}f(x^{(i)}, y^{(i)})] = E_{Z|Y}[E_{X|Y,Z}[z_{i,j}f(x^{(i)}, y^{(i)})]] \]

\[ = E_{z^{(i)}|y^{(i)}}[E_{x^{(i)}|y^{(i), z^{(i)}}}[z_{i,j}f(x^{(i)}, y^{(i)})]] \]

\[ = p(z_{i,j} = 1|y^{(i)})E_{x^{(i)}|y^{(i), z^{(i)} = j}}[f(x^{(i)}, y^{(i)})] \]

where (33) follows from the assumption that the observations are independent. For the first term of (34), \( p(z_{i,j} = 1|y^{(i)}) \) is the posterior probability of \( z^{(i)} = j \) given the observation \( y^{(i)} \), and is
analogous to that of the hidden state of the standard Gaussian mixture model [23]

\[ \hat{z}_{i,j} = p(z_{i,j} = 1|y^{(i)}) \]  
\[ = p(z^{(i)} = j|y^{(i)}) \]  
\[ = \frac{\alpha_j p(y^{(i)}|z^{(i)} = j)}{\sum_{k=1}^{K} \alpha_k p(y^{(i)}|z^{(i)} = k)}. \]

The functions \( f(x^{(i)}, y^{(i)}) \) are at most quadratic in \( x_t^{(i)} \). Hence for the second term of (34), the expectations of the functions \( f(x^{(i)}, y^{(i)}) \) only depend on the first and second moments of the states of the jth mixture component conditioned on the observed data \( y_t^{(i)} \), (18-20). The expectations are obtained using the Kalman smoothing filter [25] (also see Appendix II) as described in Section III.

The \( Q \) function is obtained by replacing the random variables \( z_{i,j}, (z_{i,j}x_t^{(i)}), (z_{i,j}P_t^{(i)}) \) and \( (z_{i,j}P_{t-1}^{(i)}) \) in the complete data log-likelihood (15) with the corresponding expectations \( \hat{z}_{i,j}, (\hat{z}_{i,j}\hat{x}_{t|j}^{(i)}), (\hat{z}_{i,j}\hat{P}_{t|j}^{(i)}), \) and \( (\hat{z}_{i,j}\hat{P}_{t-1|j}^{(i)}) \),

\[ Q(\Theta; \hat{\Theta}) = \sum_{i,j} \hat{z}_{i,j} \log \alpha_j \]  
\[ - \frac{1}{2} \sum_{i,j} \hat{z}_{i,j} \sum_{t=1}^{T} \text{tr} \left[ R_j^{-1} \left( y_t^{(i)}y_t^{(i)^T} - y_t^{(i)}(\hat{x}_{t|j}^{(i)})^T C_j^T - C_j \hat{x}_{t|j}^{(i)}y_t^{(i)^T} + C_j \hat{P}_{t|j}^{(i)} C_j^T \right) \right] \]  
\[ - \frac{1}{2} \sum_{i,j} \hat{z}_{i,j} \sum_{t=2}^{T} \text{tr} \left[ Q_j^{-1} \left( \hat{P}_{t|j}^{(i)} - \hat{P}_{t-1|j}^{(i)} A_j - A_j(\hat{P}_{t-1|j}^{(i)})^T + A_j \hat{P}_{t-1|j}^{(i)} A_j^T \right) \right] \]  
\[ - \frac{1}{2} \sum_{i,j} \hat{z}_{i,j} \text{tr} \left[ S_j^{-1} \left( \hat{P}_{1,1|j}^{(i)} - \hat{x}_{1|j}^{(i)} \mu_j - \mu_j(\hat{x}_{1|j}^{(i)})^T + \hat{N}_j \mu_j \mu_j^T \right) \right] \]  
\[ - \frac{\tau}{2} \sum_{i,j} \hat{z}_{i,j} \log |R_j| - \frac{\tau - 1}{2} \sum_{i,j} \hat{z}_{i,j} \log |Q_j| - \frac{1}{2} \sum_{i,j} \hat{z}_{i,j} \log |S_j| . \]

Finally, defining the aggregated expectations (17), the \( Q \) function can be rewritten as (16).

C. M-Step

In the M-step of the EM algorithm (14), the reparameterization of the model is obtained by maximizing the \( Q \) function by taking the partial derivative with respect to each parameter and setting it to zero. The maximization problem with respect to each parameter appears in two
common forms. The first is a maximization with respect to a square matrix $X$

$$X^* = \arg\!\max_X -\frac{1}{2} \text{tr} (X^{-1}A) - \frac{b}{2} \log |X|.$$  \hspace{1cm} (39)

Maximizing by taking the derivative and setting to zero yields the following solution

$$\frac{\partial}{\partial X} -\frac{1}{2} \text{tr} (X^{-1}A) - \frac{b}{2} \log |X| = \frac{1}{2} X^{-T} A^T X^{-T} - \frac{b}{2} X^{-T} = 0$$  \hspace{1cm} (40)

$$= A^T - b X^T = 0$$  \hspace{1cm} (41)

$$\Rightarrow X^* = \frac{1}{b} A.$$  \hspace{1cm} (42)

The second form is a maximization problem with respect to a matrix $X$ of the form

$$X^* = \arg\!\max_X -\frac{1}{2} \text{tr} \left[ D(-BX^T - XB^T + X C X^T) \right]$$  \hspace{1cm} (43)

where $D$ and $C$ are symmetric and invertible matrices. Again, maximizing by taking the derivative and setting to zero yields the solution

$$\frac{\partial}{\partial X} -\frac{1}{2} \text{tr} \left[ D(-BX^T - XB^T + X C X^T) \right] = -\frac{1}{2} (-DB - D^T B + D^T X C^T + DXC) = 0$$  \hspace{1cm} (44)

$$= DB - DXC = 0$$  \hspace{1cm} (45)

$$\Rightarrow X^* = BC^{-1}$$  \hspace{1cm} (46)

where (46) and (47) follow from the fact that $D$ and $C$ are invertible symmetric covariance matrices. The optimal parameters can be found by collecting the relevant terms in the $Q$ function and maximizing.

1) Observation Matrix:

$$C^*_j = \arg\!\max_{C_j} -\frac{1}{2} \text{tr} \left[ R_j^{-1} (-\Gamma_j C_j^T - C_j \Gamma_j^T + C_j \Phi_j C_j^T) \right]$$  \hspace{1cm} (48)

This is of the form in (43), hence the solution is given by $C^*_j = \Gamma_j (\Phi_j)^{-1}$.

2) Observation Noise Covariance:

$$R^*_j = \arg\!\max_{R_j} -\frac{1}{2} \text{tr} \left[ R_j^{-1} (\Lambda_j - \Gamma_j C_j^T - C_j \Gamma_j^T + C_j \Phi_j C_j^T) \right] - \frac{\tau \tilde{N}_j}{2} \log |R_j|$$  \hspace{1cm} (49)
This is of the form in (39), hence the solution is

\[
R_j^* = \frac{1}{\tau N_j} (\Lambda_j - \Gamma_j C_j^T - C_j \Gamma_j^T + C_j \Phi_j C_j^T) \tag{50}
\]

\[
= \frac{1}{\tau N_j} (\Lambda_j - \Gamma_j \Phi_j^{-1} \Gamma_j^T - \Gamma_j \Phi_j^{-1} \Gamma_j^T + \Gamma_j \Phi_j^{-1} \Phi_j \Phi_j^{-1} \Gamma_j^T) \tag{51}
\]

\[
= \frac{1}{\tau \hat{N}_j} (\Lambda_j - \Gamma_j \Phi_j^{-1} \Gamma_j^T). \tag{52}
\]

Substituting the optimal value \( C_j^* \) into (51) yields \( R_j^* = \frac{1}{\tau \hat{N}_j} (\Lambda_j - C_j^* \Gamma_j^T) \).

3) State Transition Matrix:

\[
A_j^* = \arg\max_{A_j} -\frac{1}{2} \text{tr} \left[ Q_j^{-1} \left( -\Psi_j A_j^T - A_j \Psi_j^T + A_j \phi_j A_j^T \right) \right] \tag{53}
\]

This is of the form in (43), hence \( A_j^* = \Psi_j (\phi_j)^{-1} \).

4) State Noise Covariance:

\[
Q_j^* = \arg\max_{Q_j} -\frac{1}{2} \text{tr} \left[ Q_j^{-1} \left( \varphi_j - \Psi_j A_j^T - A_j \Psi_j^T + A_j \phi_j A_j^T \right) \right] - \frac{(\tau - 1) \hat{N}_j}{2} \log |Q_j| \tag{54}
\]

This is of the form in (39), hence the solution can be computed as

\[
Q_j^* = \frac{1}{(\tau - 1) \hat{N}_j} \left( \varphi_j - \Psi_j A_j^T - A_j \Psi_j^T + A_j \phi_j A_j^T \right) \tag{55}
\]

\[
= \frac{1}{(\tau - 1) \hat{N}_j} \left( \varphi_j - \Psi_j \phi_j^{-1} \Psi_j^T - \Psi_j \phi_j^{-1} A_j \Psi_j^T + \Psi_j \phi_j^{-1} \phi_j \phi_j^{-1} \Psi_j^T \right) \tag{56}
\]

\[
= \frac{1}{(\tau - 1) \hat{N}_j} \left( \varphi_j - \Psi_j \phi_j^{-1} \Psi_j^T \right) \tag{57}
\]

Substituting the optimal value \( A_j^* \) into (56) yields \( Q_j^* = \frac{1}{(\tau - 1) \hat{N}_j} \left( \varphi_j - A_j^* \Psi_j^T \right) \).

5) Initial State Mean:

\[
\mu_j^* = \arg\max_{\mu_j} -\frac{1}{2} \text{tr} \left[ S_j^{-1} \left( -\xi_j \mu_j^T - \mu_j \xi_j^T + \hat{N}_j \mu_j \mu_j^T \right) \right] \tag{58}
\]

This is of the form in (43), hence the solution is given by \( \mu_j^* = \frac{1}{\hat{N}_j} \xi_j \).

6) Initial State Covariance:

\[
S_j^* = \arg\max_{S_j} -\frac{1}{2} \text{tr} \left[ S_j^{-1} \left( \eta_j - \xi_j \mu_j^T - \mu_j \xi_j^T + \hat{N}_j \mu_j \mu_j^T \right) \right] - \frac{\hat{N}_j}{2} \log |S_j| \tag{59}
\]
This is of the form in (39), hence the solution is given by

\[ S^*_j = \frac{1}{\hat{N}_j} \left( \eta_j - \xi_j\mu_j^T - \mu_j\xi_j^T + \hat{N}_j\mu_j\mu_j^T \right) \]  

(60)

\[ = \frac{1}{\hat{N}_j} \left( \eta_j - \frac{1}{\hat{N}_j}\xi_j\xi_j^T - \frac{1}{\hat{N}_j}\xi_j\xi_j^T + \frac{1}{\hat{N}_j}\xi_j\xi_j^T \right). \]  

(61)

Substituting the optimal value \( \mu_j^* \) into (61) yields

\[ S^*_j = \frac{1}{\hat{N}_j}\eta_j - \mu_j^*(\mu_j^*)^T. \]  

7) Class Probabilities: A Lagrangian multiplier is used to enforce that \( \{\alpha_j\} \) sum to 1,

\[ \alpha = \arg\max_{\alpha,\lambda} \sum_j \hat{N}_j \log \alpha_j + \lambda \left( \sum_j \alpha_j - 1 \right) \]  

(62)

where \( \alpha = \{\alpha_1, \ldots, \alpha_K\} \). Taking the derivatives of (62) and setting to zero,

\[ \frac{\partial}{\partial \lambda} = \sum_j \alpha_j - 1 = 0 \Rightarrow \sum_j \alpha_j = 1 \]  

(63)

and

\[ \frac{\partial}{\partial \alpha_j} = \frac{\hat{N}_j}{\alpha_j} + \lambda = 0 \Rightarrow \alpha_j\lambda = -\hat{N}_j \]  

(64)

\[ \lambda \sum_j \alpha_j = -\sum_j \hat{N}_j \]  

(65)

\[ \lambda = -N \]  

(66)

where (66) follows from condition (63) and \( \sum_j \hat{N}_j = N \). Substituting for \( \lambda \) in (65), and solving for \( \alpha_j \) yields \( \alpha_j^* = \frac{\hat{N}_j}{N} \).

APPENDIX II

KALMAN SMOOTHING FILTER

The Kalman smoothing filter [18], [25] estimates the mean and covariance of the state \( x_t \) of a linear dynamical system, parameterized by \( \Theta = \{A, Q, C, R, \mu, S\} \), when conditioned on the entire observed sequence \( \{y_1, \ldots, y_T\} \). The Kalman filter can also be used to efficiently compute the log-likelihood of the observed sequence, and is used in the E-step of the EM algorithm for learning a linear dynamical system from a single observed sequence [25]. Define the expectations
conditioned on the observed sequence from time $t = 1$ to $t = s$,

\[
\hat{x}_t^s = E_{x|y_1, \ldots, y_s}(x_t) \quad (67)
\]
\[
\hat{V}_t^s = E_{x|y_1, \ldots, y_s}((x_t - \hat{x}_t^s)(x_t - \hat{x}_t^s)^T) \quad (68)
\]
\[
\hat{V}_{t,t-1}^s = E_{x|y_1, \ldots, y_s}((x_t - \hat{x}_t^s)(x_{t-1} - \hat{x}_{t-1}^s)^T) \quad (69)
\]

then the mean and covariances conditioned on the entire observed sequence are $\hat{x}_t^T$, $\hat{V}_t^T$, and $\hat{V}_{t,t-1}^T$. The estimates are calculated using a set of recursive equations: For $t = 1, \ldots, \tau$

\[
\hat{x}_{t-1}^T = A\hat{x}_{t-1}^{t-1} \quad (70)
\]
\[
\hat{V}_{t-1}^T = A\hat{V}_{t-1}^{t-1}A^T + Q \quad (71)
\]
\[
K_t = \hat{V}_{t-1}^T C^T (C\hat{V}_{t-1}^{t-1}C^T + R)^{-1} \quad (72)
\]
\[
\hat{x}_t^T = \hat{x}_{t-1}^T + K_t(y_t - C\hat{x}_{t-1}^{t-1}) \quad (73)
\]
\[
\hat{V}_t^T = \hat{V}_{t-1}^T - K_tC\hat{V}_{t-1}^{t-1} \quad (74)
\]

where the initial conditions are $\hat{x}_1^0 = \mu$ and $\hat{V}_1^0 = S$. The estimates $\hat{x}_t^T$ and $\hat{V}_t^T$ are obtained with the backward recursions. For $t = \tau, \ldots, 1$

\[
J_{t-1} = \hat{V}_{t-1}^{t-1} A^T (\hat{V}_{t-1}^{t-1})^{-1} \quad (75)
\]
\[
\hat{x}_{t-1}^T = \hat{x}_{t-1}^{t-1} + J_{t-1}(\hat{x}_t^T - A\hat{x}_{t-1}^{t-1}) \quad (76)
\]
\[
\hat{V}_{t-1}^T = \hat{V}_{t-1}^{t-1} + J_{t-1}(\hat{V}_t^T - \hat{V}_{t-1}^{t-1})J_{t-1}^T \quad (77)
\]

The covariance $\hat{V}_{t,t-1}^T$ is computed recursively, for $t = \tau, \ldots, 2$

\[
\hat{V}_{t-1,t-2}^T = \hat{V}_{t-1}^{t-1} J_{t-2}^T + J_{t-1}(\hat{V}_{t,t-1}^T - A\hat{V}_{t-1}^{t-1})J_{t-2}^T \quad (78)
\]

with initial condition $\hat{V}_{\tau,\tau-1}^T = (I - K_\tau C)A\hat{V}_{\tau-1}^{t-1}$. Finally, the data log-likelihood can also be computed efficiently using the “innovations” form [25]

\[
\log p(y_1^\tau) = \sum_{t=1}^\tau \log p(y_t|y_1^{t-1}) \quad (79)
\]
\[
= -\frac{1}{2} \sum_{t=1}^\tau \log \left| C\hat{V}_{t}^{t-1}C^T + R \right| - \frac{1}{2} \sum_{t=1}^\tau \|y_t - C\hat{x}_{t}^{t-1}\|^2_{(C\hat{V}_{t}^{t-1}C^T + R)} \quad (80)
\]
If \( R \) is an i.i.d. or diagonal covariance matrix (i.e. \( R = r I_m \)), then the filter can be computed efficiently using the matrix inversion lemma.

**ACKNOWLEDGMENT**

The authors would like to thank Gianfranco Doretto and Stefano Soatto for providing the synthetic test sequences used in [15], the Washington State Department of Transportation [20] for providing the videos of highway traffic, and Daniel J. Dailey for providing the loop-sensor data. This work was funded by NSF award IIS-0534985.

**REFERENCES**


