This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

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 studies 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 the model, and the model is related to previous works in linear systems, machine learning, time-series clustering, control theory, 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 that have traditionally been challenging for computer vision (e.g. fire, steam, water, vehicle and pedestrian traffic, etc.). When compared with state-of-the-art methods in motion segmentation, including both temporal texture methods and traditional representations (e.g. optical flow or other localized motion representations), the mixture of dynamic textures achieves superior performance in the problems of clustering and segmenting video of such processes.**

***Index Terms*— Dynamic texture, temporal textures, video modeling, video clustering, motion segmentation, mixture models, linear dynamical systems, time-series clustering, Kalman filter, probabilistic models, expectation-maximization.**

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 *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 beehive. 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]–[19]. In fact, the dynamic texture framework has been shown to have great potential for video synthesis [13], [19], image registration [14], motion segmentation [15], [18]–[22], and video classification [16], [17]. 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 significant limitation of the original dynamic texture model [13] is, however, its inability to provide a perceptual decomposition into multiple regions, each of which belongs to a semantically different visual process: 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. One possibility to address this problem is to apply the dynamic texture model locally [15], by splitting the video into a collection of localized spatio-temporal patches, fitting the dynamic texture to each patch, and clustering the resulting models. However, this method, along with other recent proposals [18], [22], lacks some of the attractive properties of the original dynamic texture model: a clear interpretation as a probabilistic generative model for video, and the necessary robustness to operate without manual initialization.

To address these limitations, we note that while the holistic dynamic texture model of [13] is not suitable for such scenes, the underlying generative framework is. 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 volume is then modeled as a simple dynamic texture. This leads to a natural extension of the dynamic texture model, a *mixture of dynamic textures* [20] (or mixture of linear dynamical systems), that we study in this work. The mixture of dynamic textures is a generative model, where a collection of video sequences (or video patches) are modeled as samples from a set of underlying dynamic textures.

Digital Object Indentifier 10.1109/TPAMI.2007.70738 0162-8828/$25.00 © 2007 IEEE

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

It provides a natural probabilistic framework for clustering video, and for video segmentation through the clustering of spatio-temporal patches.

In addition to introducing the dynamic texture mixture as a generative model for video, we report on three main contributions. First, an expectation-maximization (EM) algorithm is derived for maximum-likelihood estimation of the parameters of a dynamic texture mixture. Second, the relationships between the mixture model and various other models previously proposed, including mixtures of factor analyzers, linear dynamical systems, and switched linear dynamic models, are analyzed. Finally, we demonstrate the applicability of the 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 describe 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], [13] 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

$$\left\{ \begin{array}{rl} x_{t+1} = Ax_t + v_t \\ y_t = Cx_t + w_t \end{array} \right. \quad (1)$$

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* (e.g. containing the *principal components* of the video sequence when learned with [13]). 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 $Q \in \mathbb{S}_+^n$ is a positive-definite $n \times n$ matrix. The *observation noise* w_t is also zero mean and Gaussian, with covariance R, i.e. $w_t \sim \mathcal{N}(0, R,)$ where $R \in \mathbb{S}_+^m$. Note that the model adopted throughout this work, which supports an initial state x_1 of arbitrary mean and covariance, i.e. $x_1 \sim \mathcal{N}(\mu, S)$, is a slight extension of the model originally proposed in [13][1]. This extension produces a richer video model that can capture variability in the initial frame, and is necessary for learning a dynamic texture from multiple video samples with different initial frames (as is the case in clustering and segmentation problems). The dynamic texture is specified by the parameters $\Theta = \{A, Q, C, R, \mu, S\}$, and can be represented by the graphical model of Figure 1a.

[1]The initial condition in [13] is specified by a fixed initial vector $x_0 \in \mathbb{R}^n$, or equivalently $x_1 \sim \mathcal{N}(Ax_0, Q)$.

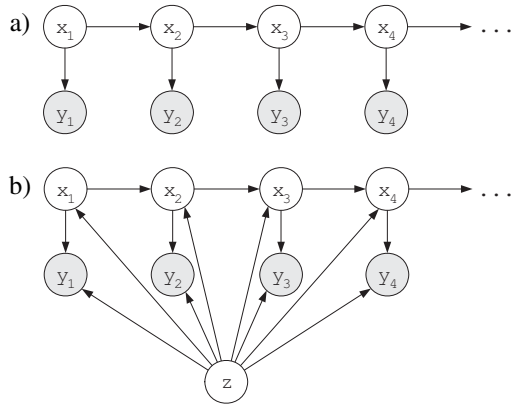

Fig. 1. a) Dynamic texture; b) Dynamic texture mixture. The hidden variable z selects the parameters of the remaining nodes.

It can be shown [23], [24] from this definition that the distributions of the initial state, the conditional state, and the conditional observation are

$$p(x_1) = G(x_1, \mu, S) \quad (2)$$

$$p(x_t|x_{t-1}) = G(x_t, Ax_{t-1}, Q) \quad (3)$$

$$p(y_t|x_t) = G(y_t, Cx_t, R) \quad (4)$$

where $G(x, \mu, \Sigma) = (2\pi)^{-n/2}|\Sigma|^{-1/2}e^{-\frac{1}{2}\|x-\mu\|_\Sigma^2}$ is the n-dimensional multivariate Gaussian distribution, and $\|x\|_\Sigma^2 = x^T\Sigma^{-1}x$ is the Mahalanobis distance with respect to the covariance matrix Σ. Letting $x_1^\tau = (x_1, \cdots, x_\tau)$ and $y_1^\tau = (y_1, \cdots, y_\tau)$ be a sequence of states and observations, the joint distribution is

$$p(x_1^\tau, y_1^\tau) = p(x_1)\prod_{t=2}^{\tau} p(x_t|x_{t-1})\prod_{t=1}^{\tau} p(y_t|x_t). \quad (5)$$

A number of methods are available to learn the parameters of the dynamic texture from a training video sequence, including maximum-likelihood methods (e.g. expectation-maximization [25]), non-iterative subspace methods (e.g. N4SID [26]) or a suboptimal, but computationally efficient, procedure [13].

B. Mixture of dynamic textures

Under the dynamic texture mixture model, the observed video sequence y_1^τ 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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

different parameters. For example, in a video scene containing fire and smoke, a random video patch taken from the video will contain either fire or smoke, and a collection of video patches can be represented as a sample from a mixture of two dynamic textures.

Formally, given component priors $\alpha = \{\alpha_1, \ldots, \alpha_K\}$ 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 α.
2) Sampling an observation y_1^τ from the dynamic texture component of parameters Θ_z.

The probability of a sequence y_1^τ under this model is

$$p(y_1^\tau) = \sum_{j=1}^{K} \alpha_j p(y_1^\tau | z = j) \tag{6}$$

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

$$\begin{cases} x_{t+1} = A_z x_t + v_t \\ y_t = C_z x_t + w_t \end{cases} \tag{7}$$

where the random variable $z \sim \text{multinomial}(\alpha_1, \cdots, \alpha_K)$ signals the mixture component from which the observations are drawn, the initial condition is given by $x_1 \sim \mathcal{N}(\mu_z, S_z)$, and the noise processes by $v_t \sim \mathcal{N}(0, Q_z)$ and $w_t \sim \mathcal{N}(0, R_z)$. The conditional distributions of the states and observations, given the component index z, are

$$p(x_1|z) = G(x_1, \mu_z, S_z) \tag{8}$$

$$p(x_t|x_{t-1}, z) = G(x_t, A_z x_{t-1}, Q_z) \tag{9}$$

$$p(y_t|x_t, z) = G(y_t, C_z x_t, R_z), \tag{10}$$

and the overall joint distribution is

$$p(y_1^\tau, x_1^\tau, z) = p(z)p(x_1|z) \prod_{t=2}^{\tau} p(x_t|x_{t-1}, z) \prod_{t=1}^{\tau} p(y_t|x_t, z). \tag{11}$$

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 [27], 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 makes the complexity of exact inference for a mixture of K dynamic textures K times that of the underlying dynamic texture.

III. Parameter estimation using EM

Given a set of i.i.d. video sequences $\{y^{(i)}\}_{i=1}^N$, we would like to learn the parameters Θ of a mixture of dynamic textures that best fits the data in the maximum-likelihood sense [23], i.e.

$$\Theta^* = \underset{\Theta}{\text{argmax}} \sum_{i=1}^{N} \log p(y^{(i)}; \Theta). \tag{12}$$

When the probability distribution depends on hidden variables (i.e. the output of the system is observed, but its state is unknown), the maximum-likelihood solution can be found with the EM algorithm [28]. 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

$$\text{E} - \text{Step}: \ \mathcal{Q}(\Theta; \hat{\Theta}) = \text{E}_{X,Z|Y;\hat{\Theta}}(\log p(X, Y, Z; \Theta)) \tag{13}$$

$$\text{M} - \text{Step}: \ \hat{\Theta}^* = \underset{\Theta}{\text{argmax}}\, \mathcal{Q}(\Theta; \hat{\Theta}) \tag{14}$$

where $p(X, Y, Z; \Theta)$ is the complete-data likelihood of the observations, hidden states, and hidden assignment variables, parameterized by Θ. 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 drawn independently and have zero-mean, but the algorithm could be trivially extended to the case of non-zero means. All equations follow the notation of Table I.

N	number of observed sequences.
τ	length of an observed sequence.
K	number of mixture components.
i	index over the set of observed sequences.
j	index over the components of the mixture.
t	time index of a sequence.
$y^{(i)}$	the i^{th} observed sequence.
$y_t^{(i)}$	the observation at time t of $y^{(i)}$.
$x^{(i)}$	the state sequence corresponding to $y^{(i)}$.
$x_t^{(i)}$	the state at time t of $x^{(i)}$.
$z^{(i)}$	the mixture component index for the i^{th} sequence.
$\mathbf{z}_{i,j}$	binary variable which indicates ($\mathbf{z}_{i,j}$=1) that $y^{(i)}$ is drawn from component j.
α_j	the probability of the j^{th} component, $p(z = j)$.
Θ_j	the parameters of the j^{th} component.
X	$\{x^{(i)}\}$ for all i.
Y	$\{y^{(i)}\}$ for all i.
Z	$\{z^{(i)}\}$ for all i.

TABLE I
NOTATION FOR EM FOR MIXTURE OF DYNAMIC TEXTURES

A. EM algorithm for mixture of dynamic textures

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 [28], we introduce a vector $\mathbf{z}_i \in \{0, 1\}^K$, such that $\mathbf{z}_{i,j} = 1$ if and only if $z^{(i)} = j$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

$$\ell(X,Y,Z) = \sum_{i,j} \mathbf{z}_{i,j} \log \alpha_j - \frac{1}{2}\sum_{i,j} \mathbf{z}_{i,j} \log |S_j| - \frac{1}{2}\sum_{i,j} \mathbf{z}_{i,j} \mathrm{tr}\left[S_j^{-1}\left(P_{1,1}^{(i)} - x_1^{(i)}\mu_j^T - \mu_j {x_1^{(i)}}^T + \mu_j\mu_j^T\right)\right] \tag{15}$$
$$- \frac{\tau}{2}\sum_{i,j} \mathbf{z}_{i,j} \log |R_j| - \frac{1}{2}\sum_{i,j} \mathbf{z}_{i,j} \sum_{t=1}^{\tau} \mathrm{tr}\left[R_j^{-1}\left(y_t^{(i)} {y_t^{(i)}}^T - y_t^{(i)} {x_t^{(i)}}^T C_j^T - C_j x_t^{(i)} {y_t^{(i)}}^T + C_j P_{t,t}^{(i)} C_j^T\right)\right]$$
$$- \frac{\tau-1}{2}\sum_{i,j} \mathbf{z}_{i,j} \log |Q_j| - \frac{1}{2}\sum_{i,j} \mathbf{z}_{i,j} \sum_{t=2}^{\tau} \mathrm{tr}\left[Q_j^{-1}\left(P_{t,t}^{(i)} - P_{t,t-1}^{(i)} A_j^T - A_j {P_{t,t-1}^{(i)}}^T + A_j P_{t-1,t-1}^{(i)} A_j^T\right)\right]$$

$$\mathcal{Q}(\Theta;\hat{\Theta}) = -\frac{1}{2}\sum_j \mathrm{tr}\left[R_j^{-1}\left(\Lambda_j - \Gamma_j C_j^T - C_j \Gamma_j^T + C_j \Phi_j C_j^T\right)\right] - \frac{1}{2}\sum_j \mathrm{tr}\left[S_j^{-1}\left(\eta_j - \xi_j\mu_j^T - \mu_j\xi_j^T + \hat{N}_j \mu_j \mu_j^T\right)\right] \tag{16}$$
$$- \frac{1}{2}\sum_j \mathrm{tr}\left[Q_j^{-1}\left(\varphi_j - \Psi_j A_j^T - A_j \Psi_j^T + A_j \phi_j A_j^T\right)\right] + \sum_j \hat{N}_j \left[\log \alpha_j - \frac{\tau}{2}\log|R_j| - \frac{\tau-1}{2}\log|Q_j| - \frac{1}{2}\log|S_j|\right]$$

The complete-data log-likelihood is (up to a constant) given by (15) where $P_{t,t}^{(i)} = x_t^{(i)}(x_t^{(i)})^T$ and $P_{t,t-1}^{(i)} = x_t^{(i)}(x_{t-1}^{(i)})^T$. Applying the expectation of (13) to (15) yields the $\mathcal{Q}$ function (16) where

$$\begin{array}{ll} \hat{N}_j = \sum_i \hat{\mathbf{z}}_{i,j} & \Phi_j = \sum_i \hat{\mathbf{z}}_{i,j} \sum_{t=1}^{\tau} \hat{P}_{t,t|j}^{(i)} \\ \xi_j = \sum_i \hat{\mathbf{z}}_{i,j} \hat{x}_{1|j}^{(i)} & \varphi_j = \sum_i \hat{\mathbf{z}}_{i,j} \sum_{t=2}^{\tau} \hat{P}_{t,t|j}^{(i)} \\ \eta_j = \sum_i \hat{\mathbf{z}}_{i,j} \hat{P}_{1,1|j}^{(i)} & \phi_j = \sum_i \hat{\mathbf{z}}_{i,j} \sum_{t=2}^{\tau} \hat{P}_{t-1,t-1|j}^{(i)} \\ \Psi_j = \sum_i \hat{\mathbf{z}}_{i,j} \sum_{t=2}^{\tau} \hat{P}_{t,t-1|j}^{(i)} & \\ \Lambda_j = \sum_i \hat{\mathbf{z}}_{i,j} \sum_{t=1}^{\tau} y_t^{(i)} (y_t^{(i)})^T & \\ \Gamma_j = \sum_i \hat{\mathbf{z}}_{i,j} \sum_{t=1}^{\tau} y_t^{(i)} (\hat{x}_{t|j}^{(i)})^T & \end{array} \tag{17}$$

are the aggregates of the expectations

$$\hat{x}_{t|j}^{(i)} = \mathrm{E}_{x^{(i)}|y^{(i)},z^{(i)}=j}\left(x_t^{(i)}\right) \tag{18}$$
$$\hat{P}_{t,t|j}^{(i)} = \mathrm{E}_{x^{(i)}|y^{(i)},z^{(i)}=j}\left(P_{t,t}^{(i)}\right) \tag{19}$$
$$\hat{P}_{t,t-1|j}^{(i)} = \mathrm{E}_{x^{(i)}|y^{(i)},z^{(i)}=j}\left(P_{t,t-1}^{(i)}\right) \tag{20}$$

and the posterior assignment probability is

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

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

$$\hat{x}_{t|j}^{(i)} = \mathrm{E}_{x^{(i)}|y^{(i)},z^{(i)}=j}(x_t^{(i)}) \tag{22}$$
$$\hat{V}_{t,t|j}^{(i)} = \mathrm{cov}_{x^{(i)}|y^{(i)},z^{(i)}=j}(x_t^{(i)}, x_t^{(i)}) \tag{23}$$
$$\hat{V}_{t,t-1|j}^{(i)} = \mathrm{cov}_{x^{(i)}|y^{(i)},z^{(i)}=j}(x_t^{(i)}, x_{t-1}^{(i)}). \tag{24}$$

The second-order moments of (19) and (20) are then calculated as $\hat{P}_{t,t|j}^{(i)} = \hat{V}_{t,t|j}^{(i)} + \hat{x}_{t|j}^{(i)}(\hat{x}_{t|j}^{(i)})^T$ 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 data likelihood $p(y^{(i)}|z^{(i)}=j)$ is 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,

$$\begin{array}{ll} C_j^* = \Gamma_j(\Phi_j)^{-1}, & R_j^* = \frac{1}{\tau\hat{N}_j}\left(\Lambda_j - C_j^*\Gamma_j\right), \\ A_j^* = \Psi_j(\phi_j)^{-1}, & Q_j^* = \frac{1}{(\tau-1)\hat{N}_j}\left(\varphi_j - A_j^*\Psi_j^T\right), \\ \mu_j^* = \frac{1}{\hat{N}_j}\xi_j, & S_j^* = \frac{1}{\hat{N}_j}\eta_j - \mu_j^*(\mu_j^*)^T, \\ \alpha_j^* = \frac{\hat{N}_j}{N}. & \end{array} \tag{25}$$

Algorithm 1 EM for a Mixture of Dynamic Textures

Input: N sequences $\{y^{(i)}\}_{i=1}^N$, number of components K.
Initialize $\{\Theta_j, \alpha_j\}$ for $j = 1$ to K.
repeat
  {Expectation Step}
  for $i = \{1,\ldots,N\}$ and $j = \{1,\ldots,K\}$ **do**
    Compute the expectations (18-21) with the Kalman smoothing filter (Appendix II) on y_i and Θ_j.
  end for
  {Maximization Step}
  for $j = 1$ to K **do**
    Compute aggregate expectations (17).
    Compute new parameters $\{\Theta_j, \alpha_j\}$ with (25).
  end for
until convergence
Output: $\{\Theta_j, \alpha_j\}_{j=1}^K$

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 component assignment $z^{(i)}$; and 2) the likelihood 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 $z^{(i)} = j$.

B. Initialization Strategies

It is known that the accuracy of parameter estimates produced by EM is dependent on how the algorithm is initialized. In the remainder of this section, we present three initialization

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

strategies that we have empirically found to be effective for learning mixtures of dynamic textures.

1) Initial seeding: If an initial clustering of the data is available (e.g. by specification of initial contours for segmented regions), then each mixture component is learned by application of the method of [13] to each of the initial clusters.

2) Random trials: Several trials of EM are run with different random initializations, and the parameters which best fit the data, in the maximum-likelihood sense, are selected. For each EM trial, each mixture component is initialized by application of the method of [13] to a randomly selected example from the dataset.

3) Component splitting: The EM algorithm is run with an increasing number of mixture components, a common strategy in the speech-recognition community [29]:

1) Run the EM algorithm with $K = 1$ mixture components.
2) Duplicate a mixture component, and perturb the new component's parameters.
3) Run the EM algorithm, using the new mixture model as initialization.
4) Repeat Steps 2 and 3 until the desired number of components is reached.

For the mixture of dynamic textures, we use the following selection and perturbation rules: 1) the mixture component with the largest misfit in the state-space (i.e. the mixture component with the largest eigenvalue of Q) is selected for duplication; and 2) the principal component whose coefficients have the largest initial variance (i.e. the column of C associated with the largest variance in S) is scaled by 1.01.

IV. Connections to the literature

Although novel as a tool for modeling video, with application to problems such as clustering and segmentation, the mixture of dynamic textures and the proposed EM algorithm are related to various previous works in adaptive filtering, statistics and machine learning, time-series clustering, and video segmentation.

A. Adaptive filtering and control

In the control-theory literature, [30] proposes an adaptive filter based on banks of Kalman filters running in parallel, where each Kalman filter models a mode of a physical process. The output of the adaptive filter is the average of the outputs of the individual Kalman filters, weighted by the posterior probability that the observation was drawn from the filter. This is an inference procedure for the hidden state of a mixture of dynamic textures conditioned on the observation. The key difference with respect to this work is that, while [30] focuses on *inference* on the mixture model with known parameters, this work focuses on *learning* the parameters of the mixture model. The work of [30] is the forefather of other multiple-model based methods in adaptive estimation and control [31]–[33], but none address the learning problem.

B. Models proposed in statistics and machine learning

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 an LDS [25], [34], [35]. The LDS is a generalization of the factor analysis model [24], a statistical model which explains an observed vector as a combination of measurements which are driven by independent factors. Similarly the mixture of dynamic textures is a generalization of the mixture of factor analyzers[2], and the EM algorithm for a dynamic texture mixture reduces to the EM algorithm for learning a mixture of factor analyzers [36].

The dynamic texture mixture is also related to "switching" linear dynamical models, where the parameters of a LDS are selected via a separate Markovian switching variable as time progresses. Variations of these models include [37], [38] where only the observation matrix C switches, [39]–[41] where the state parameters switch (A and Q), and [42], [43] where the observation and state parameters switch (C, R, A, and Q). These models have one state variable that evolves according to the active system parameters at each time step. This makes the switching model a mixture of an exponentially increasing number of LDSs with time-varying parameters.

In contrast to switching models with a single state variable, the switching state-space model proposed in [44] switches the observed variable between the output of different LDSs 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 α_j, and the Markovian transition matrix of the switching variable is equal to the identity matrix. The ability of the switching state-space model to switch at each time step results in a posterior distribution that is a Gaussian mixture with a number of terms that increases exponentially with time [44].

While the mixture of dynamic textures is closely related to both switching LDS models and the model of [44], the fact that it selects only one LDS per observed sequence makes the posterior distribution a mixture of a *constant* number of Gaussians. This key difference has consequences of significant practical importance. First, when the number of components increases exponentially (as is the case for models that involve switching), exact inference becomes intractable, and the EM-style of learning requires approximate methods (e.g. variational approximations) which are, by definition, sub-optimal. Second, because the exponential increase in the number of degrees of freedom is not accompanied by an exponential

[2]Restricting the LDS parameters $S = Q = I_n$, $\mu = 0$, $A = 0$, and R as a diagonal matrix yields the factor analysis model. Similarly for the mixture of dynamic textures, setting $S_j = Q_j = I$ and $A_j = 0$ for each factor analysis component, and $\tau = 1$ (since there are no temporal dynamics) yields the mixture of factor analyzers.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

increase in the amount of available data (which only grows linearly with time), the difficulty of the learning problem increases with sequence length. None of these problems affect the dynamic texture mixture, for which exact inference is tractable, allowing the derivation of the exact EM algorithm presented above.

C. Time-series clustering

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. When compared to the literature in this field [45], 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: 1) [46] measures the distance between two time-series with the KL divergence or the Chernoff measure, which are estimated non-parametrically in the spectral domain; and 2) [47] measures similarity by comparing the PCA subspaces and the means of the time-series, but disregards the dynamics of the time-series. The well known connection between EM and K-means makes these algorithms somewhat related to the dynamic texture mixture, but they lack a precise probabilistic interpretation. Finally, the dynamic texture mixture is related to the ARMA mixture proposed in [48]. 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 dynamic texture mixture is based on a first-order Markovian assumption.

D. Video segmentation

The idea of applying dynamic texture representations to the segmentation of video has previously appeared in the video literature. In fact, some of the inspiration for our work was the promise shown for temporal texture segmentation (e.g. smoke and fire) by the dynamic texture model of [13]. For example, [15] segments video by clustering patches of dynamic texture using the level-sets framework and the Martin distance. More recently, [22] clusters pixel-intensities (or local texture features) using auto-regressive processes (AR) and level-sets, and [18] segments video by clustering pixels with similar trajectories in time, using generalized PCA (GPCA). While these methods have shown promise, they do not exploit the probabilistic nature of the dynamic texture representation for the segmentation itself. On the other hand, the segmentation algorithms proposed in the following section are statistical procedures that leverage on the mixture of dynamic texture to perform optimal inference. This results in greater robustness to variations due to the stochasticity of the video appearance and dynamics, leading to superior segmentation results, as will be demonstrated in Sections V and VI.

Finally, it is worth mentioning that we have previously proposed a graphical model which represents video as a collection of layers, where each layer is modeled as a dynamic texture [19], and a Markov random field (MRF) prior is included to guarantee spatial coherence of the segmentation. When compared to the dynamic texture mixture, this model has significantly larger computational requirements. We have not, so far, been able to apply it in the context of experiments of the scale discussed in the following sections.

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 [49], [50], background modeling [51], image segmentation and layering [6], [52]–[56], or retrieval [56], [57]. The dynamic texture mixture extends this class of methods to problems involving video of particle ensembles subject to stochastic motion. We consider, in particular, the problems of clustering and segmentation.

A. Video Clustering

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. 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

$$\ell_i = \operatorname*{argmax}_j \log p(y^{(i)}|z^{(i)} = j) + \log \alpha_j. \tag{26}$$

B. Motion Segmentation

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], [50], [52], [53], 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. For spatio-temporal textures (e.g. video of smoke and fire), traditional mixture-based motion models are not capable of capturing these dimensions, due to their inability to account for the stochastic nature of the underlying motion. The mixture of dynamic textures extends the application of mixture-based

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

segmentation algorithms to video composed of spatio-temporal textures. As in most previous mixture-based approaches to video segmentation, the process consists of two steps. The mixture model is first learned, and the video is then segmented by assigning video locations to mixture components.

In the learning stage, the video is first represented as a bag of patches. For spatio-temporal texture segmentation, a patch of dimensions $p \times p \times q$ is extracted from each location in the video sequence (or along a regularly spaced grid)[3] where p and q should be large enough to capture the distinguishing characteristics of the various components of the local motion field. Note that this is unlike methods that model the changes of appearance of single pixels (e.g. [18] and the AR model of [22]) and, therefore, have no ability to capture the spatial coherence of the local motion field. If the segmentation boundaries are not expected to change over time, q can be set to the length of the video sequence. The set of spatio-temporal patches is then clustered with recourse to the EM algorithm of Section III. The second step, segmentation, scans the video locations sequentially. At each location, a patch is extracted, and assigned to one of the mixture components, according to (26). The location is then declared to belong to the segmentation region associated with that cluster.

It is interesting to note that the mixture of dynamic textures can be used to efficiently segment very long video sequences, 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 with the learned mixture. The segmentation step only requires computing the patch log-likelihoods under each mixture component, i.e. (73). Since the conditional covariances $\hat{V}_t^{t-1}$ and $\hat{V}_t^t$, and the Kalman filter gains K_t do not depend on the observation y_t, they can be pre-computed. The computational steps required for the data-likelihood of a single patch therefore reduce to computing, $\forall t = \{1, \ldots, \tau\}$,

$$\hat{x}_t^{t-1} = A\hat{x}_{t-1}^{t-1}, \tag{27}$$

$$\hat{x}_t^t = \hat{x}_t^{t-1} + K_t(y_t - C\hat{x}_t^{t-1}), \tag{28}$$

$$p(y_t|y_1^{t-1}) = -\frac{1}{2}\log|M| - \frac{m}{2}\log(2\pi) - \frac{1}{2}(y_t - C\hat{x}_t^{t-1})^T M^{-1}(y_t - C\hat{x}_t^{t-1}). \tag{29}$$

where $M = C\hat{V}_t^{t-1}C^T + R$. Hence, computing the data-likelihood under one mixture component of a single patch requires $O(5\tau)$ matrix-vector multiplications. For a mixture of K dynamic textures and N patches, the computation is $O(5\tau KN)$ matrix-vector multiplications.

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. The initialization strategies from Section III-B were used. The observation noise was assumed to be independent and identically distributed (i.e. $R = \sigma^2 I_m$), the initial state covariance S was assumed to be diagonal, 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 [58].

[3]Although overlapping patches violate the independence assumption, they work well in practice and are commonly adopted in computer vision.

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

$$\begin{array}{lll} \mu \sim \mathcal{U}_n(-5,5), & S \sim \mathcal{W}(I_n, n), & C \sim \mathcal{N}_{m,n}(0,1), \\ Q \sim \mathcal{W}(I_n, n), & \lambda_0 \sim \mathcal{U}_1(0.1,1), & A_0 \sim \mathcal{N}_{n,n}(0,1), \\ \sigma^2 \sim \mathcal{W}(1,2), & R = \sigma^2 I_m, & A = \lambda_0 A_0 / \lambda_{max}(A_0). \end{array}$$

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 Σ 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 $\lambda_{max}(A_0)$ 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 the mixture of dynamic textures (DytexMix), and three multivariate-series clustering algorithms from the time-series literature. The latter are based on variations of K-means for various similarity measures: PCA subspace similarity (Singhal) [47]; the KL divergence (KakKL) [46]; and the Chernoff measure (KakCh) [46]. As a baseline, the data was also clustered with K-means [50] using the Euclidean distance (K-means) and the cosine similarity (K-means-c) on feature vectors formed by concatenating each time-series. The correctness of a clustering is measured quantitatively using the Rand index [59] between the clustering and the ground-truth. Intuitively, the Rand index corresponds to the probability of pair-wise agreement between the clustering and the ground-truth, i.e. the probability that the assignment of any two items will be correct with respect to each other (in the same cluster, or in different clusters). For each algorithm, the average Rand index was computed for each value of $K = \{2, 3, \ldots, 8\}$, by averaging over 100 random trials of the clustering experiment. We refer to this synthetic experiment setup as "SyntheticA". The clustering algorithms were also tested on two variations of the experiment based on time-series that were more difficult to cluster. In the first (SyntheticB), the K random LDSs were forced to share the same observation matrix (C), therefore forcing all time-series to be defined in similar subspaces. In the second (SyntheticC), the LDSs had large observation noise, i.e. $\sigma^2 \sim \mathcal{W}(16, 2)$. Note that these

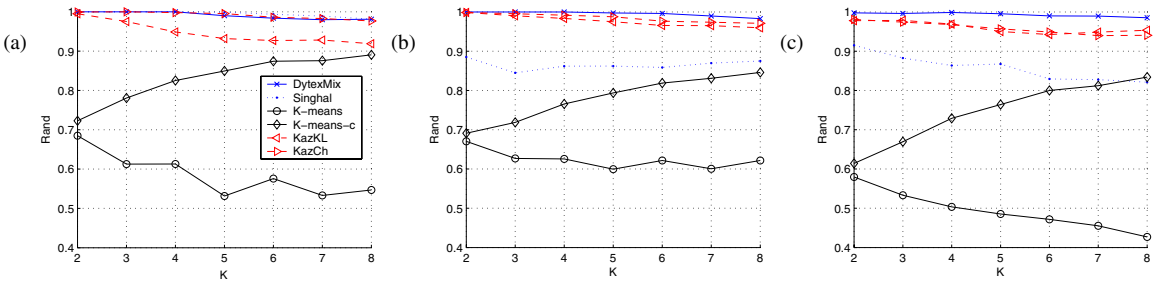

Fig. 2. Time-series clustering results on three synthetic problems: (a) SyntheticA; (b) SyntheticB; and (c) SyntheticC. Plots show the Rand index versus the number of clusters (K) for six time-series clustering algorithms.

	SyntheticA	SyntheticB	SyntheticC
DytexMix	0.991	**0.995**	**0.993**
Singhal [47]	**0.995**	0.865	0.858
KakKL [46]	0.946	0.977	0.960
KakCh [46]	0.991	0.985	0.958
K-means	0.585	0.624	0.494
K-means-c	0.831	0.781	0.746

TABLE II
OVERALL RAND INDEX ON THE THREE SYNTHETIC EXPERIMENTS FOR SIX CLUSTERING ALGORITHMS.

variations are typically 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 the video 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 Rand index, computed by averaging over K. In SyntheticA (Figure 2a), Singhal, KakCh, and DytexMix achieved comparable performance (overall Rand of 0.995, 0.991, and 0.991) 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 (0.831) rather than K-means (0.585). In SyntheticB (Figure 2b), the results were different: the DytexMix performed best (0.995), closely followed by KakCh and KakKL (0.985 and 0.977). On the other hand, Singhal did not perform well (0.865) because all the time-series have similar PCA subspaces. Finally, in SyntheticC (Figure 2c), the DytexMix repeated the best performance (0.993), followed by KakKL and KakCh (0.960 and 0.958), with Singhal performing the worst again (0.858). In this case, the difference between the performance of DytexMix and those of KakKL and KakCh 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 [47] and [46], 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. Clustering was performed on 253 video sequences collected by the Washington Department of Transportation (WSDOT) on interstate I-5 in Seattle, Washington [60]. Each video clip is 5 seconds long and the collection spanned about 20 hours over two days. The video sequences were converted to grayscale, and normalized to have size 48×48 pixels, zero mean, and 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 in 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. For $K = 2$, the algorithm forms two clusters that correspond to fast-moving and slow-moving traffic. Similarly, for $K = 5$, the algorithm creates two clusters for fast-moving traffic, and three clusters for slow-moving traffic (which correspond to medium, slow, and stopped traffic).

C. Motion segmentation

Several experiments were conducted to evaluate the usefulness of dynamic texture mixtures for video segmentation. In all cases, two initialization methods were considered: the manual specification of a rough initial segmentation contour (referred to as DytexMixIC), and the component splitting strategy of Section III-B (DytexMixCS). The segmentation results

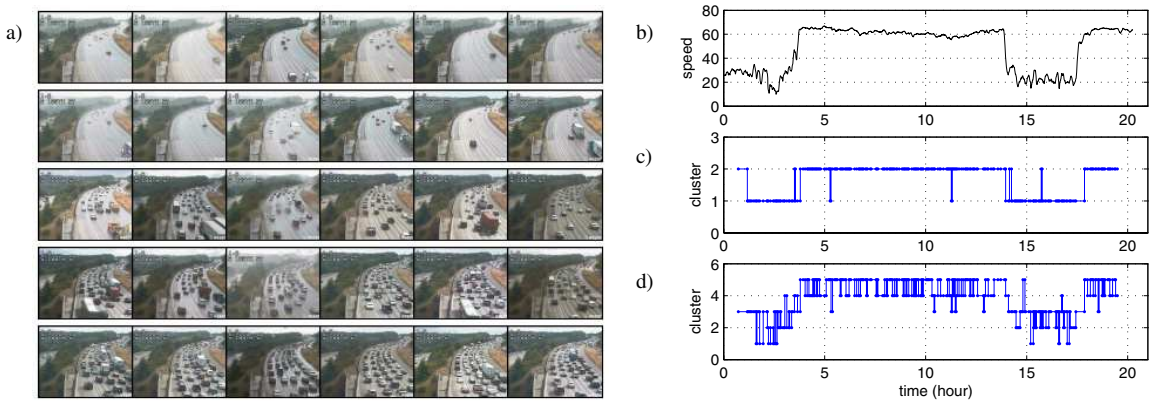

Fig. 3. Clustering of traffic video: (a) six typical sequences from the five clusters; (b) speed measurements from the highway loop sensor over time; and the clustering index over time for (c) 2 clusters, and (d) 5 clusters.

video	size	patch size	K
ocean-steam	$220\times220\times140$	$7\times7\times140$	2
ocean-appearance	$160\times110\times120$	$5\times5\times120$	2
ocean-dynamics	$160\times110\times120$	$5\times5\times120$	2
ocean-fire	$160\times110\times120$	$5\times5\times5$	2
synthdb	$160\times110\times60$	$5\times5\times60$	$2,3,4$
highway traffic	$160\times112\times51$	$5\times5\times51$	4
bridge traffic	$160\times113\times51$	$5\times5\times51$	4
fountains	$160\times110\times75$	$5\times5\times75$	3
pedestrian 1	$238\times158\times200$	$7\times7\times20$	2
pedestrian 2	$238\times158\times200$	$7\times7\times20$	2

TABLE III
VIDEOS IN MOTION SEGMENTATION EXPERIMENT. K IS THE NUMBER OF CLUSTERS.

are compared with those produced by several segmentation procedures previously proposed in the literature: the level-sets method of [22] using Ising models (Ising); generalized PCA (GPCA) [18], and two algorithms representative of the state-of-the-art for traditional optical-flow based motion segmentation. The first method (NormCuts) is based on normalized cuts [61] and the "motion profile" representation proposed in [61], [62][4]. The second (OpFlow) represents each pixel as a feature-vector containing the average optical-flow over a 5×5 window, and clusters the feature-vectors using the mean-shift algorithm [63].

Preliminary evaluation revealed various limitations of the different techniques. For example, the optical flow methods cannot deal with the stochasticity of microscopic textures (e.g. water and smoke), performing much better for textures composed of non-microscopic primitives (e.g. video of crowded scenes composed of objects, such as cars or pedestrians, moving at a distance). On the other hand, level-sets and GPCA perform best for microscopic textures. Due to these limitation, and for the sake of brevity, we limit the comparisons that follow to the methods that performed best for each type of video under consideration. In some cases, the experiments were also restricted by implementation constraints. For example, the available implementations of the level-sets method can only be applied to video composed of two textures.

In all experiments the video are grayscale, and the video patches are either 5×5 or 7×7 pixels, depending on the image size (see Table III for more details). The patches are normalized to have zero temporal mean, and unit variance. Unless otherwise specified, the dimension of the state-space is $n=10$. Finally, *all* segmentations are post-processed with a 5×5 "majority" smoothing filter.

1) Segmentation of synthetic video: We start with the four synthetic sequences studied in [15]: 1) steam over an ocean background (ocean-steam); 2) ocean with two regions rotated by ninety degrees, i.e. regions of identical dynamics but different appearance (ocean-appearance); 3) ocean with two regions of identical appearance but different dynamics (ocean-dynamics); and 4) fire superimposed on an ocean background (ocean-fire). The segmentations of the first three videos are shown in Figure 4. Qualitatively, the segmentations produced by DytexMixIC and Ising ($n=2$) are similar, with Ising performing slightly better with respect to the localization of the segmentation borders. While both these methods improve on the segmentations of [15], GPCA can only segment ocean-steam, failing on the other two sequences. We next consider the segmentation of a sequence containing ocean and fire (Figure 5a). This sequence is more challenging because the boundary of the fire region is not stationary (i.e. the region changes over time as the flame evolves). Once again, DytexMixIC ($n=2$) and Ising ($n=2$) produce comparable segmentations (Figure 5b and 5c), and are capable of tracking the flame over time. We were not able to produce any meaningful segmentation with GPCA on this sequence.

In summary, both DytexMixIC and Ising perform qualitatively well on the sequences from [15], but GPCA does not. The next section will compare these algorithms quantitatively, using a much larger database of synthetic video.

2) Segmentation of synthetic video database: The segmentation algorithms were evaluated quantitatively on a database of 300 synthetic sequences. These consist of three groups of one hundred videos, with each group generated from a common ground-truth template, for $K=\{2,3,4\}$. The segments were randomly selected from a set of 12 textures, which included grass, sea, boiling water, moving escalator,

[4]For this representation, we used a patch of size 15×15 and a motion profile neighborhood of 5×5.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

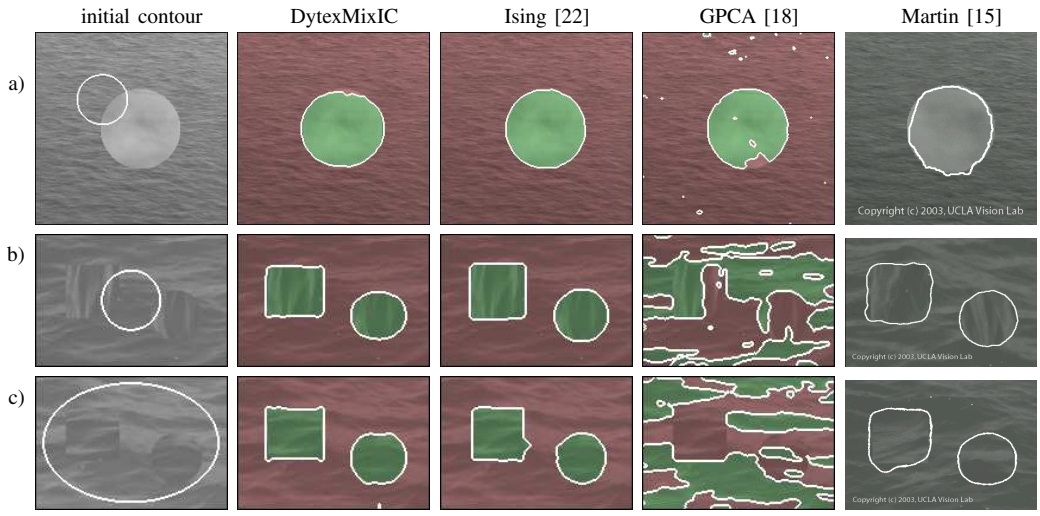

Fig. 4. Segmentation of synthetic video from [15]: (a) ocean-steam; (b) ocean-appearance; and (c) ocean-dynamics. The first column shows a video frame and the initial contour, and the remaining columns show the segmentations from DytexMixIC, Ising [22], GPCA [18], and Martin distance (from [15]).

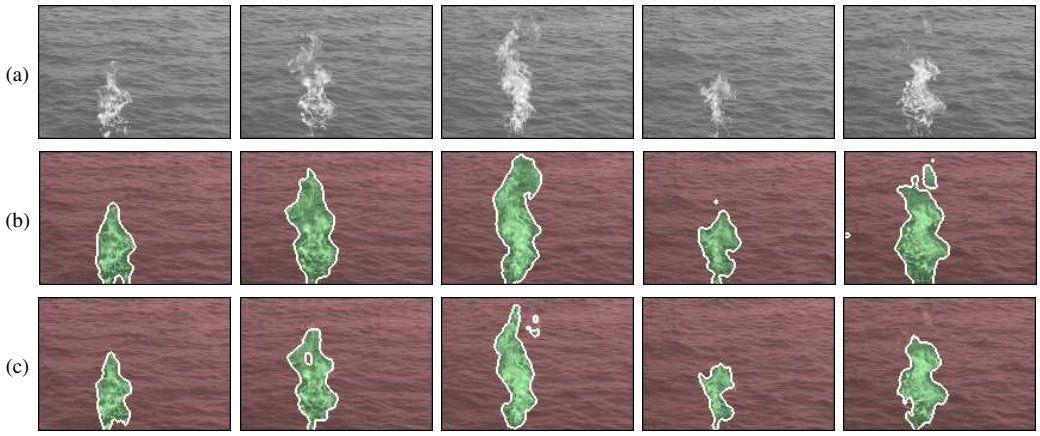

Fig. 5. Segmentation of ocean-fire [15]: (a) video frames; and the segmentation using (b) DytexMixIC; and (c) Ising [22].

fire, steam, water, and plants. Examples from the database can be seen in Figure 7a and 8a, along with the initial contours provided to the segmentation algorithms.

All sequences were segmented with DytexMixIC, DytexMixCS, and GPCA [18]. Due to implementation limitations, the algorithms of [22], Ising, AR, and AR0 (AR without mean) were applied only for $K = 2$. All methods were run with different orders of the motion models (n), while the remaining parameters for each algorithm were fixed throughout the experiment. Performance was evaluated with the Rand index [59] between segmentation and ground-truth. In addition, two baseline segmentations were included: 1) "Baseline Random", which randomly assigns pixels; and 2) "Baseline Init", which is the initial segmentation (i.e. initial contour) provided to the algorithms. The database and segmentation results are available from [58].

Figure 6a shows the average Rand index resulting from the segmentation algorithms for different orders n, and Table IV presents the best results for each algorithm. For all K, DytexMixIC achieved the best overall performance, i.e. largest average Rand index. For $K = 2$, Ising also performs well (0.879), but is inferior to DytexMixIC (0.916). The remaining algorithms performed poorly, with GPCA performing close to random pixel assignment.

While the average Rand index quantifies the overall performance on the database, it does not provide insight on the characteristics of the segmentation failures, i.e. whether there are many small errors, or a few gross ones. To overcome this limitation, we have also examined the segmentation precision of all algorithms. Given a threshold θ, segmentation precision is defined as the percentage of segmentations deemed to be correct with respect to the threshold, i.e. the percentage with Rand index larger than θ. Figure 6b plots the precision of the segmentation algorithms for different threshold levels. One interesting observation can be made when $K = 2$. In this case, for a Rand threshold of 0.95 (corresponding to tolerance of about 1 pixel error around the border), the precisions of DytexMixIC and Ising are, respectively, 73% and 51%. On the other hand, for very high thresholds (e.g 0.98), Ising has a larger precision (31% versus 14% of DytexMixIC). This suggests that Ising is very good at finding the exact boundary when nearly perfect segmentations are possible, but is more prone to dramatic segmentation failures. On the other hand, DytexMixIC is more robust, but not as precise near borders.

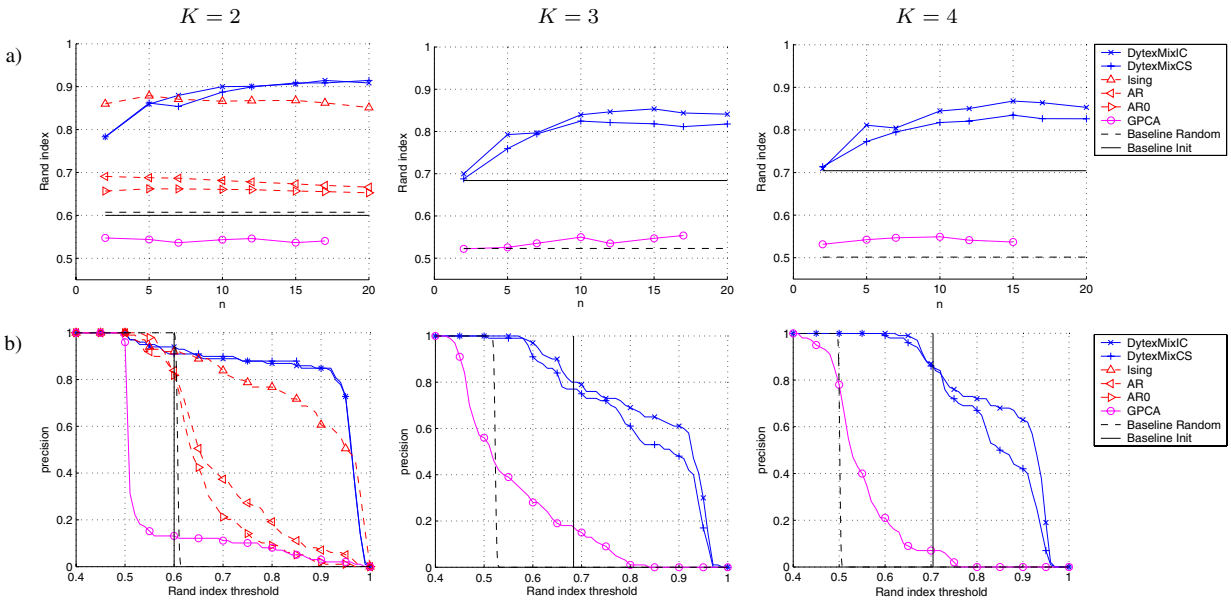

Fig. 6. Results on the synthetic database: a) average Rand index versus the order of the motion model (n); and b) segmentation precision for the best n for each algorithm. Each column presents the results for 2, 3, or 4 segments in the database.

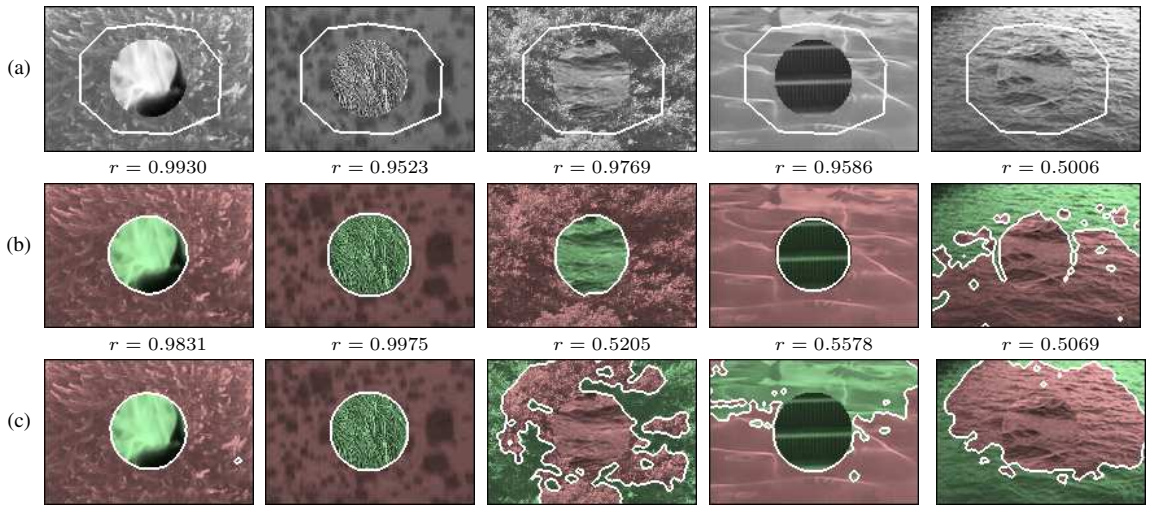

Fig. 7. Examples of segmentation of synthetic database ($K = 2$): a) a video frame and the initial contour; segmentation with: b) DytexMixIC; and c) Ising [22]. The Rand index (r) of each segmentation is shown above the image.

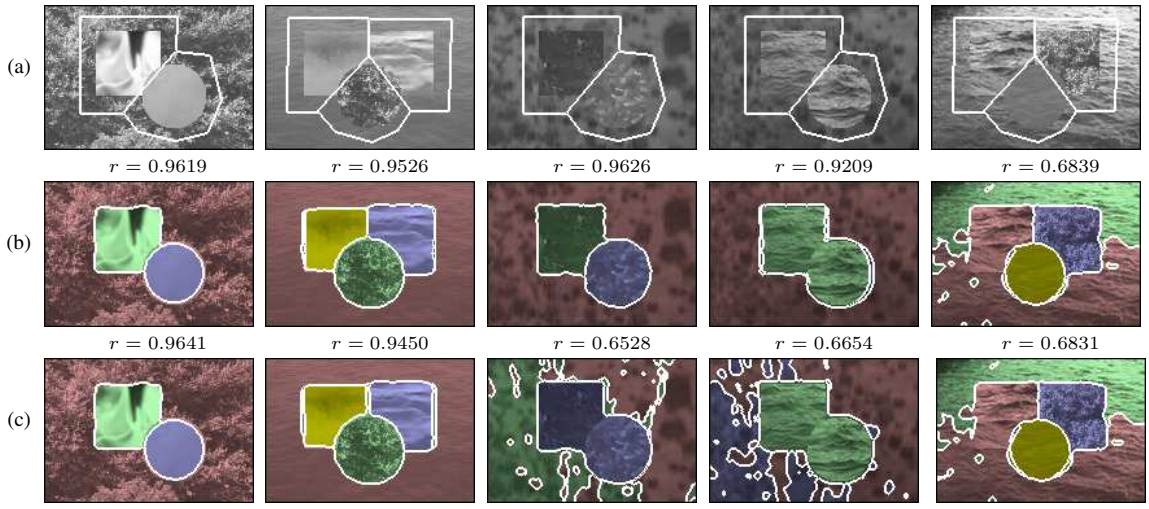

Fig. 8. Examples of segmentation of synthetic database ($K = 3$ and $K = 4$): a) a video frame and the initial contour; b) segmentation using DytexMixIC; and c) DytexMixCS. The Rand index (r) of each segmentation is shown above the image.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

Algorithm	$K = 2$	$K = 3$	$K = 4$
DytexMixIC	**0.915** (17)	**0.853** (15)	**0.868** (15)
DytexMixCS	**0.915** (20)	0.825 (10)	0.835 (15)
Ising [22]	0.879 (05)	n/a	n/a
AR [22]	0.690 (02)	n/a	n/a
AR0 [22]	0.662 (05)	n/a	n/a
GPCA [18]	0.548 (02)	0.554 (17)	0.549 (10)
Baseline Rand.	0.607	0.523	0.501
Baseline Init	0.600	0.684	0.704

TABLE IV

THE BEST AVERAGE RAND INDEX FOR EACH SEGMENTATION ALGORITHM ON THE SYNTHETIC DATABASE. THE ORDER OF THE MODEL (n) IS SHOWN IN PARENTHESIS.

An example of these properties is shown in Figure 7. The first column presents a sequence for which both methods work well, while the second column shows an example where Ising is more accurate near the border, but where DytexMixIC still finds a good segmentation. The third and fourth columns present examples where DytexMixIC succeeds and Ising fails (e.g. in the fourth column, Ising confuses the bright escalator edges with the wave ripples). Finally, both methods fail in the fifth column, due to the similarity of the background and foreground water.

With respect to initialization, it is clear from Figure 6 and Table IV that, even in the absence of an initial contour, the mixture of dynamic textures (DytexMixCS) outperforms all other methods considered, including those that require an initial contour (e.g. Ising) and those that do not (e.g. GPCA). Again, this indicates that video segmentation with the dynamic texture mixture is quite robust. Comparing DytexMixIC and DytexMixCS, the two methods achieve equivalent performance for $K = 2$, with DytexMixIC performing slightly better for $K = 3$ and $K = 4$. With multiple textures, manual specification of the initial contour reduces possible ambiguities due to similarities between pairs of textures. An example is given in the third column of Figure 8, where the two foreground textures are similar, while the background texture could perceptually be split into two regions (particles moving upward and to the right, and those moving upward and to the left). In the absence of initial contour, DytexMixCS prefers this alternative interpretation. Finally, the first two columns of Figure 8 show examples where both methods perform well, while in the fourth and fifth columns both fail (e.g. when two water textures are almost identical).

In summary, the quantitative results on a large database of synthetic video textures indicate that the mixture of dynamic textures (both with or without initial contour specification) is superior to all other video texture segmentation algorithms considered. While Ising also achieves acceptable performance, it has two limitations: 1) it requires an initial contour, and 2) it can only segment video composed of two regions. Finally, AR and GPCA do not perform well on this database.

3) Segmentation of real video: We finish with segmentation results on five real video sequences, depicting a water fountain, highway traffic, and pedestrian crowds. While a precise evaluation is not possible, because there is no segmentation ground-truth, the results are sufficiently different to support a qualitative ranking of the different approaches. Figure 9 illustrates the performance on the first sequence, which depicts a water fountain with three regions of different dynamics: water flowing down a wall, falling water, and turbulent water in a pool. While DytexMixCS separates the three regions, GPCA does not produce a sensible segmentation[5]. This result confirms the observations obtained with the synthetic database of the previous section.

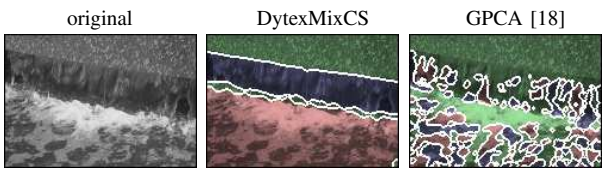

Fig. 9. Segmentation of a fountain scene using DytexMixCS and GPCA.

We next consider macroscopic dynamic textures, in which case segmentation performance is compared against that of traditional motion-based solutions, i.e. the combination of normalized-cuts with motion profiles (NormCuts), or optical flow with mean-shift clustering (OpFlow). Figure 10a shows a scene of highway traffic, for which DytexMixCS is the only method that correctly segments the video into regions of traffic that move away from the camera (the two large regions on the right) and traffic 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. This could be avoided by applying an inverse perspective mapping to the scene, but we have not considered any such geometric transformations. Figure 10b shows another example of segmentation of vehicle traffic 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 NormCuts and OpFlow, segmented regions tend to extend over multiple lanes, incoming and outgoing traffic are merged, and the same lane is frequently broken into a number of sub-regions.

The final two videos are of pedestrian scenes. The first scene, shown in Figure 11a, contains sparse pedestrian traffic, i.e. with large gaps between pedestrians. DytexMixCS (Figure 11b) segments people moving up the walkway from people moving down the walkway. The second scene (Figure 11c) contains a large crowd moving up the walkway, with only a few people moving in the opposite direction. Again, DytexMixCS (Figure 11d) segmented the groups moving in different directions, even in instances where only one person is surrounded by the crowd and moving in the opposite direction. Figures 11e and 11f show the segmentations produced by the traditional methods. The segmentation produced by NormCuts contains gross errors, e.g. frame 2 at the far end of the

[5]The other methods could not be applied to this sequence because it contains three regions.

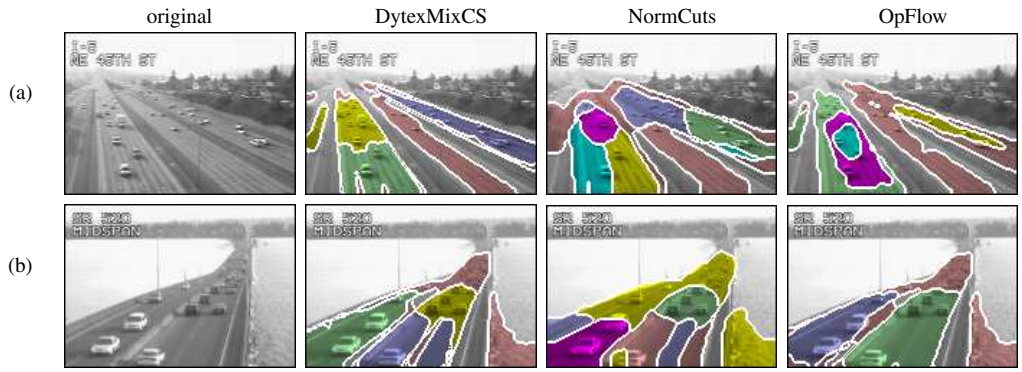

Fig. 10. Segmentation of traffic scenes: (top) highway traffic; (bottom) vehicle traffic on bridge; The left column shows a frame from the original videos, while the remaining columns show the segmentations.

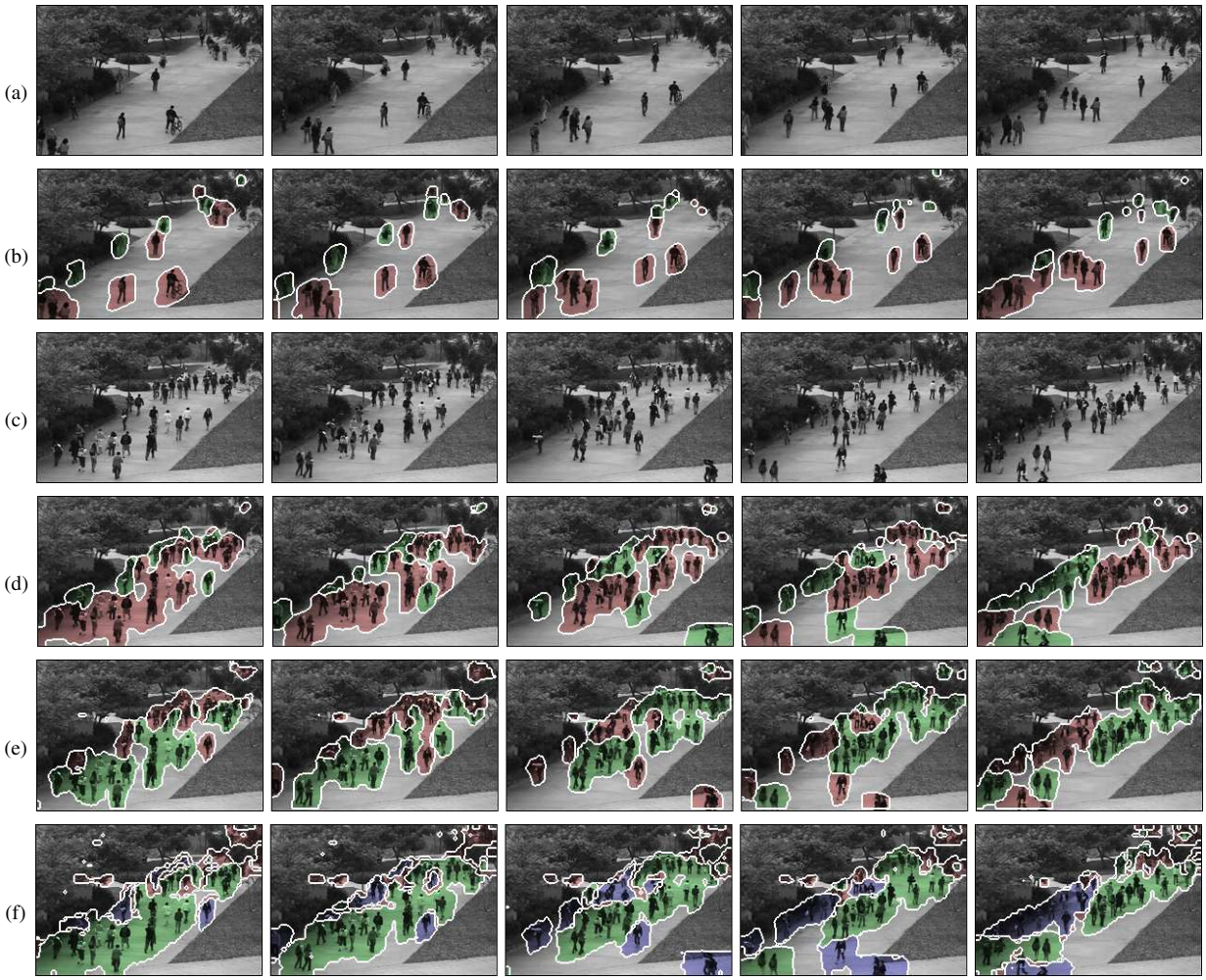

Fig. 11. Segmentation of two pedestrian scenes: (a) pedestrian scene with sparse traffic, and (b) the segmentation by DytexMixCS; (c) Pedestrian scene with heavy traffic, and the segmentations by (d) DytexMixCS; (e) NormCuts, and (f) OpFlow.

walkway. While the segmentation achieved with OpFlow is more comparable to that of DytexMixCS, it tends to over-segment the people moving down the walkway.

Finally, we illustrate the point that the mixture of dynamic textures can be used to efficiently segment very long video sequences. Using the procedure discussed in Section V-B, a continuous hour of pedestrian video was segmented with the mixture model learned from Figure 11c, and is available for visualization in the companion website [58]. It should be noted that this segmentation required no reinitialization at any point, or any other type of manual supervision. We consider this a significant result, given that this sequence contains a fair variability of traffic density, various outlying events (e.g. bicycles, skateboarders, or even small vehicles that pass through, pedestrians that change course or stop to chat, etc.), and variable environmental conditions (such as varying clouds shadows). None of these factors appear to influence

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

the performance of the DytexMixCS algorithm. To the best of our knowledge, this is the first time that a coherent video segmentation, over a time span of this scale, has been reported for a crowded scene.

VII. Conclusions

In this work, we have studied 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, cars, and people), and that the model provides a natural framework for clustering such processes. In the application of motion segmentation, the experimental results indicate that the mixture of dynamic textures provides better segmentations than other state-of-the-art methods, based on either dynamic textures or on traditional representations. Some of the results, namely the segmentation of pedestrian scenes, suggest that the dynamic texture mixture could be the basis for the design of computer vision systems capable of tackling problems, such as the monitoring and surveillance of crowded environments, which currently have great societal interest.

There are also some issues which we leave open for future work. One example is how to incorporate, in the dynamic texture mixture framework, some recent developments in asymptotically efficient estimators based on non-iterative subspace methods [64]. By using such estimators in the M-step of the EM algorithm, it may be possible to reduce the number of hidden variables required in the E-step, and consequently improve the convergence properties of EM. It is currently unclear if the optimality of these estimators is compromised when the initial state has arbitrary covariance, or when the LDS is learned from multiple sample paths, as is the case for dynamic texture mixtures.

Another open question is that of the identifiability of a LDS, when the initial state has arbitrary covariance. It is well known, in the system identification literature, that the parameters of a LDS can only be identified from a single spatio-temporal sample if the covariance of the initial condition satisfies a Lyapunov condition. In the absence of identifiability, the solution may not be unique, may not exist, learning may not converge, or the estimates may not be consistent. It is important to note that none of these problems are of great concern for the methods discussed in this paper. For example, it is well known that EM is guaranteed to converge to a local maximum of the likelihood function, and produces asymptotically consistent parameter estimates. These properties are not contingent on the identifiability of the LDS components. Although the local maxima of the likelihood could be ridges (i.e. not limited to points but supported by manifolds), in which case the optimal component parameters would not be unique, there would be no consequence for segmentation or clustering as long as all computations are based on likelihoods (or other probabilistic distance measures such as the Kullback-Leibler divergence). Non-uniqueness could, nevertheless, be problematic for procedures that rely on direct comparison of the component parameters (e.g. based on their Euclidean distances), which we do not advise. In any case, it would be interesting to investigate more thoroughly the identifiability question. The fact that we have not experienced convergence speed problems, in the extensive experiments discussed above, indicates that likelihood ridges are unlikely. In future work, we will attempt to understand this question more formally.

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 (see Table I for notation). Using (11) and the indicator variables $\mathbf{z}_{i,j}$, the complete-data log-likelihood is

$$\begin{aligned}
\ell(X,Y,Z) &= \sum_{i=1}^{N} \log p(x^{(i)}, y^{(i)}, z^{(i)}) && (30)\\
&= \sum_{i=1}^{N} \log \prod_{j=1}^{K} \left[p(x^{(i)}, y^{(i)}, z^{(i)} = j) \right]^{\mathbf{z}_{i,j}} && (31)\\
&= \sum_{ij} \mathbf{z}_{i,j} \log \Bigg[\alpha_j p(x_1^{(i)} | z^{(i)} = j) && (32)\\
&\quad \cdot \prod_{t=2}^{\tau} p(x_t^{(i)} | x_{t-1}^{(i)}, z^{(i)} = j) \prod_{t=1}^{\tau} p(y_t^{(i)} | x_t^{(i)}, z^{(i)} = j) \Bigg]\\
&= \sum_{i,j} \mathbf{z}_{i,j} \Bigg[\log \alpha_j + \log p(x_1^{(i)} | z^{(i)} = j) && (33)\\
&\quad + \sum_{t=2}^{\tau} \log p(x_t^{(i)} | x_{t-1}^{(i)}, z^{(i)} = j)\\
&\quad + \sum_{t=1}^{\tau} \log p(y_t^{(i)} | x_t^{(i)}, z^{(i)} = j) \Bigg].
\end{aligned}$$

Note that, from (8)-(10), the sums of the log-conditional probability terms are of the form

$$\sum_{i,j} a_{i,j} \sum_{t=t_0}^{t_1} \log G(b_t, c_{j,t}, M_j) = \qquad (34)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

$$-\frac{n}{2}(t_1-t_0+1)\log 2\pi\sum_{i,j}a_{i,j}$$
$$-\frac{1}{2}\sum_{i,j}a_{i,j}\sum_{t=t_0}^{t_1}||b_t-c_{j,t}||^2_{M_j}$$
$$-\frac{t_1-t_0+1}{2}\sum_{i,j}a_{i,j}\log|M_j|.$$

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:

$$\begin{aligned}\ell(X,Y,Z) &= \sum_{i,j}\mathbf{z}_{i,j}\log\alpha_j-\frac{1}{2}\sum_{i,j}\mathbf{z}_{i,j}\log|S_j| \qquad (35)\\ &-\frac{1}{2}\sum_{i,j}\mathbf{z}_{i,j}\left\|x_1^{(i)}-\mu_j\right\|^2_{S_j}-\frac{\tau-1}{2}\sum_{i,j}\mathbf{z}_{i,j}\log|Q_j|\\ &-\frac{1}{2}\sum_{i,j}\mathbf{z}_{i,j}\sum_{t=2}^{\tau}\left\|x_t^{(i)}-A_jx_{t-1}^{(i)}\right\|^2_{Q_j}\\ &-\frac{1}{2}\sum_{i,j}\mathbf{z}_{i,j}\sum_{t=1}^{\tau}\left\|y_t^{(i)}-C_jx_t^{(i)}\right\|^2_{R_j}-\frac{\tau}{2}\sum_{i,j}\mathbf{z}_{i,j}\log|R_j|\end{aligned}$$

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 E-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 $\mathbf{z}_{i,j}f(x^{(i)},y^{(i)})$, for some functions f of $x^{(i)}$ and $y^{(i)}$, and its expectation is

$$\begin{aligned}&\mathrm{E}_{X,Z|Y}\left(\mathbf{z}_{i,j}f(x^{(i)},y^{(i)})\right) &(36)\\ &=\mathrm{E}_{Z|Y}\left(\mathrm{E}_{X|Y,Z}\left(\mathbf{z}_{i,j}f(x^{(i)},y^{(i)})\right)\right) &(37)\\ &=\mathrm{E}_{z^{(i)}|y^{(i)}}\left(\mathrm{E}_{x^{(i)}|y^{(i)},z^{(i)}}\left(\mathbf{z}_{i,j}f(x^{(i)},y^{(i)})\right)\right) &(38)\\ &=p(\mathbf{z}_{i,j}=1|y^{(i)})\mathrm{E}_{x^{(i)}|y^{(i)},z^{(i)}=j}\left(f(x^{(i)},y^{(i)})\right) &(39)\end{aligned}$$

where (38) follows from the assumption that the observations are independent. For the first term of (39), $p(\mathbf{z}_{i,j}=1|y^{(i)})$ is the posterior probability of $z^{(i)}=j$ given the observation $y^{(i)}$,

$$\begin{aligned}\hat{\mathbf{z}}_{i,j} &= p(\mathbf{z}_{i,j}=1|y^{(i)}) = p(z^{(i)}=j|y^{(i)}) &(40)\\ &= \frac{\alpha_j p(y^{(i)}|z^{(i)}=j)}{\sum_{k=1}^K\alpha_k p(y^{(i)}|z^{(i)}=k)}. &(41)\end{aligned}$$

The functions $f(x^{(i)},y^{(i)})$ are at most quadratic in $x_t^{(i)}$. Hence, the second term of (39) only depends on the first and second moments of the states conditioned on $y^{(i)}$ and component j (18-20), and are computed as described in Section III-A. Finally, the $\mathcal{Q}$ function (16) is obtained by first replacing the random variables $\mathbf{z}_{i,j}$, $(\mathbf{z}_{i,j}x_t^{(i)})$, $(\mathbf{z}_{i,j}P_{t,t}^{(i)})$ and $(\mathbf{z}_{i,j}P_{t,t-1}^{(i)})$ in the complete-data log-likelihood (15) with the corresponding expectations $\hat{\mathbf{z}}_{i,j}$, $(\hat{\mathbf{z}}_{i,j}\hat{x}_{t|j}^{(i)})$, $(\hat{\mathbf{z}}_{i,j}\hat{P}_{t,t|j}^{(i)})$, and $(\hat{\mathbf{z}}_{i,j}\hat{P}_{t,t-1|j}^{(i)})$, and then defining the aggregated expectations (17).

C. M-Step

In the M-step of the EM algorithm (14), the reparameterization of the model is obtained by maximizing the $\mathcal{Q}$ 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

$$X^*=\operatorname*{argmax}_X-\frac{1}{2}\mathrm{tr}\left(X^{-1}A\right)-\frac{b}{2}\log|X|. \qquad (42)$$

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

$$\begin{aligned}\frac{\partial}{\partial X}\frac{-1}{2}\mathrm{tr}\left(X^{-1}A\right)-\frac{b}{2}\log|X| &= 0 &(43)\\ \frac{1}{2}X^{-T}A^TX^{-T}-\frac{b}{2}X^{-T} &= 0 &(44)\\ A^T-bX^T &= 0 &(45)\\ \Rightarrow\quad X^*=\frac{1}{b}A. & &(46)\end{aligned}$$

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

$$X^*=\operatorname*{argmax}_X-\frac{1}{2}\mathrm{tr}\left[D(-BX^T-XB^T+XCX^T)\right] \qquad (47)$$

where D and C are symmetric and invertible matrices. The maximum is given by

$$\begin{aligned}\frac{\partial}{\partial X}\frac{-1}{2}\mathrm{tr}\left[D(-BX^T-XB^T+XCX^T)\right] &= 0 &(48)\\ -\frac{1}{2}(-DB-D^TB+D^TXC^T+DXC) &= 0 &(49)\\ DB-DXC &= 0 &(50)\\ \Rightarrow\quad X^*=BC^{-1}. & &(51)\end{aligned}$$

The optimal parameters are found by collecting the relevant terms in (16) and maximizing.

1) Observation Matrix:

$$C_j^*=\operatorname*{argmax}_{C_j}-\frac{1}{2}\mathrm{tr}\left[R_j^{-1}\left(-\Gamma_jC_j^T-C_j\Gamma_j^T+C_j\Phi_jC_j^T\right)\right]$$

This is of the form in (47), hence the solution is given by $C_j^*=\Gamma_j\left(\Phi_j\right)^{-1}$.

2) Observation Noise Covariance:

$$\begin{aligned}R_j^* &= \operatorname*{argmax}_{R_j}-\frac{\tau\hat{N}_j}{2}\log|R_j| &(52)\\ &\quad-\frac{1}{2}\mathrm{tr}\left[R_j^{-1}\left(\Lambda_j-\Gamma_jC_j^T-C_j\Gamma_j^T+C_j\Phi_jC_j^T\right)\right]\end{aligned}$$

This is of the form in (42), hence the solution is

$$\begin{aligned}R_j^* &= \frac{1}{\tau\hat{N}_j}\left(\Lambda_j-\Gamma_jC_j^T-C_j\Gamma_j^T+C_j\Phi_jC_j^T\right) &(53)\\ &= \frac{1}{\tau\hat{N}_j}\left(\Lambda_j-C_j^*\Gamma_j^T\right) &(54)\end{aligned}$$

where (54) follows from substituting for the optimal value C_j^*.

3) State Transition Matrix:

$$A_j^* = \underset{A_j}{\operatorname{argmax}} -\frac{1}{2}\mathrm{tr}\left[Q_j^{-1}\left(-\Psi_j A_j^T - A_j\Psi_j^T + A_j\phi_j A_j^T\right)\right]$$

This is of the form in (47), hence $A_j^* = \Psi_j\left(\phi_j\right)^{-1}$.

4) State Noise Covariance:

$$\begin{aligned} Q_j^* &= \underset{Q_j}{\operatorname{argmax}} -\frac{(\tau-1)\hat{N}_j}{2}\log|Q_j| \\ &\quad -\frac{1}{2}\mathrm{tr}\left[Q_j^{-1}\left(\varphi_j - \Psi_j A_j^T - A_j\Psi_j^T + A_j\phi_j A_j^T\right)\right] \end{aligned} \tag{55}$$

This is of the form in (42), hence the solution can be computed as

$$\begin{aligned} Q_j^* &= \frac{1}{(\tau-1)\hat{N}_j}\left(\varphi_j - \Psi_j A_j^T - A_j\Psi_j^T + A_j\phi_j A_j^T\right) \\ &= \frac{1}{(\tau-1)\hat{N}_j}\left(\varphi_j - A_j^*\Psi_j^T\right). \end{aligned} \tag{56}$$

where (56) follows from substituting for the optimal value A_j^*.

5) Initial State Mean:

$$\mu_j^* = \underset{\mu_j}{\operatorname{argmax}} -\frac{1}{2}\mathrm{tr}\left[S_j^{-1}\left(-\xi_j\mu_j^T - \mu_j\xi_j^T + \hat{N}_j\mu_j\mu_j^T\right)\right]$$

This is of the form in (47), hence the solution is given by $\mu_j^* = \frac{1}{\hat{N}_j}\xi_j$.

6) Initial State Covariance:

$$\begin{aligned} S_j^* &= \underset{S_j}{\operatorname{argmax}} -\frac{\hat{N}_j}{2}\log|S_j| \\ &\quad -\frac{1}{2}\mathrm{tr}\left[S_j^{-1}\left(\eta_j - \xi_j\mu_j^T - \mu_j\xi_j^T + \hat{N}_j\mu_j\mu_j^T\right)\right] \end{aligned} \tag{57}$$

This is of the form in (42), hence the solution is given by

$$S_j^* = \frac{1}{\hat{N}_j}\left(\eta_j - \xi_j\mu_j^T - \mu_j\xi_j^T + \hat{N}_j\mu_j\mu_j^T\right) \tag{58}$$

$$= S_j^* = \frac{1}{\hat{N}_j}\eta_j - \mu_j^*(\mu_j^*)^T. \tag{59}$$

where (59) follows from substituting for the optimal value μ_j^*.

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

$$\alpha = \underset{\alpha,\lambda}{\operatorname{argmax}} \sum_j \hat{N}_j\log\alpha_j + \lambda\left(\sum_j \alpha_j - 1\right) \tag{60}$$

where $\alpha = \{\alpha_1, \cdots, \alpha_K\}$. The optimal value is $\alpha_j^* = \frac{\hat{N}_j}{N}$.

APPENDIX II
KALMAN SMOOTHING FILTER

The Kalman smoothing filter [24], [25] estimates the mean and covariance of the state x_t of an LDS, conditioned on the entire observed sequence $\{y_1, \ldots, y_\tau\}$. It can also be used to efficiently compute the log-likelihood of the observed sequence. Define the expectations conditioned on the observed sequence from time $t = 1$ to $t = s$,

$$\hat{x}_t^s = \mathrm{E}_{x|y_1,\ldots,y_s}(x_t) \tag{61}$$

$$\hat{V}_t^s = \mathrm{E}_{x|y_1,\ldots,y_s}((x_t - \hat{x}_t^s)(x_t - \hat{x}_t^s)^T) \tag{62}$$

$$\hat{V}_{t,t-1}^s = \mathrm{E}_{x|y_1,\ldots,y_s}((x_t - \hat{x}_t^s)(x_{t-1} - \hat{x}_{t-1}^s)^T) \tag{63}$$

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

$$\hat{V}_t^{t-1} = A\hat{V}_{t-1}^{t-1}A^T + Q, \tag{64}$$

$$K_t = \hat{V}_t^{t-1}C^T(C\hat{V}_t^{t-1}C^T + R)^{-1} \tag{65}$$

$$\hat{V}_t^t = \hat{V}_t^{t-1} - K_tC\hat{V}_t^{t-1} \tag{66}$$

$$\hat{x}_t^{t-1} = A\hat{x}_{t-1}^{t-1} \tag{67}$$

$$\hat{x}_t^t = \hat{x}_t^{t-1} + K_t(y_t - C\hat{x}_t^{t-1}) \tag{68}$$

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

$$J_{t-1} = \hat{V}_{t-1}^{t-1}A^T(\hat{V}_t^{t-1})^{-1} \tag{69}$$

$$\hat{x}_{t-1}^\tau = \hat{x}_{t-1}^{t-1} + J_{t-1}(\hat{x}_t^\tau - A\hat{x}_{t-1}^{t-1}) \tag{70}$$

$$\hat{V}_{t-1}^\tau = \hat{V}_{t-1}^{t-1} + J_{t-1}(\hat{V}_t^\tau - \hat{V}_t^{t-1})J_{t-1}^T \tag{71}$$

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

$$\hat{V}_{t-1,t-2}^\tau = \hat{V}_{t-1}^{t-1}J_{t-2}^T + J_{t-1}(\hat{V}_{t,t-1}^\tau - A\hat{V}_{t-1}^{t-1})J_{t-2}^T \tag{72}$$

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

$$\log p(y_1^\tau) = \sum_{t=1}^{\tau}\log p(y_t|y_1^{t-1}) \tag{73}$$

$$= \sum_{t=1}^{\tau}\log G(y_t, C\hat{x}_t^{t-1}, C\hat{V}_t^{t-1}C^T + R) \tag{74}$$

If R is an i.i.d. or diagonal covariance matrix (e.g. $R = rI_m$), then the filter can be computed efficiently using the matrix inversion lemma.

ACKNOWLEDGMENTS

The authors thank Gianfranco Doretto and Stefano Soatto for the synthetic sequences from [15], [16], the Washington State DOT [60] for the videos of highway traffic, Daniel Dailey for the loop-sensor data, Rene Vidal, Dheeraj Singaraju, and Atiyeh Ghoreyshi for code from [18], [22], Pedro Moreno for helpful discussions, and the anonymous reviewers for insightful comments. This work was funded by NSF award IIS-0534985, and NSF IGERT award DGE-0333451.

REFERENCES

[1] B. K. P. Horn, *Robot Vision*. McGraw-Hill Book Company, New York, 1986.

[2] B. Horn and B. Schunk, "Determining optical flow," *Artificial Intelligence*, vol. 17, pp. 185–204, 1981.

[3] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. DARPA Image Understanding Workshop*, 1981, pp. 121–130.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

[4] J. Barron, D. Fleet, and S. Beauchemin, "Performance of optical flow techniques," *IJCV*, vol. 12, pp. 43–77, 1994.

[5] P. Anandan, J. Bergen, K. Hanna, and R. Hingorani, "Hierarchical model-based motion estimation," *Motion Analysis and Image Sequence Processing*, pp. 1–22, 1993.

[6] J. Wang and E. Adelson, "Representing moving images with layers," *IEEE Trans. IP*, vol. 3, no. 5, pp. 625–38, 1994.

[7] H. Sawhney and S. Ayer, "Compact representations of videos through dominant and multiple motion estimation," *IEEE Trans. in Pattern Analysis and Machine Intelligence*, vol. 18, no. 8, pp. 814–30, 1996.

[8] M. Hansen, P. Anandan, K. Dana, G. Wal, and P. Burt, "Real-time scene stabilization and mosaic construction," in *Proc. DARPA Image Understanding Workshop*, 1994, pp. 457–63.

[9] M. Isard and A. Blake, "Condensation - conditional density propagation for visual tracking," *IJCV*, vol. 29(1), pp. 5–28, 1998.

[10] M. Irani, B. Rousso, and S. Peleg, "Detecting and tracking multiple moving objects using temporal integration," in *European Conference on Computer Vision*, 1992, pp. 282–7.

[11] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE TPAMI*, vol. 25, no. 5, pp. 564–75, 2003.

[12] S. Soatto, G. Doretto, and Y. N. Wu, "Dynamic textures," in *IEEE Intl. Conf. on Computer Vision*, 2001, pp. 439–46.

[13] G. Doretto, A. Chiuso, Y. N. Wu, and S. Soatto, "Dynamic textures," *IJCV*, vol. 51, no. 2, pp. 91–109, 2003.

[14] A. W. Fitzgibbon, "Stochastic rigidity: image registration for nowhere-static scenes," in *ICCV*, vol. 1, 2001, pp. 662–70.

[15] G. Doretto, D. Cremers, P. Favaro, and S. Soatto, "Dynamic texture segmentation," in *ICCV*, vol. 2, 2003, pp. 1236–42.

[16] P. Saisan, G. Doretto, Y. Wu, and S. Soatto, "Dynamic texture recognition," in *CVPR*, vol. 2, 2001, pp. 58–63.

[17] A. B. Chan and N. Vasconcelos, "Probabilistic kernels for the classification of auto-regressive visual processes," in *IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1, 2005, pp. 846–851.

[18] R. Vidal and A. Ravichandran, "Optical flow estimation & segmentation of multiple moving dynamic textures," in *IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 2, 2005, pp. 516–21.

[19] A. B. Chan and N. Vasconcelos, "Layered dynamic textures," in *NIPS 18*, 2006, pp. 203–10.

[20] ——, "Mixtures of dynamic textures," in *IEEE Intl. Conf. on Computer Vision*, vol. 1, 2005, pp. 641–7.

[21] L. Cooper, J. Liu, and K. Huang, "Spatial segmentation of temporal texture using mixture linear models," in *Dynamical Vision Workshop in the IEEE Intl. Conf. of Computer Vision*, 2005.

[22] A. Ghoreyshi and R. Vidal, "Segmenting dynamic textures with Ising descriptors, ARX models and level sets," in *Dynamical Vision Workshop in the European Conf. on Computer Vision*, 2006.

[23] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice-Hall, 1993.

[24] S. Roweis and Z. Ghahramani, "A unifying review of linear Gaussian models," *Neur. Comp.*, vol. 11(2), pp. 305–45, 1999.

[25] R. H. Shumway and D. S. Stoffer, "An approach to time series smoothing and forecasting using the EM algorithm," *Journal of Time Series Analysis*, vol. 3, no. 4, pp. 253–264, 1982.

[26] P. V. Overschee and B. D. Moor, "N4SID: Subspace algorithms for the identification of combined deterministic-stochastic systems," *Automatica*, vol. 30, pp. 75–93, 1994.

[27] F. V. Jensen, *Bayesian Networks and Decision Graphs*. Springer-Verlag, 2001.

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

[29] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. Y. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, *The HTK Book*. Cambridge University Engineering Dept., 2006.

[30] D. T. Magill, "Optimal adaptive estimation of sampled stochastic processes," *IEEE TAC*, vol. 10(4), pp. 434–9, 1965.

[31] D. G. Lainiotis, "Partitioning: a unifying framework for adaptive systems, I: Estimation; II: Control," *Proc. IEEE*, vol. 64, no. 8, pp. 1126–43,1182–98, 1976.

[32] K. S. Narendra and J. Balakrishnan, "Adaptive control using multiple models," *IEEE TAC*, vol. 42(2), pp. 171–87, 1997.

[33] R. G. Brown, "A new look at Magill adaptive filter as a practical means of multiple hypothesis testing," *IEEE Trans. on Circuits and Systems*, vol. 30, no. 10, pp. 765–8, 1983.

[34] V. Digalakis, J. R. Rohlicek, and M. Ostendorf, "ML estimation of a stochastic linear system with the EM algorithm and its application to speech recognition," *IEEE Trans. on Speech and Audio Processing*, vol. 1, no. 4, pp. 431–442, 1993.

[35] Z. Ghahramani and G. Hinton, "Parameter estimation for linear dynamical systems," Department of Computer Science, University of Toronto," Tech Report CRG-TR-96-2, 1996.

[36] ——, "The EM algorithm for mixtures of factor analyzers," Department of Computer Science, University of Toronto," Tech Report CRG-TR-96-1, 1997.

[37] R. Shumway and D. Stoffer, "Dynamic linear models with switching," *J. Amer. Stat. Assoc.*, vol. 86, pp. 763–769, 1991.

[38] Y. Wu, G. Hua, and T. Yu, "Switching observation models for contour tracking in clutter," in *CVPR*, 2003, pp. 295–302.

[39] M. Isard and A. Blake, "A mixed-state condensation tracker with automatic model-switching," in *ICCV*, 1998, pp. 107–12.

[40] V. Pavlović, B. J. Frey, and T. S. Huang, "Time-series classification using mixed-state dynamic Bayesian networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 1999.

[41] V. Pavlović, J. Rehg, and J. MacCormick, "Learning switching linear models of human motion," in *NIPS 13*, 2000.

[42] C.-J. Kim, "Dynamic linear models with Markov-switching," *Journal of Econometrics*, vol. 60, pp. 1–22, 1994.

[43] S. M. Oh, J. M. Rehg, T. Balch, and F. Dellaert, "Learning and inference in parametric switching linear dynamic systems," in *IEEE International Conference on Computer Vision*, vol. 2, 2005, pp. 1161–8.

[44] Z. Ghahramani and G. E. Hinton, "Variational learning for switching state-space models," *Neur. Comp.*, vol. 12(4), pp. 831–64, 2000.

[45] T. W. Liao, "Clustering of time series data – a survey," *Pattern Recognition*, vol. 38, pp. 1857–74, 2005.

[46] Y. Kakizawa, R. H. Shumway, and M. Taniguchi, "Discrimination and clustering for multivariate time series," *Journal of the American Statistical Association*, vol. 93, no. 441, pp. 328–40, 1998.

[47] A. Singhal and D. E. Seborg, "Clustering of multivariate time-series data," in *Amer. Control Conf.*, vol. 5, 2002, pp. 3931–6.

[48] Y. Xiong and D.-Y. Yeung, "Time series clustering with ARMA mixtures," *Pattern Recogn.*, vol. 37, pp. 1675–89, 2004.

[49] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Prentice Hall, 2002.

[50] R. Duda, P. Hart, and D. Stork, *Pattern Classification*. John Wiley and Sons, 2001.

[51] C. Stauffer and E. Grimson, "Learning patterns of activity using real-time tracking," *TPAMI*, vol. 22(8), pp. 747–57, 2000.

[52] A. Jepson and M. Black, "Mixture models for optical flow computation," in *CVPR*, 1993, pp. 760–1.

[53] Y. Weiss, "Smoothness in layers: Motion segmentation using nonparametric mixture estimation," in *ICCV*, 1997, pp. 520–6.

[54] N. Vasconcelos and A. Lippman, "Empirical bayesian motion segmentation," *IEEE TPAMI*, vol. 23(2), pp. 217–21, 2001.

[55] B. Frey and N. Jojic, "Estimating mixture models of images and inferring spatial transformations using the EM algorithm," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 1999, pp. 416–22.

[56] C. Carson, S. Belongie, H. Greenspan, and J. Malik, "Blobworld: Color- and texture-based image segmentation using EM and its application to image querying and classification," *IEEE Trans. on PAMI*, vol. 24, no. 8, pp. 1026–38, 2002.

[57] N. Vasconcelos, "Minimum probability of error image retrieval," *IEEE Trans. Signal Proc.*, vol. 52(8), pp. 2322–36, 2004.

[58] "Mixtures of dynamic textures." [Online]. Available: http://www.svcl.ucsd.edu/projects/motiondytex

[59] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, pp. 193–218, 1985.

[60] "Washington state department of transportation." [Online]. Available: http://www.wsdot.wa.gov

[61] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE TPAMI*, vol. 22(8), pp. 888–905, 2000.

[62] ——, "Motion segmentation and tracking using normalized cuts," in *ICCV*, 1999, pp. 1154–60.

[63] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," *TPAMI*, vol. 24(5), pp. 603–19, 2002.

[64] D. Bauer, "Comparing the CCA subspace method to pseudo maximum likelihood methods in the case of no exogenous inputs," *Journal of Time Series Analysis*, vol. 26, pp. 631–668, 2005.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

Antoni B. Chan received the BS and MEng degrees in electrical engineering from Cornell University in 2000 and 2001, respectively. He is a PhD student in the Statistical Visual Computing Lab in the ECE Department at the University of California, San Diego. In 2005, he was a summer intern at Google in New York City, and from 2001 to 2003, he was a visiting scientist in the Vision and Image Analysis Lab at Cornell.

Nuno Vasconcelos received the licenciatura in electrical engineering and computer science from the Universidade do Porto, Portugal, in 1988, and the MS and PhD degrees from the Massachusetts Institute of Technology in 1993 and 2000, respectively. From 2000 to 2002, he was a member of the research staff at the Compaq Cambridge Research Laboratory, which in 2002 became the HP Cambridge Research Laboratory. In 2003, he joined the Electrical and Computer Engineering Department at the University of California, San Diego, where he heads the Statistical Visual Computing Laboratory. He is the recipient of a US National Science Foundation CAREER award, a Hellman Fellowship, and has authored more than 50 peer-reviewed publications. His work spans various areas, including computer vision, machine learning, signal processing and compression, and multimedia systems. He is a member of the IEEE.