# Layered Dynamic Textures

Antoni B. Chan, *Member*, *IEEE*, and Nuno Vasconcelos, *Senior Member*, *IEEE*

**Abstract**—A novel video representation, the *layered dynamic texture* (LDT), is proposed. The LDT is a generative model, which represents a video as a collection of stochastic layers of different appearance and dynamics. Each layer is modeled as a temporal texture sampled from a different linear dynamical system. The LDT model includes these systems, a collection of hidden layer assignment variables (which control the assignment of pixels to layers), and a Markov random field prior on these variables (which encourages smooth segmentations). An EM algorithm is derived for maximum-likelihood estimation of the model parameters from a training video. It is shown that exact inference is intractable, a problem which is addressed by the introduction of two approximate inference procedures: a Gibbs sampler and a computationally efficient variational approximation. The trade-off between the quality of the two approximations and their complexity is studied experimentally. The ability of the LDT to segment videos into layers of coherent appearance and dynamics is also evaluated, on both synthetic and natural videos. These experiments show that the model possesses an ability to group regions of *globally homogeneous*, but *locally heterogeneous*, *stochastic dynamics* currently unparalleled in the literature.

**Index Terms**—Dynamic texture, temporal textures, video modeling, motion segmentation, mixture models, linear dynamical systems, Kalman filter, Markov random fields, probabilistic models, expectation-maximization, variational approximation, Gibbs sampling.

## 1 Introduction

Traditional motion representations, based on optical flow, are inherently local and have significant difficulties when faced with aperture problems and noise. The classical solution to this problem is to regularize the optical flow field [1], [2], [3], [4], but this introduces undesirable smoothing across motion edges or regions where the motion is, by definition, not smooth (e.g., vegetation in outdoors scenes). It does not also provide any information about the objects that compose the scene, although the optical flow field could be subsequently used for motion segmentation. More recently, there have been various attempts to model videos as a superposition of layers subject to homogeneous motion. While layered representations exhibited significant promise in terms of combining the advantages of regularization (use of global cues to determine local motion) with the flexibility of local representations (little undue smoothing), and a truly object-based representation, this potential has so far not fully materialized. One of the main limitations is their dependence on parametric motion models, such as affine transforms, which assume a piecewise planar world that rarely holds in practice [5], [6]. In fact, layers are usually formulated as "cardboard" models of the world that are warped by such transformations, and then, stitched to form the frames in a video stream [5]. This severely limits the types of videos that can be synthesized: while the concept of layering showed most promise for the representation of scenes composed of ensembles of objects subject to homogeneous motion (e.g., leaves blowing in the wind, a flock of birds, a picket fence, or highway traffic), very little progress has so far been demonstrated in actually modeling such scenes.

Recently, there has been more success in modeling complex scenes as *dynamic textures* or more precisely, samples from stochastic processes defined over space and time [7], [8], [9], [10]. This work has demonstrated that *global stochastic modeling* of both video dynamics and appearance is much more powerful than the classic global modeling as "cardboard" figures under parametric motion. In fact, the dynamic texture (DT) has shown a surprising ability to abstract a wide variety of complex patterns of motion and appearance into a *simple* spatiotemporal model. One major current limitation is, however, its inability to decompose visual processes consisting of *multiple, co-occurring, and dynamic textures*, for example, a flock of birds flying in front of a water fountain or highway traffic moving at different speeds, *into separate regions of distinct but homogeneous dynamics*. In such cases, the global nature of the existing DT model makes it inherently ill-equipped to segment the video into its constituent regions.

To address this problem, various extensions of the DT have been recently proposed in the literature [8], [11], [12]. These extensions have emphasized the *application* of the standard DT model to video segmentation, rather than exploiting the probabilistic nature of the DT representation to propose a *global generative model for videos*. They represent the video as a collection of *localized* spatiotemporal patches (or pixel trajectories). These are then modeled with dynamic textures, or similar time-series representations, whose parameters are *clustered* to produce the desired segmentations. Due to their local character, these representations cannot account for *globally homogeneous* textures that exhibit substantial *local heterogeneity*. These types of textures are common in both urban settings, where the video dynamics

• *The authors are with the Department of Electrical and Computer Engineering, University of California, San Diego, 9500 Gilman Drive, Mail Code 0407, La Jolla, CA 92093-0409.*
*E-mail: abchan@ucsd.edu, nuno@ece.ucsd.edu.*

*Manuscript received 15 Aug. 2008; revised 25 Dec. 2008; accepted 23 Apr. 2009; published online 5 May 2009.*
*Recommended for acceptance by Q. Ji, A. Torralba, T. Huang, E. Sudderth, and J. Luo.*
*For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMISI-2008-08-0516.*
*Digital Object Identifier no. 10.1109/TPAMI.2009.110.*

frequently combine global motion and stochasticity (e.g., vehicle traffic around a square, or pedestrian traffic around a landmark), and natural scenes (e.g., a flame that tilts under the influence of the wind, or water rotating in a whirlpool).

In this work, we address this limitation by introducing a new generative model for videos, which we denote by the *layered dynamic texture* (LDT) [13]. This consists of augmenting the dynamic texture with a discrete *hidden* variable that enables the assignment of different dynamics to different regions of the video. The hidden variable is modeled as a Markov random field (MRF) to ensure spatial smoothness of the regions, and conditioned on the state of this hidden variable, each region of the video is a standard DT. By introducing a shared dynamic representation for all pixels in a region, the new model is a layered representation. When compared with traditional layered models, it replaces layer formation by "warping cardboard figures" with sampling from the generative model (for both dynamics and appearance) provided by the DT. This enables a much richer video representation. Since each layer is a DT, the model can also be seen as a *multistate dynamic texture*, which is capable of assigning different dynamics and appearance to different image regions. In addition to introducing the model, we derive the EM algorithm for maximum-likelihood estimation of its parameters from an observed video sample. Because exact inference is computationally intractable (due to the MRF), we present two approximate inference algorithms: a Gibbs sampler and a variational approximation. Finally, we apply the LDT to motion segmentation of challenging video sequences.

The remainder of the paper is organized as follows: We begin with a brief review of previous work in DT models in Section 2. In Section 3, we introduce the LDT model. In Section 4, we derive the EM algorithm for parameter learning. Sections 5 and 6 then propose the Gibbs sampler and variational approximation. Finally, in Section 7, we present an experimental evaluation of the two approximate inference algorithms and apply the LDT to motion segmentation of both synthetic and real video sequences.

## 2 Related Work

The DT is a generative model for videos, which accounts for both the appearance and dynamics of a video sequence by modeling it as an observation from a linear dynamical system (LDS) [7], [14]. It can be thought of as an extension of the hidden Markov models commonly used in speech recognition, and is the model that underlies the Kalman filter frequently employed in control systems. It combines the notion of a hidden state, which encodes the dynamics of the video, and an observed variable, from which the appearance component is drawn at each time instant, conditionally on the state at that instant. The DT and its extensions have a variety of computer vision applications, including video texture synthesis [7], [15], [16], video clustering [17], image registration [18], and motion classification [9], [10], [16], [19], [20], [21], [22].

The original DT model has been extended in various ways in the literature. Some of these extensions aim to overcome limitations of the learning algorithm. For example, Siddiqi et al. [23] learn a stable DT, which is more suitable for synthesis, by iteratively adding constraints to the least-squares problem of [7]. Other extensions aim to overcome limitations of the original model. To improve the DT's ability for texture synthesis, Yuan et al. [15] model the hidden state variable with a closed-loop dynamic system that uses a feedback mechanism to correct errors with respect to a reference signal, while Costantini et al. [24] decompose videos as a multidimensional tensor using a higher order SVD. Spatially and temporally, homogeneous texture models are proposed in [25], using STAR models, and in [26], using a multiscale autoregressive process. Other extensions aim to increase the representational power of the model, e.g., by introducing nonlinear observation functions. Liu et al. [27], [28] model the observation function as a mixture of linear subspaces (i.e., a mixture of PCA), which are combined into a global coordinate system, while Chan and Vasconcelos [20] learn a nonlinear observation function with kernel PCA. Doretto and Soatto [29] extend the DT to represent dynamic shape and appearance. Finally, Ghanem and Ahuja [16] introduce a nonparametric probabilistic model that relates texture dynamics with variations in Fourier phase, capturing both the global coherence of the motion within the texture and its appearance. Like the original DT, all these extensions are global models with a single-state variable. This limits their ability to model a video as a composition of distinct regions of coherent dynamics and prevents their direct application to video segmentation.

A number of applications of DT (or similar) models to segmentation have also been reported in the literature. Doretto et al. [8] model spatiotemporal patches extracted from a video as DTs, and cluster them using level sets and the Martin distance. Ghoreyshi and Vidal [12] cluster pixel intensities (or local texture features) using autoregressive processes (AR) and level sets. Vidal and Ravichandran [11] segment a video by clustering pixels with similar trajectories in time using generalized PCA (GPCA), while Vidal [30] clusters pixel trajectories lying in multiple moving hyperplanes using an online recursive algorithm to estimate polynomial coefficients. Cooper et al. [31] cluster spatiotemporal cubes using robust GPCA. While these methods have shown promise, they do not exploit the probabilistic nature of the DT representation for the segmentation itself. A different approach is proposed by [32], which segments high-density crowds with a Lagrangian particle dynamics model, where the flow field generated by a moving crowd is treated as an aperiodic dynamical system. Although promising, this work is limited to scenes where the optical flow can be reliably estimated (e.g., crowds of moving people, but not moving water).

More related to the extensions proposed in this work is the *dynamic texture mixture* (DTM) that we have previously proposed in [17]. This is a model for collections of video sequences, and has been successfully used for motion-based video segmentation through clustering of spatiotemporal patches. The main difference with respect to the LDT now proposed is that (like all clustering models) the DTM is not a *global generative model* for videos of *co-occurring* textures (as is the case of the LDT). Hence, the application of the DTM to segmentation requires decomposing the video into a collection of small spatiotemporal patches, which are then

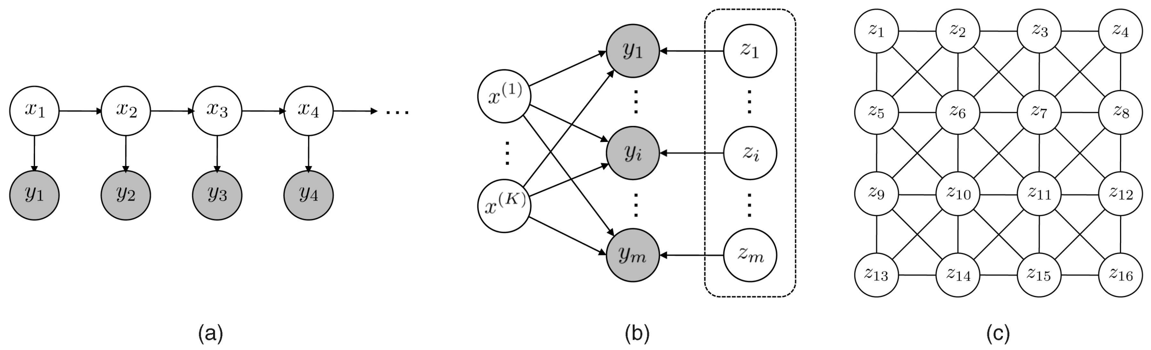

Fig. 1. (a) Graphical model of the DT. $x_t$ and $y_t$ are the hidden state and observed video frame at time $t$. (b) Graphical model of the LDT. $y_i$ is an observed pixel process and $x^{(j)}$ a hidden state process. $z_i$ assigns $y_i$ to one of the state processes, and the collection $\{z_i\}$ is modeled as an MRF. (c) Example of a $4 \times 4$ layer assignment MRF.

clustered. The *localized* nature of this video representation is problematic for the segmentation of textures which are *globally* homogeneous but exhibit substantial variation between neighboring locations, such as the rotating motion of the water in a whirlpool. Furthermore, patch-based segmentations have poor boundary accuracy, due to the artificial boundaries of the underlying patches, and the difficulty of assigning a patch that overlaps multiple regions to any of them.

On the other hand, the LDT models videos as a collection of layers, offering a truly *global* model of the appearance and dynamics of each layer, and avoiding boundary uncertainty. With respect to time-series models, the LDT is related to switching linear dynamical models, which are LDSs that can switch between different parameter sets over time [33], [34], [35], [36], [37], [38], [39], [40]. In particular, it is most related to the switching state-space LDS [40], which models the observed variable by switching between the outputs of a set of independent LDSs. The fundamental difference between the two models is that while Ghahramani and Hinton [40] switch parameters in *time* using a hidden-Markov model (HMM), the LDT switches parameters in *space* (i.e., within the dimensions of the observed variable) using an MRF. This substantially complicates all statistical inference, leading to very different algorithms for learning and inference with LDTs.

## 3 Layered Dynamic Textures

We begin with a review of the DT, followed by the introduction of the LDT model.

### 3.1 Dynamic Texture

A DT [7] is a generative model, which treats a video as a sample from an LDS. The model separates the visual component and the underlying dynamics into two stochastic processes; dynamics are represented as a time-evolving hidden state process $x_t \in \mathbb{R}^n$ and observed video frames $y_t \in \mathbb{R}^m$ as linear functions of the state vector. Formally, the DT has the graphical model of Fig. 1a and system equations

$$\begin{cases} x_t = Ax_{t-1} + v_t, \\ y_t = Cx_t + w_t + \bar{y}, \end{cases} \tag{1}$$

where $A \in \mathbb{R}^{n\times n}$ is a transition matrix, $C \in \mathbb{R}^{m\times n}$ is an observation matrix, and $\bar{y} \in \mathbb{R}^m$ is the observation mean. The state and observation noise processes are normally distributed, as $v_t \sim \mathcal{N}(0, Q)$ and $w_t \sim \mathcal{N}(0, R)$, where $Q \in \mathbb{S}^n_+$ and $R \in \mathbb{S}^m_+$ (typically, $R$ is assumed i.i.d., $R = rI_m$) and $\mathbb{S}^n_+$ is the set of positive-definite $n \times n$ symmetric matrices. The initial state is distributed as $x_1 \sim \mathcal{N}(\xi, Q)$, where $\xi \in \mathbb{R}^n$ is the initial condition of the state sequence.[1] There are several methods for estimating the parameters of the DT from training data, including maximum-likelihood estimation (e.g., EM algorithm [41]), noniterative subspace methods (e.g., N4SID [42], CCA [43]), or least squares [7].

One interpretation of the DT model, when the columns of $C$ are orthonormal (e.g., when learned with [7]), is that they are the principal components of the video sequence. Under this interpretation, the state vector is the set of PCA coefficients of each video frame and evolves according to a Gauss-Markov process (in time). An alternative interpretation considers a single pixel as it evolves over time. Each coordinate of the state vector $x_t$ defines a one-dimensional temporal trajectory, and the pixel value is a weighted sum of these trajectories, according to the weighting coefficients in the corresponding row of $C$. This is analogous to the discrete Fourier transform, where a signal is represented as a weighted sum of complex exponentials but, for the DT, the trajectories are not necessarily orthogonal. This interpretation illustrates the ability of the DT to model a given motion at different intensity levels (e.g., cars moving from the shade into sunlight) by simply scaling rows of $C$. Regardless of the interpretation, the DT is a global model, and thus, unable to represent a video as a composition of homogenous regions with distinct appearance and dynamics.

### 3.2 Layered Dynamic Textures

In this work, we consider videos composed of various textures, e.g., the combination of fire, smoke, and water shown on the right side of Fig. 2. As shown in Fig. 2, this type of video can be modeled by encoding each texture as a separate *layer*, with its own state sequence and observation matrix. Different regions of the spatiotemporal video volume are assigned to each texture, and conditioned on this assignment, each region evolves as a standard DT. The video is a composite of the various layers.

1. By including the initial state $\xi$ (i.e., initial frame), the DT represents both the transient and stationary behaviors of the observed video. Alternatively, the DT model that forces $\xi = 0$ represents only the stationary dynamics of the video. In practice, we have found that the DT that includes the initial state $\xi$ performs better at segmentation and synthesis, since it is a better model for the particular observed video. The initial condition can also be specified with $x_0 \in \mathbb{R}^n$, as in [7], where $\xi = Ax_0$.

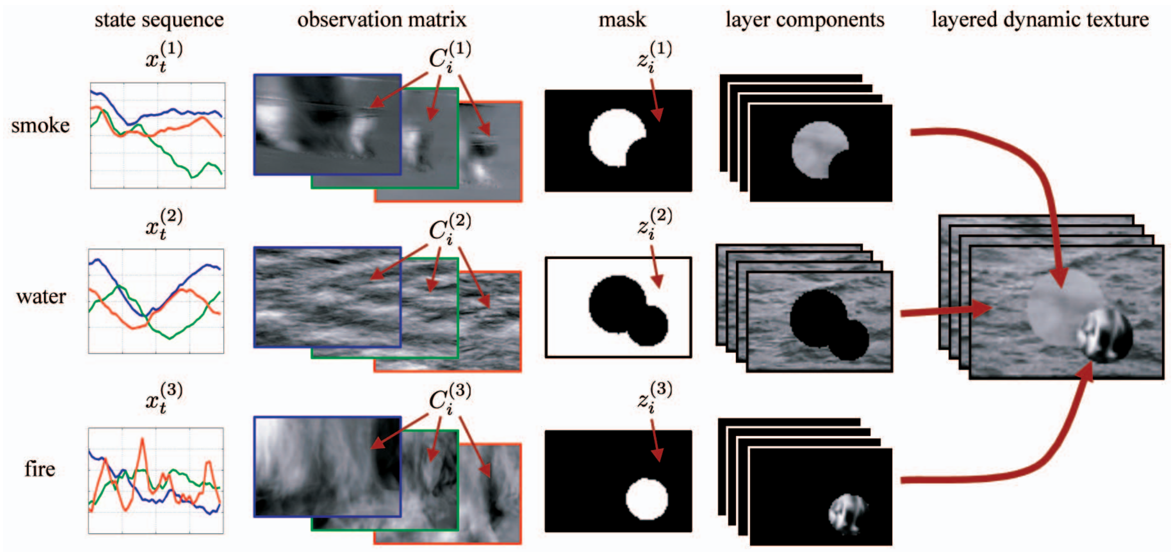

Fig. 2. Generative model for a video with multiple dynamic textures (smoke, water, and fire). The three textures are modeled with separate state sequences and observation matrices. The textures are then masked and composited to form the layered video.

Formally, the graphical model for the *layered dynamic texture* is shown in Fig. 1b. Each of the $K$ layers has a state process $x^{(j)} = \{x_t^{(j)}\}_{t=1}^{\tau}$ that evolves separately, where $\tau$ is the temporal length of the video. The video $Y = \{y_i\}_{i=1}^m$ contains $m$ pixels trajectories $y_i = \{y_{i,t}\}_{t=1}^{\tau}$, which are assigned to one of the layers through the hidden variable $z_i$. The collection of hidden variables $Z = \{z_i\}_{i=1}^m$ is modeled as an MRF to ensure spatial smoothness of the layer assignments (e.g., Fig. 1c). The model equations are

$$\begin{cases} x_t^{(j)} = A^{(j)} x_{t-1}^{(j)} + v_t^{(j)}, & j \in \{1, \cdots, K\}, \\ y_{i,t} = C_i^{(z_i)} x_t^{(z_i)} + w_{i,t} + \bar{y}_i^{(z_i)}, & i \in \{1, \cdots, m\}, \end{cases} \quad (2)$$

where $C_i^{(j)} \in \mathbb{R}^{1\times n}$ is the transformation from the hidden state to the observed pixel and $\bar{y}_i^{(j)} \in \mathbb{R}$ is the observation mean for *each* pixel $y_i$ and *each* layer $j$. The noise processes are $v_t^{(j)} \sim \mathcal{N}(0, Q^{(j)})$ and $w_{i,t} \sim \mathcal{N}(0, r^{(z_i)})$, and the initial state is given by $x_1^{(j)} \sim \mathcal{N}(\xi^{(j)}, Q^{(j)})$, where $Q^{(j)} \in \mathbb{S}_+^n$, $r^{(j)} \in \mathbb{R}_+$, and $\xi^{(j)} \in \mathbb{R}^n$. Given layer assignments, the LDT is a superposition of DTs defined over different regions of the video volume, and estimating the parameters of the LDT reduces to estimating those of the DT of each region. When layer assignments are unknown, the parameters can be estimated with the EM algorithm (see Section 4). We next derive the joint probability distribution of the LDT.

### 3.3 Joint Distribution of the LDT

As is typical for mixture models, we introduce an indicator variable $z_i^{(j)}$ of value 1 if and only if $z_i = j$, and 0 otherwise. The LDT model assumes that the state processes $X = \{x^{(j)}\}_{j=1}^K$ and the layer assignments $Z$ are independent, i.e., the layer dynamics are independent of its location. Under this assumption, the joint distribution factors are

$$p(X, Y, Z) = p(Y|X, Z)p(X)p(Z), \quad (3)$$

$$= \prod_{i=1}^m \prod_{j=1}^K p(y_i|x^{(j)}, z_i = j)^{z_i^{(j)}} \prod_{j=1}^K p(x^{(j)}) p(Z). \quad (4)$$

Each state sequence is a Gauss-Markov process, with distribution

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

where the individual state densities are

$$p(x_1^{(j)}) = G(x_1^{(j)}, \xi^{(j)}, Q^{(j)}), \quad (6)$$

$$p(x_t^{(j)}|x_{t-1}^{(j)}) = G(x_t^{(j)}, A^{(j)} x_{t-1}^{(j)}, Q^{(j)}), \quad (7)$$

and $G(x, \mu, \Sigma) = (2\pi)^{-n/2} |\Sigma|^{-1/2} e^{-\frac{1}{2}\|x-\mu\|_\Sigma^2}$ is an $n$-dimensional Gaussian distribution of mean $\mu$ and covariance $\Sigma$, and $\|a\|_\Sigma^2 = a^T \Sigma^{-1} a$. When conditioned on state sequences and layer assignments, pixel values are independent and pixel trajectories distributed as

$$p(y_i|x^{(j)}, z_i = j) = \prod_{t=1}^{\tau} p(y_{i,t}|x_t^{(j)}, z_i = j), \quad (8)$$

$$p(y_{i,t}|x_t^{(j)}, z_i = j) = G(y_{i,t}, C_i^{(j)} x_t^{(j)} + \bar{y}_i^{(j)}, r^{(j)}). \quad (9)$$

Finally, the layer assignments are jointly distributed as

$$p(Z) = \frac{1}{\mathcal{Z}_Z} \prod_{i=1}^m V_i(z_i) \prod_{(i,i')\in\mathcal{E}} V_{i,i'}(z_i, z_{i'}), \quad (10)$$

where $\mathcal{E}$ is the set of edges of the MRF, $\mathcal{Z}_Z$ a normalization constant (partition function), and $V_i$ and $V_{i,i'}$ are the potential functions of the form

$$V_i(z_i) = \prod_{j=1}^K \left(\alpha_i^{(j)}\right)^{z_i^{(j)}} = \begin{cases} \alpha_i^{(1)}, z_i = 1, \\ \quad\vdots \\ \alpha_i^{(K)}, z_i = K, \end{cases} \quad (11)$$

$$V_{i,i'}(z_i, z_{i'}) = \gamma_2 \prod_{j=1}^K \left(\frac{\gamma_1}{\gamma_2}\right)^{z_i^{(j)} z_{i'}^{(j)}} = \begin{cases} \gamma_1, z_i = z_{i'}, \\ \gamma_2, z_i \neq z_{i'}, \end{cases}$$

where $V_i$ is the prior probability of each layer, while $V_{i,i'}$ attributes higher probability to configurations with neighboring pixels in the same layer. In this work, we treat the MRF as a prior on $Z$, which controls the smoothness of the layers. The parameters of the potential functions of each layer could be learned, in a manner similar to [44], but we have so far found this to be unnecessary.

## 4 Parameter Estimation with the EM Algorithm

Given a training video $Y$, the parameters $\Theta = \{C_i^{(j)}, A^{(j)}, r^{(j)}, Q^{(j)}, \xi^{(j)}, \bar{y}_i^{(j)}\}_{j=1}^K$ of the LDT are learned by maximum-likelihood [45]

$$\Theta^* = \arg\max_{\Theta} \log p(Y) = \arg\max_{\Theta} \log \sum_{X,Z} p(Y,X,Z). \tag{12}$$

Since the data likelihood depends on hidden variables (state sequence $X$ and layer assignments $Z$), this problem can be solved with the EM algorithm [46], which iterates between

$$\mathrm{E-Step}: \mathcal{Q}(\Theta;\hat{\Theta}) = \mathbb{E}_{X,Z|Y;\hat{\Theta}}[\ell(X,Y,Z;\Theta)], \tag{13}$$

$$\mathrm{M-Step}: \hat{\Theta}' = \arg\max_{\Theta} \mathcal{Q}(\Theta;\hat{\Theta}), \tag{14}$$

where $\ell(X,Y,Z;\Theta) = \log p(X,Y,Z;\Theta)$ is the complete-data log-likelihood, parameterized by $\Theta$, and $\mathbb{E}_{X,Z|Y;\hat{\Theta}}$ the expectation with respect to $X$ and $Z$, conditioned on $Y$, parameterized by the current estimates $\hat{\Theta}$. We next derive the E and M steps for the LDT model.

### 4.1 Complete Data Log-Likelihood

Taking the logarithm of (4), the complete data log-likelihood is

$$\begin{aligned}\ell(X,Y,Z) = &\sum_{i=1}^m \sum_{j=1}^K z_i^{(j)} \sum_{t=1}^\tau \log p\big(y_{i,t} | x_t^{(j)}, z_i = j\big) \\ &+ \sum_{j=1}^K \left( \log p\big(x_1^{(j)}\big) + \sum_{t=2}^\tau \log p\big(x_t^{(j)} | x_{t-1}^{(j)}\big) \right) \\ &+ \log p(Z).\end{aligned} \tag{15}$$

Using (6), (7), and (9) and dropping terms that do not depend on the parameters $\Theta$ (and thus, play no role in the M-step)

$$\begin{aligned}\ell(X,Y,Z) = &-\frac{1}{2}\sum_{j=1}^K \sum_{i=1}^m z_i^{(j)} \sum_{t=1}^\tau \Big( \big\| y_{i,t} - \bar{y}_i^{(j)} - C_i^{(j)} x_t^{(j)} \big\|^2_{r^{(j)}} \\ &+ \log r^{(j)} \Big) - \frac{1}{2}\sum_{j=1}^K \Bigg( \big\| x_1^{(j)} - \xi^{(j)} \big\|^2_{Q^{(j)}} \\ &+ \sum_{t=2}^\tau \big\| x_t^{(j)} - A^{(j)} x_{t-1}^{(j)} \big\|^2_{Q^{(j)}} + \tau \log \big|Q^{(j)}\big| \Bigg),\end{aligned} \tag{16}$$

where $\|x\|^2_\Sigma = x^T \Sigma^{-1} x$. Note that $p(Z)$ can be ignored since the parameters of the MRF are constants. Finally, the complete data log-likelihood is

$$\begin{aligned}\ell(X,Y,Z) = &-\frac{1}{2}\sum_{j=1}^K \sum_{i=1}^m z_i^{(j)} \sum_{t=1}^\tau \frac{1}{r^{(j)}} \Big( \big(y_{i,t} - \bar{y}_i^{(j)}\big)^2 \\ &- 2\big(y_{i,t} - \bar{y}_i^{(j)}\big) C_i^{(j)} x_t^{(j)} + C_i^{(j)} P_{t,t}^{(j)} C_i^{(j)^T} \Big) \\ &- \frac{1}{2}\sum_{j=1}^K \mathrm{tr}\Big( Q^{(j)^{-1}} \Big( P_{1,1}^{(j)} - x_1^{(j)} \xi^{(j)^T} - \xi^{(j)} x_1^{(j)^T} \\ &+ \xi^{(j)} \xi^{(j)^T} \Big)\Big) - \frac{1}{2}\sum_{j=1}^K \sum_{t=2}^\tau \mathrm{tr}\Big[ Q^{(j)^{-1}} \Big( P_{t,t}^{(j)} \\ &- P_{t,t-1}^{(j)} A^{(j)^T} - A^{(j)} P_{t,t-1}^{(j)^T} + A^{(j)} P_{t-1,t-1}^{(j)} A^{(j)^T} \Big)\Big] \\ &- \frac{\tau}{2}\sum_{j=1}^K \sum_{i=1}^m z_i^{(j)} \log r^{(j)} - \frac{\tau}{2}\sum_{j=1}^K \log\big|Q^{(j)}\big|,\end{aligned} \tag{17}$$

where we define $P_{t,t}^{(j)} = x_t^{(j)} x_t^{(j)^T}$ and $P_{t,t-1}^{(j)} = x_t^{(j)} x_{t-1}^{(j)^T}$.

### 4.2 E-Step

From (17), it follows that the E-step of (13) requires conditional expectations of two forms

$$\begin{aligned}\mathbb{E}_{X,Z|Y}\big[f\big(x^{(j)}\big)\big] &= \mathbb{E}_{X|Y}\big[f\big(x^{(j)}\big)\big], \\ \mathbb{E}_{X,Z|Y}\big[z_i^{(j)} f\big(x^{(j)}\big)\big] &= \mathbb{E}_{Z|Y}\big[z_i^{(j)}\big] \mathbb{E}_{X|Y,z_i=j}\big[f\big(x^{(j)}\big)\big],\end{aligned} \tag{18}$$

for some function $f$ of $x^{(j)}$, and where $\mathbb{E}_{X|Y,z_i=j}$ is the conditional expectation of $X$ given the observation $Y$ and that the $i$th pixel belongs to layer $j$. In particular, the E-step requires

$$\begin{aligned}\hat{x}_t^{(j)} &= \mathbb{E}_{X|Y}\big[x_t^{(j)}\big], & \hat{P}_{t,t}^{(j)} &= \mathbb{E}_{X|Y}\big[P_{t,t}^{(j)}\big], \\ \hat{z}_i^{(j)} &= \mathbb{E}_{Z|Y}\big[z_i^{(j)}\big], & \hat{P}_{t,t-1}^{(j)} &= \mathbb{E}_{X|Y}\big[P_{t,t-1}^{(j)}\big], \\ \hat{x}_{t|i}^{(j)} &= \mathbb{E}_{X|Y,z_i=j}\big[x_t^{(j)}\big], & \hat{P}_{t,t|i}^{(j)} &= \mathbb{E}_{X|Y,z_i=j}\big[P_{t,t}^{(j)}\big].\end{aligned} \tag{19}$$

Defining, for convenience, the aggregate statistics

$$\begin{aligned}\phi_1^{(j)} &= \textstyle\sum_{t=1}^{\tau-1} \hat{P}_{t,t}^{(j)}, & \phi_2^{(j)} &= \textstyle\sum_{t=2}^{\tau} \hat{P}_{t,t}^{(j)}, \\ \psi^{(j)} &= \textstyle\sum_{t=2}^{\tau} \hat{P}_{t,t-1}^{(j)}, & \hat{N}_j &= \textstyle\sum_{i=1}^{m} \hat{z}_i^{(j)}, \\ \Phi_i^{(j)} &= \textstyle\sum_{t=1}^{\tau} \hat{P}_{t,t|i}^{(j)}, & \gamma^{(j)} &= \textstyle\sum_{t=1}^{\tau} \hat{x}_{t|i}^{(j)}, \\ \Gamma_i^{(j)} &= \textstyle\sum_{t=1}^{\tau} \big(y_{i,t} - \bar{y}_i^{(j)}\big) \hat{x}_{t|i}^{(j)},\end{aligned} \tag{20}$$

and substituting (20) and (17) into (13), leads to the $\mathcal{Q}$ function

$$\begin{aligned}\mathcal{Q}(\Theta;\hat{\Theta}) = &-\frac{1}{2}\sum_{j=1}^K \frac{1}{r^{(j)}} \sum_{i=1}^m \hat{z}_i^{(j)} \Bigg( \sum_{t=1}^\tau \big(y_{i,t} - \bar{y}_i^{(j)}\big)^2 \\ &- 2C_i^{(j)} \Gamma_i^{(j)} + C_i^{(j)} \Phi_i^{(j)} C_i^{(j)^T} \Bigg) - \frac{1}{2}\sum_{j=1}^K \mathrm{tr}\Big[ Q^{(j)^{-1}} \\ &\cdot \Big( \hat{P}_{1,1}^{(j)} - \hat{x}_1^{(j)} \xi^{(j)^T} - \xi^{(j)} \hat{x}_1^{(j)^T} + \xi^{(j)} \xi^{(j)^T} + \phi_2^{(j)} \\ &- \psi^{(j)} A^{(j)^T} - A^{(j)} \psi^{(j)^T} + A^{(j)} \phi_1^{(j)} A^{(j)^T} \Big)\Big] \\ &- \frac{\tau}{2}\sum_{j=1}^K \hat{N}_j \log r^{(j)} - \frac{\tau}{2}\sum_{j=1}^K \log\big|Q^{(j)}\big|.\end{aligned} \tag{21}$$

Since it is not known to which layer each pixel $y_i$ is assigned, the evaluation of the expectations of (19) requires marginalization over all configurations of $Z$. Hence, the $\mathcal{Q}$ function is intractable. Two possible approximations are discussed in Sections 5 and 6.

### 4.3 M-Step

The M-step of (14) updates the parameter estimates by maximizing the $\mathcal{Q}$ function. As usual, a (local) maximum is found by taking the partial derivative with respect to each parameter and setting it to zero (see Appendix A for complete derivation), yielding the estimates

$$\begin{aligned}
C_i^{(j)^*} &= \Gamma_i^{(j)^T} \Phi_i^{(j)^{-1}}, \\
A^{(j)^*} &= \psi^{(j)} \phi_1^{(j)^{-1}}, \\
\xi^{(j)*} &= \hat{x}_1^{(j)}, \\
\bar{y}_i^{(j)*} &= \frac{1}{\tau}\sum_{t=1}^{\tau} y_{i,t} - \frac{1}{\tau} C_i^{(j)} \gamma_i^{(j)}, \\
r^{(j)^*} &= \frac{1}{\tau \hat{N}_j} \sum_{i=1}^{m} \hat{z}_i^{(j)} \left( \sum_{t=1}^{\tau} \left(y_{i,t} - \bar{y}_i^{(j)}\right)^2 - C_i^{(j)^*} \Gamma_i^{(j)} \right), \\
Q^{(j)^*} &= \frac{1}{\tau} \left( \hat{P}_{1,1}^{(j)} - \xi^{(j)^*} \left(\xi^{(j)^*}\right)^T + \phi_2^{(j)} - A^{(j)^*} \psi^{(j)^T} \right).
\end{aligned} \tag{22}$$

The M-step of LDT learning is similar to that of LDS learning [41], [47], with two significant differences: 1) Each row $C_i^{(j)}$ of $C^{(j)}$ is estimated separately, conditioning all statistics on the assignment of pixel $i$ to layer $j$ ($z_i = j$) and 2) the estimate of the observation noise $r^{(j)}$ of each layer is a soft average of the unexplained variance of each pixel, weighted by the posterior probability $\hat{z}_i^{(j)}$ that pixel $i$ belongs to layer $j$.

### 4.4 Initialization Strategies

As is typical for the EM algorithm, the quality of the (locally) optimal solution depends on the initialization of the model parameters. In most cases, the approximate E-step also requires an initial estimate of the expected layer assignments $\hat{z}_i^{(j)}$. If an initial segmentation is available, both problems can be addressed easily: the model parameters can be initialized by learning a DT for each region, using [7], and the segmentation mask can be used as the initial $\hat{z}_i^{(j)}$. In our experience, a good initial segmentation can frequently be obtained with the DTM of [17]. Otherwise, when an initial segmentation is not available, we adopt a variation of the centroid splitting method of [48]. The EM algorithm is run with an increasing number of components. We start by learning an LDT with $K = 1$. A new layer is then added by duplicating the existing layer with the largest state-space noise (i.e., with the largest eigenvalue of $Q^{(j)}$). The new layer is perturbed by scaling the transition matrix $A$ by 0.99, and the resulting LDT used to initialize EM. For the approximate E-step, the initial $\hat{z}_i^{(j)}$ are estimated by approximating each pixel of the LDT with a DTM [17], where the parameters of the $j$th mixture component are identical to the parameters of the $j$th layer, $\{A^{(j)}, Q^{(j)}, \xi^{(j)}, C_i^{(j)}, r^{(j)}, \bar{y}_i^{(j)}\}$. The $\hat{z}_i^{(j)}$ estimate is the posterior probability that the pixel $y_i$ belongs to the jth mixture component, i.e., $\hat{z}_i^{(j)} \approx p(z_i = j|y_i)$. In successive E-steps, the estimates $\hat{z}_i^{(j)}$ from the previous E-step are used to initialize the current E-step. This produces an LDT with $K = 2$. The process is repeated with the successive introduction of new layers, by perturbation of existing ones, until the desired $K$ is reached. Note that perturbing the transition matrix coerces EM to learn layers with distinct dynamics.

## 5 Approximate Inference by Gibbs Sampling

The expectations of (19) require intractable conditional probabilities. For example, $P(X|Y) = \sum_Z P(X, Z|Y)$ requires the enumeration of all configurations of $Z$, an operation of exponential complexity on the MRF dimensions, and intractable for even moderate frame sizes. One commonly used solution to this problem is to rely on a Gibbs sampler [49] to draw samples from the posterior distribution $p(X, Z|Y)$ and approximate the desired expectations by sample averages. Given some initial state $\tilde{Z}$, each iteration of the Gibbs sampler for the LDT alternates between sampling $\tilde{X}$ from $p(X|Y, \tilde{Z})$ and sampling $\tilde{Z}$ from $p(Z|\tilde{X}, Y)$.

### 5.1 Sampling from $p(Z|X, Y)$

Using Bayes rule, the conditional distribution $p(Z|X, Y)$ can be rewritten as

$$p(Z|X,Y) = \frac{p(X,Y,Z)}{p(X,Y)} = \frac{p(Y|X,Z)p(X)p(Z)}{p(X,Y)} \tag{23}$$

$$\begin{aligned}
&\propto p(Y|X,Z)p(Z) \propto \prod_{i=1}^{m}\prod_{j=1}^{K} p\left(y_i|x^{(j)}, z_i = j\right)^{z_i^{(j)}} \\
&\quad \cdot \left[ \prod_{i=1}^{m} V_i(z_i) \prod_{(i,i')\in\mathcal{E}} V_{i,i'}(z_i, z_{i'}) \right] \\
&= \prod_{i=1}^{m}\prod_{j=1}^{K} \left[\alpha_i^{(j)} p\left(y_i|x^{(j)}, z_i = j\right)\right]^{z_i^{(j)}} \prod_{(i,i')\in\mathcal{E}} V_{i,i'}(z_i, z_{i'}).
\end{aligned} \tag{24}$$

Hence, $p(Z|X, Y)$ is equivalent to the MRF-likelihood function of (10), but with modified self-potentials $\tilde{V}_i(z_i) = \alpha^{(j)} p(y_i|x^{(j)}, z_i = j)$. Thus, samples from $p(Z|X, Y)$ can be drawn using Markov-chain Monte Carlo (MCMC) for an MRF grid [50].

### 5.2 Sampling from $p(X|Z, Y)$

Given layer assignments $Z$, pixels are deterministically assigned to state processes. For convenience, we define $\mathcal{I}_j = \{i|z_i = j\}$ as the index set for the pixels assigned to layer $j$, and $Y_j = \{y_i | i \in \mathcal{I}_j\}$ as the corresponding set of pixel values. Conditioning on $Z$, we have

$$p(X,Y|Z) = \prod_{j=1}^{K} p\left(x^{(j)}, Y_j|Z\right). \tag{25}$$

Note that $p(x^{(j)}, Y_j|Z)$ is the distribution of an LDS with parameters $\tilde{\Theta}_j = \{A^{(j)}, Q^{(j)}, \tilde{C}^{(j)}, r_j, \xi^{(j)}, \tilde{y}^{(j)}\}$, where $\tilde{C}^{(j)} = [C_i^{(j)}]_{i\in\mathcal{I}_j}$ is the subset of the rows of $C^{(j)}$ corresponding to the pixels $Y_j$, and likewise for $\tilde{y}^{(j)} = [\bar{y}_i^{(j)}]_{i\in\mathcal{I}_j}$. Marginalizing (25) with respect to $X$ yields

$$p(Y|Z) = \prod_j p(Y_j|Z), \tag{26}$$

where $p(Y_j|Z)$ is the likelihood of observing $Y_j$ from LDS $\tilde{\Theta}_j$. Finally, using Bayes rule,

$$p(X|Y,Z) = \frac{p(X,Y|Z)}{p(Y|Z)} = \frac{\prod_{j=1}^K p(x^{(j)}, Y_j|Z)}{\prod_{j=1}^K p(Y_j|Z)} \tag{27}$$

$$= \prod_{j=1}^K p(x^{(j)}|Y_j, Z). \tag{28}$$

Hence, sampling from $p(X|Y,Z)$ reduces to sampling a state-sequence $x^{(j)}$ from each $p(x^{(j)}|Y_j,Z)$, which is the conditional distribution of $x^{(j)}$, given the pixels $Y_j$, under the LDS parameterized by $\tilde{\Theta}_j$. An algorithm for efficiently drawing these sequences is given in Appendix B.

## 5.3 Approximate Inference

The Gibbs sampler is first "burned-in" by running it for 100 iterations. This allows the sample distribution for $\{\tilde{X}, \tilde{Z}\}$ to converge to the true posterior distribution $p(X,Z|Y)$. Subsequent samples, drawn after every five iterations of the Gibbs sampler, are used for approximate inference.

### *5.3.1 Approximate Expectations*

The expectations in (19) are approximated by averages over the samples drawn by the Gibbs sampler, e.g., $\mathbb{E}_{X|Y}[x_t^{(j)}] \approx \frac{1}{S}\sum_{s=1}^S [\tilde{x}_t^{(j)}]_s$, where $[\tilde{x}_t^{(j)}]_s$ is the value of $x_t^{(j)}$ in the $s$th sample, and $S$ is the number of samples.

### *5.3.2 Lower Bound on $p(Y)$*

The convergence of the EM algorithm is usually monitored by tracking the likelihood $p(Y)$ of the observed data. While this likelihood is intractable, a lower bound can be computed by summing over the configurations of $\tilde{Z}$ visited by the Gibbs sampler

$$p(Y) = \sum_Z p(Y|Z)p(Z) \geq \sum_{\tilde{Z} \in \mathcal{Z}_G} p(Y|\tilde{Z})p(\tilde{Z}), \tag{29}$$

where $\mathcal{Z}_G$ is the set of unique states of $\tilde{Z}$ visited by the sampler, $p(\tilde{Z})$ is given by (10), and $p(Y|\tilde{Z})$ is given by (26), where for each observation $Y_j$, the likelihood $p(Y_j|Z)$ is computed using the Kalman filter with parameters $\tilde{\Theta}_j$ [17], [41]. Because $\mathcal{Z}_G$ tend to be the configurations of the largest likelihood, the bound in (29) is a good approximation for convergence monitoring.

### *5.3.3 MAP Layer Assignment*

Finally, segmentation requires the MAP solution $\{X^*, Z^*\} = \operatorname{argmax}_{X,Z} p(X,Z|Y)$. This is computed with deterministic annealing, as in [50].

## 6 Inference by Variational Approximation

Using Gibbs sampling for approximate inference is frequently too computationally intensive. A popular low-complexity alternative is to rely on a variational approximation. This consists of approximating the posterior distribution $p(X,Z|Y)$ by an approximation $q(X,Z)$ within some class of tractable probability distributions $\mathcal{F}$. Given an observation $Y$, the optimal variational approximation minimizes the Kullback-Leibler (KL) divergence between the two posteriors [51]:

$$q^*(X,Z) = \arg\min_{q \in \mathcal{F}} \mathrm{KL}(q(X,Z) \| p(X,Z|Y)). \tag{30}$$

Note that, because the data log-likelihood $p(Y)$ is constant for an observed $Y$,

$$\begin{aligned} &\mathrm{KL}(q(X,Z)\|p(X,Z|Y)) \\ &\quad = \int q(X,Z) \log \frac{q(X,Z)}{p(X,Z|Y)} dXdZ \end{aligned} \tag{31}$$

$$= \int q(X,Z) \log \frac{q(X,Z)p(Y)}{p(X,Y,Z)} dXdZ \tag{32}$$

$$= \mathcal{L}(q(X,Z)) + \log p(Y), \tag{33}$$

where

$$\mathcal{L}(q(X,Z)) = \int q(X,Z) \log \frac{q(X,Z)}{p(X,Y,Z)} dXdZ. \tag{34}$$

The optimization problem of (30) is thus identical to

$$q^*(X,Z) = \arg\min_{q \in \mathcal{F}} \mathcal{L}(q(X,Z)). \tag{35}$$

We next derive an optimal approximate factorial posterior distribution.

## 6.1 Approximate Factorial Posterior Distribution

The intractability of the exact posterior distribution stems from the need to marginalize over $Z$. This suggests that a tractable approximate posterior can be obtained by assuming statistical independence between pixel assignments $z_i$ and state variables $x^{(j)}$, i.e.,

$$q(X,Z) = \prod_{j=1}^K q(x^{(j)}) \prod_{i=1}^m q(z_i). \tag{36}$$

Substituting into (34) leads to

$$\begin{aligned} \mathcal{L}(q(X,Z)) = \int &\prod_{j=1}^K q(x^{(j)}) \prod_{i=1}^m q(z_i) \\ &\cdot \log \frac{\prod_{j=1}^K q(x^{(j)}) \prod_{i=1}^m q(z_i)}{p(X,Y,Z)} dXdZ. \end{aligned} \tag{37}$$

Equation (37) is minimized by sequentially optimizing each of the factors $q(x^{(j)})$ and $q(z_i)$, while holding the others constant [51]. This yields the factorial distributions (see Appendix C for derivations)

$$\begin{aligned} \log q(x^{(j)}) = &\sum_{i=1}^m h_i^{(j)} \log p(y_i|x^{(j)}, z_i = j) \\ &+ \log p(x^{(j)}) - \log \mathcal{Z}_q^{(j)}, \end{aligned} \tag{38}$$

$$\log q(z_i) = \sum_{j=1}^K z_i^{(j)} \log h_i^{(j)}, \tag{39}$$

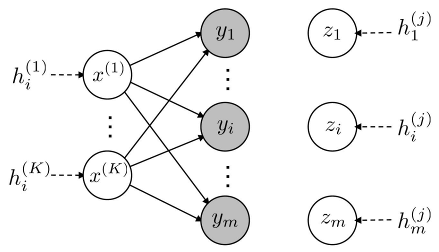

Fig. 3. Graphical model for the variational approximation of the layered dynamic texture. The influences of the variational parameters are indicated by the dashed arrows.

where $\mathcal{Z}_q^{(j)}$ is a normalization constant (Appendix C.3) and $h_i^{(j)}$ are the variational parameters

$$h_i^{(j)} = \mathbb{E}_{z_i}\left[z_i^{(j)}\right] = \frac{\alpha_i^{(j)} g_i^{(j)}}{\sum_{k=1}^K \alpha_i^{(k)} g_i^{(k)}}, \tag{40}$$

$$\begin{aligned}\log g_i^{(j)} &= \mathbb{E}_{x^{(j)}}\left[\log p(y_i|x^{(j)}, z_i = j)\right] \\ &\quad + \sum_{(i,i')\in\mathcal{E}} h_{i'}^{(j)} \log\frac{\gamma_1}{\gamma_2},\end{aligned} \tag{41}$$

and $\mathbb{E}_{x^{(j)}}$ and $\mathbb{E}_{z_i}$ are the expectations with respect to $q(x^{(j)})$ and $q(z_i)$.

The optimal factorial distributions can be interpreted as follows. The variational parameters $\{h_i^{(j)}\}$, which appear in both $q(z_i)$ and $q(x^{(j)})$, account for the dependence between $X$ and $Z$ (see Fig. 3). $h_i^{(j)}$ is the posterior probability of assigning pixel $y_i$ to layer $j$ and is estimated by the expected log-likelihood of assigning pixel $y_i$ to layer $j$, with an additional boost of $\log\frac{\gamma_1}{\gamma_2}$ per neighboring pixel also assigned to layer $j$. $h_i^{(j)}$ also weighs the contribution of each pixel $y_i$ to the factor $q(x^{(j)})$, which effectively acts as a soft assignment of pixel $y_i$ to layer $j$. Also note that, in (38), $h_i^{(j)}$ can be absorbed into $p(y_i|x^{(j)}, z_i = j)$, making $q(x^{(j)})$ the distribution of an LDS parameterized by $\hat{\Theta}_j = \{A^{(j)}, Q^{(j)}, C^{(j)}, R_j, \xi^{(j)}, \bar{y}^{(j)}\}$, where $R_j$ is a diagonal matrix with entries $[\frac{r^{(j)}}{h_1^{(j)}}, \ldots, \frac{r^{(j)}}{h_m^{(j)}}]$. Finally, $\log g_i^{(j)}$ is computed by rewriting (41) as

$$\begin{aligned}\log g_i^{(j)} &= \mathbb{E}_{x^{(j)}}\Bigg[\frac{-1}{2r^{(j)}}\sum_{t=1}^{\tau}\left\|y_{i,t} - \bar{y}_i^{(j)} - C_i^{(j)} x_t^{(j)}\right\|^2 \\ &\qquad - \frac{\tau}{2}\log 2\pi r^{(j)}\Bigg] + \sum_{(i,i')\in\mathcal{E}} h_{i'}^{(j)}\log\frac{\gamma_1}{\gamma_2}\end{aligned} \tag{42}$$

$$\begin{aligned}&= \frac{-1}{2r^{(j)}}\Bigg(\sum_{t=1}^{\tau}\left(y_{i,t} - \bar{y}_i^{(j)}\right)^2 - 2C_i^{(j)}\sum_{t=1}^{\tau}\left(y_{i,t} - \bar{y}_i^{(j)}\right) \\ &\quad \cdot \mathbb{E}_{x^{(j)}}\left[x_t^{(j)}\right] + C_i^{(j)}\sum_{t=1}^{\tau}\mathbb{E}_{x^{(j)}}\left[x_t^{(j)} x_t^{(j)T}\right] C_i^{(j)T}\Bigg) \\ &\quad - \frac{\tau}{2}\log 2\pi r^{(j)} + \sum_{(i,i')\in\mathcal{E}} h_{i'}^{(j)}\log\frac{\gamma_1}{\gamma_2},\end{aligned} \tag{43}$$

where the expectations $\mathbb{E}_{x^{(j)}}[x_t^{(j)}]$ and $\mathbb{E}_{x^{(j)}}[x_t^{(j)} x_t^{(j)T}]$ are computed with the Kalman smoothing filter [17], [41] for an LDS with parameters $\hat{\Theta}_j$.

The optimal $q^*(X, Z)$ is found by iterating through each pixel $i$, recomputing the variational parameters $h_i^{(j)}$ according to (40) and (41), until convergence. This might be computationally expensive because it requires running a Kalman smoothing filter for each pixel. The computational load can be reduced by updating batches of variational parameters at a time. In this work, we define a batch $\mathcal{B}$ as the set of nodes in the MRF with nonoverlapping Markov blankets (as in [52]), i.e., $\mathcal{B} = \{i | (i, i') \notin \mathcal{E}, \forall i' \in \mathcal{B}\}$. In practice, batch updating typically converges to the solution reached by serial updating, but is significantly faster. The variational approximation using batch (synchronous) updating is summarized in Algorithm 1.

**Algorithm 1.** Variational Approximation for LDT

```
 1: Input: LDT parameters $\Theta$, batches $\{\mathcal{B}_1, \ldots, \mathcal{B}_M\}$.
 2: Initialize $\{h_i^{(j)}\}$.
 3: repeat
 4:   {Recompute variational parameters for each batch}
 5:   for $\mathcal{B} \in \{\mathcal{B}_1, \ldots, \mathcal{B}_M\}$ do
 6:       compute $\mathbb{E}_{x^{(j)}}[x_t^{(j)}]$ and $\mathbb{E}_{x^{(j)}}[x_t^{(j)} x_t^{(j)T}]$ by running
          the Kalman smoothing filter with parameters $\hat{\Theta}_j$,
          for $j = \{1, \ldots, K\}$.
 7:       for $i \in \mathcal{B}$ do
 8:           compute $\log g_i^{(j)}$ using (43), for $j = \{1, \ldots, K\}$.
 9:           compute $h_i^{(j)}$ using (40), for $j = \{1, \ldots, K\}$.
10:       end for
11:   end for
12: until convergence of $h_i^{(j)}$
```

## 6.2 Approximate Inference

In the remainder of the section, we discuss inference with the approximate posterior $q^*(X, Z)$.

### 6.2.1 E-Step

In (19), expectations with respect to $p(X|Y)$ and $p(Z|Y)$ can be estimated as

$$\begin{aligned}\hat{x}_t^{(j)} &\approx \mathbb{E}_{x^{(j)}}\left[x_t^{(j)}\right], & \hat{P}_{t,t}^{(j)} &\approx \mathbb{E}_{x^{(j)}}\left[x_t^{(j)} x_t^{(j)T}\right], \\ \hat{z}_i^{(j)} &\approx h_i^{(j)}, & \hat{P}_{t,t-1}^{(j)} &\approx \mathbb{E}_{x^{(j)}}\left[x_t^{(j)} x_{t-1}^{(j)\,T}\right],\end{aligned} \tag{44}$$

where $\mathbb{E}_{x^{(j)}}$ is the expectation with respect to $q^*(x^{(j)})$. The remaining expectations of (19) are with respect to $p(X|Y, z_i = j)$, and can be approximated with $q^*(X|z_i = j)$ by running the variational algorithm with a binary $h_i^{(j)}$, set to enforce $z_i = j$. Note that if $m$ is large (as is the case with videos), fixing the value of a single $z_i = j$ will have little effect on the posterior, due to the combined evidence from the large number of other pixels in the layer. Hence, expectations with respect to $p(X|Y, z_i = j)$ can also be approximated with $q^*(X)$ when $m$ is large, i.e.,

$$\begin{aligned}\hat{x}_{t|i}^{(j)} &\approx \mathbb{E}_{x^{(j)}|z_i=j}\left[x_t^{(j)}\right] \approx \mathbb{E}_{x^{(j)}}\left[x_t^{(j)}\right], \\ \hat{P}_{t,t|i}^{(j)} &\approx \mathbb{E}_{x^{(j)}|z_i=j}\left[x_t^{(j)} x_t^{(j)T}\right] \approx \mathbb{E}_{x^{(j)}}\left[x_t^{(j)} x_t^{(j)T}\right],\end{aligned} \tag{45}$$

where $\mathbb{E}_{x^{(j)}|z_i=j}$ is the expectation with respect to $q^*(x^{(j)}|z_i = j)$. Finally, we note that the EM algorithm with variational E-step is guaranteed to converge. However, the approximate E-step prevents convergence to local maxima

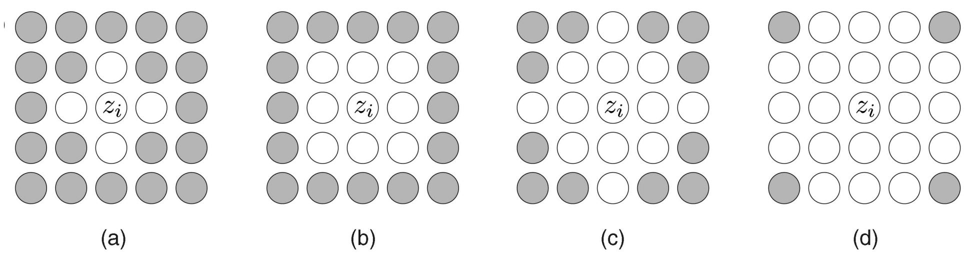

Fig. 4. MRF connectivity for node $z_i$. (a) First order (four neighbors). (b) Second order (eight neighbors). (c) Third order (12 neighbors). (d) Fourth order (20 neighbors). The nodes connected to $z_i$ are highlighted in white.

of the data log-likelihood [53]. Despite this limitation, the algorithm still performs well empirically, as shown in Section 7.

### 6.2.2 Lower Bound on $p(Y)$

Convergence is monitored with a lower bound on $p(Y)$, which follows from the nonnegativity of the KL divergence and (33)

$$\begin{aligned}\mathrm{KL}(q(X,Z)\|p(X,Z|Y)) &= \mathcal{L}(q(X,Z)) + \log p(Y) \geq 0 \\ &\Rightarrow \log p(Y) \geq -\mathcal{L}(q(X,Z)).\end{aligned} \tag{46}$$

Evaluating $\mathcal{L}$ for the optimal $q^*$ (see Appendix C.4 for derivation), the lower bound is

$$\begin{aligned}\log p(Y) \geq &\sum_j \log \mathcal{Z}_q^{(j)} - \sum_{j,i} h_i^{(j)} \log \frac{h_i^{(j)}}{\alpha_i^{(j)}} \\ &+ \sum_{(i,i')\in\mathcal{E}} \left( \log \gamma_2 + \sum_j h_i^{(j)} h_{i'}^{(j)} \log \frac{\gamma_1}{\gamma_2} \right) - \log \mathcal{Z}_Z.\end{aligned} \tag{47}$$

### 6.2.3 MAP Layer Assignment

Given the observed video $Y$, the *maximum* a posteriori layer assignment $Z$ (i.e., segmentation) is

$$Z^* = \arg\max_Z p(Z|Y) = \arg\max_Z \int p(X,Z|Y)dX \tag{48}$$

$$\approx \arg\max_Z \int q^*(X,Z)dX \tag{49}$$

$$= \arg\max_Z \int \prod_{j=1}^{K} q^*(x^{(j)}) \prod_{i=1}^{m} q^*(z_i)dX \tag{50}$$

$$= \arg\max_Z \prod_{i=1}^{m} q^*(z_i). \tag{51}$$

Hence, the MAP solution for $Z$ is approximated by the individual MAP solutions for $z_i$, i.e.,

$$z_i^* \approx \arg\max_j h_i^{(j)}, \ \forall i. \tag{52}$$

## 7 Experimental Evaluation

In this section, we present experiments that test the efficacy of the LDT model and the approximate inference algorithms. We start by comparing the two approximate inference algorithms on synthetic data, followed by an evaluation of EM learning with approximate inference. We conclude with experiments on segmentation of both synthetic and real videos. To reduce the memory and computation required to learn the LDT, we make a simplifying assumption in these experiments. We assume that $\bar{y}_i^{(j)}$ can be estimated by the empirical mean of the observed video, i.e., $\bar{y}_i^{(j)} \approx \frac{1}{\tau}\sum_{t=1}^{\tau} y_{i,t}$. This holds as long as $\tau$ is large and $A^{(j)}$ is stable,[2] which are reasonable assumptions for stationary video. Since the empirical mean is fixed for a given $Y$, we can effectively subtract the empirical mean from the video and set $\bar{y}_i^{(j)} = 0$ in the LDT. In practice, we have seen no difference in segmentation performance when using this simplified model.

## 7.1 Comparison of Approximate Inference Methods

We present a quantitative comparison of approximate inference on a synthetic data set, along with a comparison in the context of EM learning.

### 7.1.1 Synthetic Data Set

A synthetic data set of LDT samples was generated as follows. A number of LDTs of $K = 2$ components was produced by randomly sampling parameter values for each component $j = \{1, 2\}$, according to

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

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 $\Sigma$ and $d$ degrees of freedom, $\mathcal{U}_d(a,b)$ is a distribution on $\mathbb{R}^d$ vectors with each coordinate distributed uniformly between $a$ and $b$, and $\lambda_{max}(A_0^{(j)})$ is the magnitude of the largest eigenvalue of $A_0^{(j)}$. Note that $A^{(j)}$ is a random scaling of $A_0^{(j)}$ such that the system is stable (i.e., the poles of $A^{(j)}$ are within the unit circle). The MRF used first order connectivity (see Fig. 4a), with parameters $\log\gamma_1 = -\log\gamma_2 = 0.4$ and $\log\alpha_i^{(j)} = 0\ \forall i, j$.

2. Note that $\mathbb{E}[x_t] = A^{t-1}\xi$ and $\mathbb{E}[y_t] = CA^{t-1}\xi + \bar{y}$. Hence, the expected empirical mean is $\mathbb{E}[\frac{1}{\tau}\sum_{t=1}^{\tau} y_t] = C(\frac{1}{\tau}\sum_{t=1}^{\tau} A^{t-1})\xi + \bar{y}$. For large $\tau$ and stable $A$ (poles within the unit circle), $A^{t-1} \to 0$, and it follows that $\frac{1}{\tau}\sum_{t=1}^{\tau} A^{t-1} \to 0$. Hence, $\bar{y}_i^{(j)} \approx \mathbb{E}[\frac{1}{\tau}\sum_{t=1}^{\tau} y_{i,t}]$.

TABLE 1
Comparison of Approximate Inference Algorithms on Synthetic Data

| | std($\hat{z}_i$) | std($\hat{x}_t^{(j)}$) | mean($\hat{L}$) | $\hat{Z}$ Rand | time |
|---|---|---|---|---|---|
| Var | 0.062 | 0.327 | $-1.44 \times 10^5$ | 0.995 | 10.9s |
| Gibbs | 0.062 | 0.330 | $-1.44 \times 10^5$ | 0.995 | 456s |
| DTM | 0.158 | 3.883 | $-2.85 \times 10^5$ | 0.933 | 64.3s |

A set of 200 LDT parameters was sampled for all combinations of $n = \{10, 15, 20\}$ and $m = \{600, 1200\}$ (corresponding to a grid size of $30 \times 20$ and $40 \times 30$), and a time-series sample $\{X, Y, Z\}$, with temporal length 75, was drawn from each LDT, forming a synthetic data set of 1,200 time series. Finally, additional data sets, each with 1,200 time series, were formed by repeating with $K = \{3, 4\}$.

### 7.1.2 Inference Experiments

In this experiment, we compare the variational approximation (denoted as "Var") with Gibbs sampling (Gibbs). For Gibbs, expectations were approximated by averaging over 100 samples.[3] Each inference method was initialized with the DTM approximation for $\hat{z}_i^{(j)}$ discussed in Section 4.4. The conditional means of the hidden variables $\hat{z}_i = \mathbb{E}(z_i|Y)$ and $\hat{x}_t^{(j)} = \mathbb{E}(x_t^{(j)}|Y)$ were estimated and the standard deviations with respect to the ground-truth values of $z_i$ and $x_t^{(j)}$ were computed. The average value of the lower bound $\hat{L}$ of $\log P(Y)$ was also computed, along with the Rand index [54] between the true segmentation $Z$ and the approximate MAP solution $\hat{Z}$. The Rand index is a measure of clustering performance and intuitively is the probability of pairwise agreement between the clustering and the ground truth. Finally, the performance metrics were averaged over the synthetic data set for $K = 2$.

The estimation errors of the two approximate inference algorithms are presented in Table 1. Var and Gibbs have comparable performance, with the exception of a slight difference in the estimates of $x_t^{(j)}$. However, Var is significantly faster than Gibbs, with a speedup of over 40 times. Finally, although the estimation error of the DTM approximation is large for $\hat{x}_t^{(j)}$, the error of the layer assignments $\hat{z}_i$ is reasonable. This makes the DTM approximation a suitable initialization for the other inference algorithms.

### 7.1.3 EM Experiments

We next compare approximate inference in the context of the EM algorithm. LDT models were learned from the observed $Y$, using EM with the two approximate E-steps, which we denote as "VarEM" and "GibbsEM." The LDTs learned from the two EM algorithms were compared via their segmentation performance: the MAP solution $\hat{Z}$ was compared with the ground truth $Z$ using the Rand index. Finally, the Rand index was averaged over all LDTs in each synthetic data set $K = \{2, 3, 4\}$.

Fig. 5 presents the plots of Rand index versus the median runtime obtained for each method. VarEM and GibbsEM perform comparably (Rand of 0.998) for $K = 2$. However, GibbsEM outperforms VarEM when $K = \{3, 4\}$, with Rand

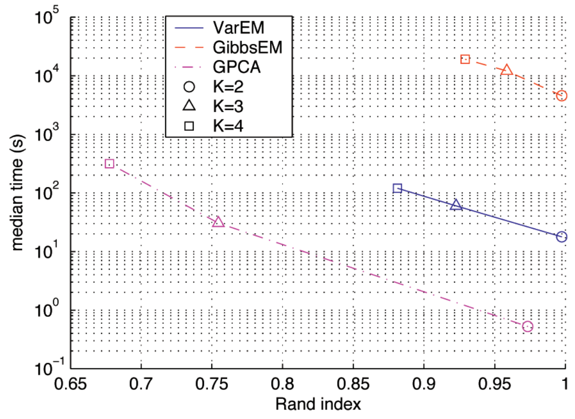

Fig. 5. Trade-off between runtime and segmentation performance using approximate inference.

0.959 and 0.929 versus 0.923 and 0.881, respectively. This difference is due to the unimodality of the approximate variational posterior; given multiple possible layer assignments (posterior modes), the variational approximation can only account for one of the configurations, effectively ignoring the other possibilities. While this behavior is acceptable when computing MAP assignments of a *learned* LDT (e.g., the inference experiments in Section 7.1.2), it may be detrimental for LDT *learning*. VarEM is not allowed to explore multiple configurations, which may lead to convergence to a poor local maximum. Poor performance of VarEM is more likely when there are multiple possible configurations, i.e., when $K$ is large (empirically, when $K \geq 3$). However, the improved performance of GibbsEM comes at a steep computational cost, with runtimes that are 150 to 250 times longer than those of VarEM. Finally, for comparison, the data were segmented with the GPCA method of [11], which is shown to perform worse than both VarEM and GibbsEM for all $K$. This is most likely due to the "noiseless" assumption of the underlying model, which makes the method susceptible to outliers, or other stochastic variations.

## 7.2 Motion Segmentation

In this section, we present results on motion segmentation using the LDT. All segmentations were obtained by learning an LDT with the EM algorithm and computing the posterior layer assignments $\hat{Z} = \operatorname{argmax}_Z p(Z|Y)$. The MRF parameters of the LDT were set to $\gamma_1 = -\gamma_2 = 5$ and $\alpha_i^{(j)} = 0, \forall i, j$, and the MRF used a first, second, or fourth order connectivity neighborhood (see Fig. 4), depending on the task. Unless otherwise noted, EM was initialized with the component splitting method of Section 4.4. Due to the significant computational cost of Gibbs sampling, we only report on the variational E-step. We also compare the LDT segmentations with those produced by various state-of-the-art methods in the literature: DTM with a patch size of $5 \times 5$ [17]; GPCA on the PCA projection of the pixel trajectories, as in [11]; level sets [12] on AR models (with order $n$) of Ising models (Ising), pixel intensities (AR), and mean-subtracted pixel intensities (AR0). Segmentations are evaluated by computing the Rand index [54] with the ground truth. We first present results on synthetic textures containing

3. Twenty five samples were drawn from four different runs of the Gibbs sampler.

4. Ising [12] could not be applied since there are more than two segments.

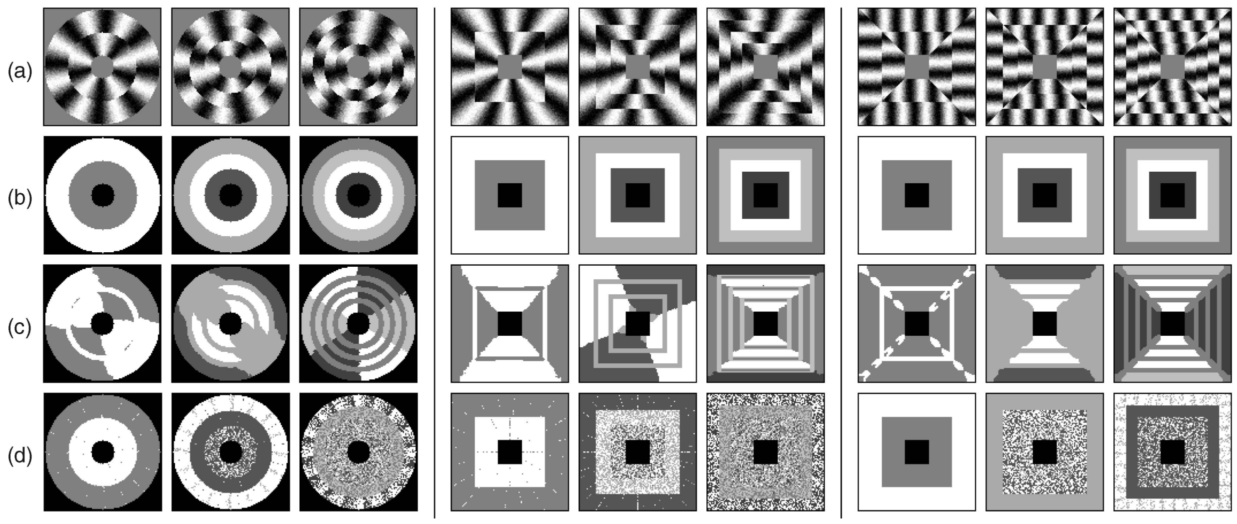

Fig. 6. Segmentation of synthetic circular motion: (a) video; segmentation using (b) LDT, (c) DTM [17], and (d) GPCA [11].

different types of circular motion. We then present a quantitative evaluation on a large texture database from [17], followed by results on real-world video. Video results are available online [55].

### 7.2.1 Synthetic Circular Motion

We first demonstrate LDT segmentation of sequences with motion that is locally varying but globally homogenous, e.g., a dynamic texture subject to circular motion. These experiments were based on videos containing several rings of distinct circular motion, as shown in Fig. 6a. Each video sequence $\mathcal{I}_{x,y,t}$ has dimensions $101 \times 101$, and was generated according to

$$\mathcal{I}_{x,y,t} = 128 \cos\left(c_r\theta + \frac{2\pi}{T_r}t + v_t\right) + 128 + w_t, \qquad (53)$$

where $\theta = \arctan(\frac{y-51}{x-51})$ is the angle of the pixel $(x, y)$ relative to the center of the video frame, $v_t \sim \mathcal{N}(0, (2\pi/50)^2)$ is the phase noise, and $w_t \sim \mathcal{N}(0, 10^2)$ is the observation noise. The parameter $T_r \in \{5, 10, 20, 40\}$ determines the speed of each ring, while $c_r$ determines the number of times the texture repeats around the ring. Here, we select $c_r$ such that all the ring textures have the same spatial period. Sequences were generated with $\{2, 3, 4\}$ circular or square rings, with a constant center patch (see Fig. 6 left and middle). Finally, a third set of dynamics was created by allowing the textures to move only horizontally or vertically (see Fig. 6 right).

The sequences were segmented with LDT (using an MRF with first order connectivity), DTM, and GPCA,[4] with $n = 2$ for all methods. The segmentation results are shown in Figs. 6b, 6c, and 6d. LDT (Fig. 6b) correctly segments all the rings, favoring global homogeneity over localized grouping of segments by texture orientation. On the other hand, DTM (Fig. 6c) tends to find incorrect segmentations based on local direction of motion. In addition, DTM sometimes incorrectly assigns one segment to the boundaries between rings, illustrating how the poor boundary accuracy of the patch-based segmentation framework can create substantial problems. Finally, GPCA (Fig. 6d) is able to correctly segment two rings, but fails when there are more. In these cases, GPCA correctly segments one of the rings, but randomly segments the remainder of the video. These results illustrate how LDT can correctly segment sequences whose motion is globally (at the ring level) homogeneous, but locally (at the patch level) heterogeneous. Both DTM and GPCA fail to exhibit this property. Quantitatively, this is reflected by the much higher average Rand scores of the segmentations produced by LDT (1.00, as compared to 0.482 and 0.826 for DTM and GPCA, respectively).

4. Ising [12] could not be applied since there are more than two segments.

### 7.2.2 Texture Database

We next present results on the texture database of [17], which contains 299 sequences with $K = \{2, 3, 4\}$ regions of different video textures (e.g., water, fire, and vegetation), as illustrated in Fig. 7a. In [17], the database was segmented with DTM, using a fixed initial contour. Although DTM was shown to be superior to other state-of-the-art methods [12], [11], the segmentations contain some errors due to the poor boundary localization discussed above. To test if using the LDT to refine the segmentations produced by DTM could substantially improve the results of [17], the LDT was initialized with the existing DTM segmentations, as described in Section 4.4. For comparison, we also applied the level-set methods of [12] (Ising, AR, and AR0), initialized with the DTM segmentations. The database was also segmented with GPCA [11], which does not require any initialization. Each method was run for several values of $n$ (where $n$ is the state-space dimension for LDT and DTM, and the AR model order for the level-set methods), and the average Rand index was computed for each $K$. In this experiment, the LDT used an MRF with the fourth order connectivity. Finally, the video was also segmented by clustering optical flow vectors [3] (GMM-OF) or motion profiles [56] (GMM-MP), averaged over time, with a Gaussian mixture model. No postprocessing was applied to the segmentations.

Table 2 shows the performance obtained, with the best $n$, by each algorithm. It is clear that LDT segmentation significantly improves the initial segmentation produced by DTM: the average Rand increases from 0.912 to 0.944, from 0.844 to 0.894, and from 0.857 to 0.916, for $K = \{2, 3, 4\}$, respectively. LDT also performs best among

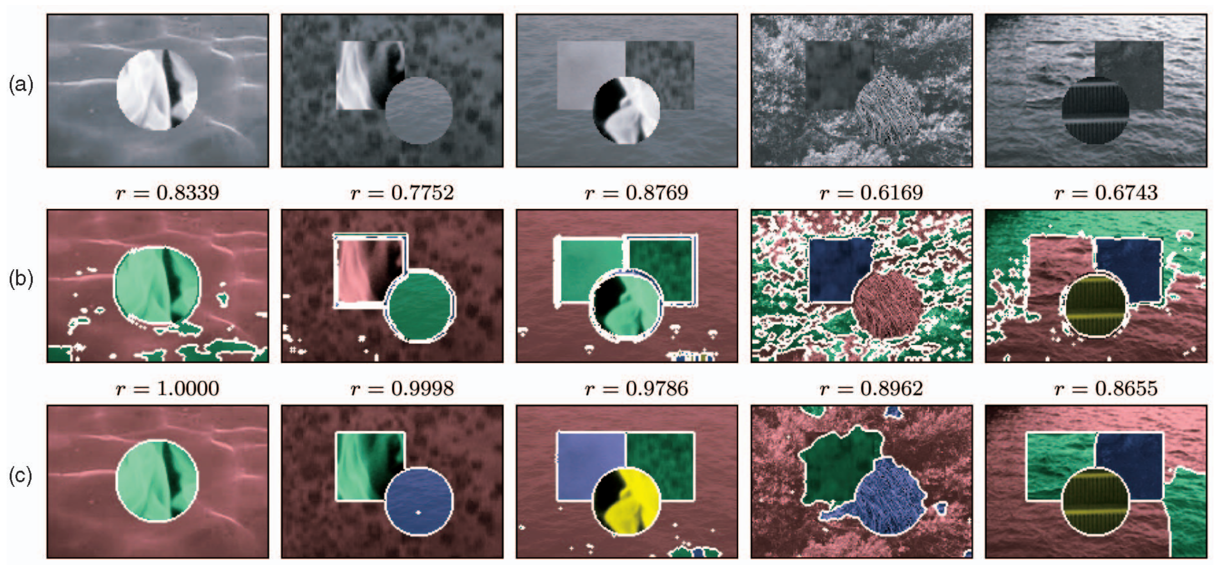

Fig. 7. Results on the texture database: (a) video; motion segmentations using (b) DTM [17], and (c) LDT. $r$ is the Rand index of the segmentation.

all algorithms, with Ising as the closest competitor (Rand 0.927). In addition, LDT and DTM both outperform the optical-flow-based methods (GMM-OF and GMM-MP), indicating that optical flow is not a suitable representation for video texture analysis. Fig. 8 shows a plot of the Rand index versus the dimension $n$ of the models, demonstrating that LDT segmentation is robust to the choice of $n$.

Qualitatively, LDT improves the DTM segmentation in three ways: 1) Segmentation boundaries are more precise, due to the region-level modeling (rather than patch level); 2) segmentations are less noisy, due to the inclusion of the MRF prior; and 3) gross errors, e.g., texture borders marked as segments, are eliminated. Several examples of these improvements are presented in Figs. 7b and 7c. From left to right, the first example is a case where the LDT corrects a noisy DTM segmentation (imprecise boundaries and spurious segments). The second and third examples are cases where the DTM produces a poor segmentation (e.g., the border between two textures erroneously marked as a segment), which the LDT corrects. The final two examples are very difficult cases. In the fourth example, the initial DTM segmentation is very poor. Albeit a substantial improvement, the LDT segmentation is still noisy. In the fifth example, the DTM splits the two water segments incorrectly (the two textures are very similar). The LDT substantially improves the segmentation, but the difficulties due to the great similarity of water patterns prove too difficult to overcome completely. More segmentation examples are available online [55].

Finally, we examine the LDT segmentation performance versus the connectivity of the MRF in Fig. 9. The average Rand increases with the order of MRF connectivity, due to the additional spatial constraints, but the gain saturates at fourth order.

### 7.2.3 *Real Video*

We next present segmentation experiments with real-video sequences. In all cases, the MRF used second order connectivity, and the state-space dimension $n$ was set to the value that produced the best segmentation for each sequence. Fig. 10a presents the segmentation of a moving ferris wheel, using LDT and DTM for $K = \{2, 3\}$. For $K = 2$, both LDT and DTM segment the static background from the moving ferris wheel. However, for $K = 3$ regions, the plausible segmentation by LDT of the foreground into two regions corresponding to the ferris wheel and a balloon moving in the wind is not matched by DTM. Instead, the latter segments the ferris wheel into two regions, according to the dominant direction of its local motion (either moving up or down), ignoring the balloon motion. This is identical to the problems found for the synthetic sequences of Fig. 6: the inability to uncover global homogeneity when the video is locally heterogeneous. On the other hand, the preference of LDT for two regions of very different sizes illustrates its robustness to this problem. The strong local heterogeneity of the optical flow in the region of the ferris wheel is well explained by the global homogeneity of the corresponding layer dynamics. Fig. 10b shows another example of this phenomenon. For $K = 3$ regions, LDT segments the windmill into regions corresponding to the moving fan blades, parts of the shaking tail piece, and the background. When segmenting into $K = 4$ regions, LDT splits the fan blade segment into two regions, which correspond to the fan blades and the internal support pieces. On the other hand, the DTM segmentations for $K = \{3, 4\}$ split the fan blades into different regions based on the orientation (vertical or horizontal) of the optical flow.

We next illustrate an interesting property of LDT segmentation with the proposed initialization: that it tends to produce a sequence of segmentations which captures a

TABLE 2
Average Rand Index for Various Segmentation Algorithms on the Texture Database (Value of $n$ in Parenthesis)

| Method | $K = 2$ | $K = 3$ | $K = 4$ |
|---|---|---|---|
| LDT | **0.944** (05) | **0.894** (12) | **0.916** (20) |
| DTM [17] | 0.912 (17) | 0.844 (15) | 0.857 (15) |
| Ising [12] | 0.927 (12) | n/a | n/a |
| AR [12] | 0.922 (10) | n/a | n/a |
| AR0 [12] | 0.917 (20) | n/a | n/a |
| GPCA [11] | 0.538 (02) | 0.518 (10) | 0.538 (10) |
| GMM-OF | 0.848 | 0.754 | 0.736 |
| GMM-MP | 0.847 | 0.792 | 0.802 |

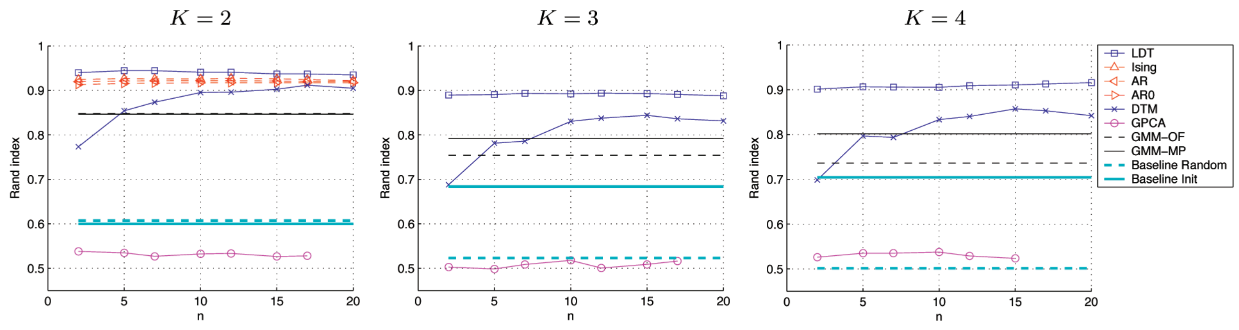

Fig. 8. Results on the texture database: Rand index versus $n$ for videos with $K = \{2, 3, 4\}$ segments.

hierarchy of scene dynamics. The whirlpool sequence of Fig. 11a contains different levels of moving and turbulent water. For $K = 2$ layers, the LDT segments the scene into regions containing moving water and still background (still water and grass). Adding another layer splits the "moving water" segment into two regions of different water dynamics: slowly moving ripples (outside of the whirlpool) and fast turbulent water (inside the whirlpool). Finally, for $K = 4$ layers, LDT splits the "turbulent water" region into two regions: the turbulent center of the whirlpool and the fast water spiraling into it. Fig. 11b shows the final segmentation, with the four layers corresponding to different levels of turbulence.

Finally, we present six other examples of LDT segmentation in Fig. 12. The first four are from the UCF database [57]. Figs. 12a, 12b, and 12c show segmentations of large pedestrian crowds. In Fig. 12a, a crowd moves in a circle around a pillar. The left side of the scene is less congested and the crowd moves faster than on the right side. In Fig. 12b, the crowd moves with three levels of speed, which are stratified into horizontal layers. In Fig. 12c, a crowd gathers at the entrance of an escalator, with people moving quickly around the edges. These segmentations show that LDT can distinguish different speeds of crowd motion, regardless of the direction in which the crowd is traveling. In Fig. 12d, the LDT segments a highway scene into still background, the fast moving traffic on the highway, and the slow traffic that merges into it. Another whirlpool is shown in Fig. 12e, where the turbulent water component is segmented from the remaining moving water. Finally, Fig. 12f presents a windmill scene from [58], which the LDT segments into regions corresponding to the windmill (circular motion), the trees waving in the wind, and the static background. These examples demonstrate the robustness of the LDT representation and its applicability to a wide range of scenes.

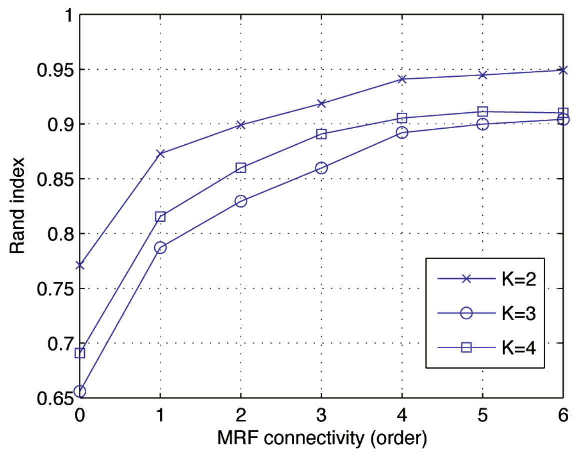

Fig. 9. Segmentation performance versus the MRF connectivity of the LDT.

## 8 Conclusions

In this work, we have introduced the layered dynamic texture, a generative model which represents a video as a layered collection of dynamic textures of different appearance and dynamics. We have also derived the EM algorithm for estimation of the maximum-likelihood model parameters from training video sequences. Because the posterior distribution of layer assignments given an observed video is computationally intractable, we have proposed two alternatives for inference with this model: a Gibbs sampler and an efficient variational approximation. The two approximate inference algorithms were compared experimentally, along with the corresponding approximate EM algorithms, on a synthetic data set. The two approximations were shown to produce comparable marginals (and MAP segmentations) when the LDT is given, but the Gibbs sampler outperformed the variational approximation in the context of EM-based model learning. However, this improvement comes with a very significant computational cost. This trade-off between computation and performance is usually observed when there is a need to rely on approximate inference with these two methods.

We have also conducted extensive experiments, with both mosaics of real textures and real-video sequences, that tested the ability of the proposed model (and algorithms) to segment videos into regions of coherent dynamics and appearance. The combination of LDT and variational inference has been shown to outperform a number of state-of-the-art methods for video segmentation. In particular, it was shown to possess a unique ability to group regions of *globally homogeneous but locally heterogeneous stochastic dynamics*. We believe that this ability is unmatched by any video segmentation algorithm currently available in the literature. The new method has also consistently produced segmentations with better spatial-localization than those possible with the *localized representations*, such as the DTM, that have previously been prevalent in the area of dynamic texture segmentation. Finally, we have demonstrated the robustness of the model, by segmenting real-video sequences depicting different classes of scenes: various types of crowds, highway traffic, and scenes containing a combination of globally homogeneous motion and highly stochastic motion (e.g., rotating windmills plus waving tree branches, or whirlpools).

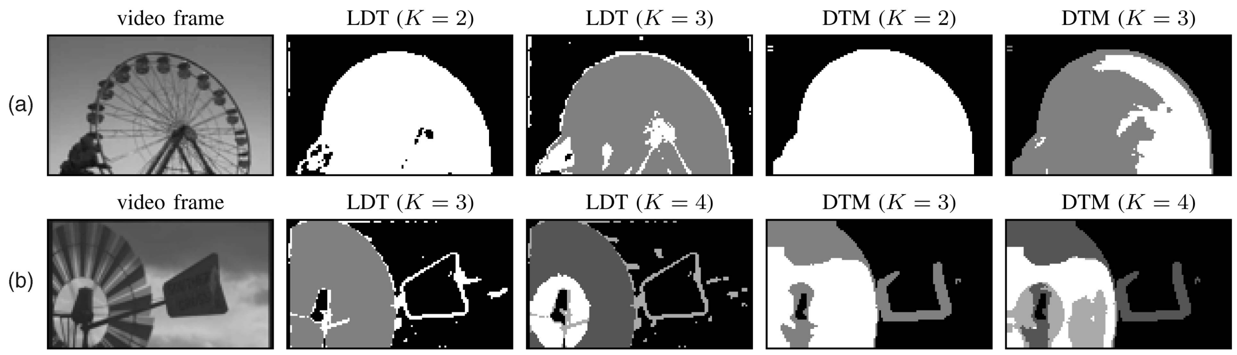

Fig. 10. Segmentation of (a) a ferris wheel and (b) a windmill, using LDT ($n = 2$ and $n = 10$) and DTM (both $n = 10$).

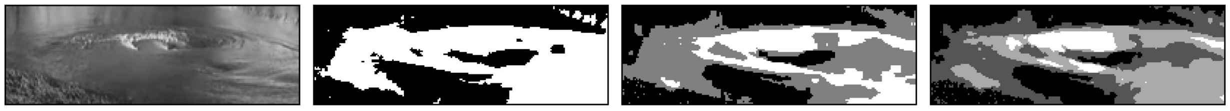

Fig. 11. Segmentation of a whirlpool using layered dynamic textures with $K = \{2, 3, 4\}$ and $n = 5$.

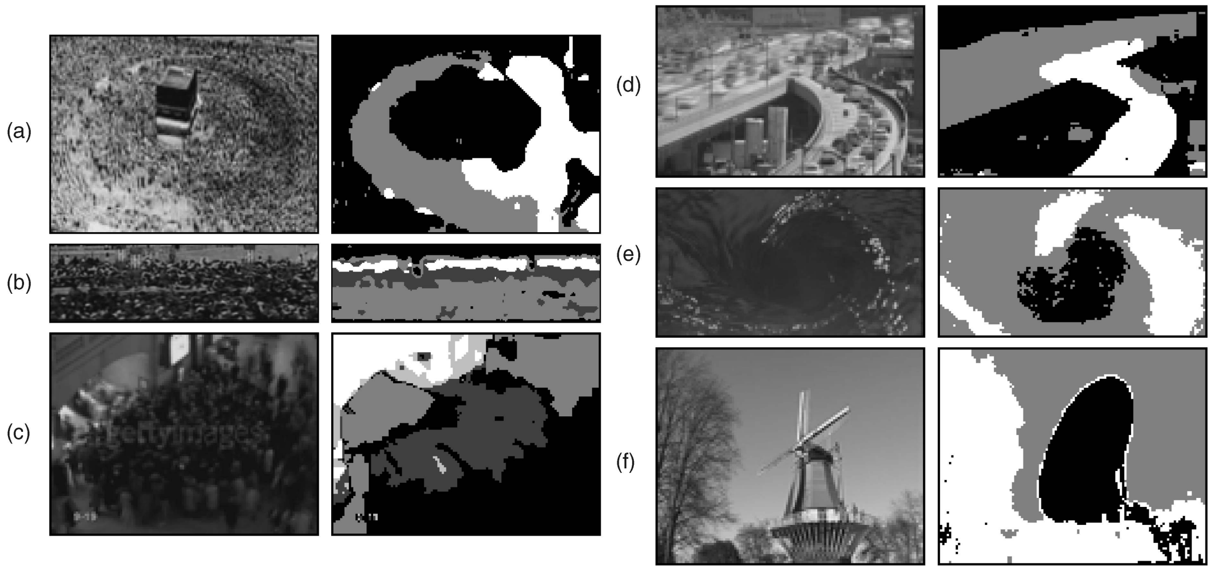

Fig. 12. Examples of motion segmentation using LDT. (a) Crowd moving around a pillar ($K = 3, n = 5$). (b) Crowd moving at different speeds ($K = 4, n = 15$). (c) Crowd around an escalator ($K = 5, n = 20$). (d) Highway on ramp ($K = 3, n = 10$). (e) Whirlpool ($K = 3, n = 10$). (f) Windmill and trees ($K = 4, n = 2$). The video is on the left and segmentation on the right.

## APPENDIX A

## DERIVATION OF THE M-STEP FOR LAYERED DYNAMIC TEXTURES

The maximization of the $\mathcal{Q}$ function with respect to the LDT parameters leads to two optimization problems. The first is a maximization with respect to a square matrix $X$ of the form

$$X^* = \arg\max_X -\frac{1}{2}\mathrm{tr}\left(X^{-1}A\right) - \frac{b}{2}\log|X|. \tag{54}$$

Taking derivatives and setting to zero yield

$$\frac{\partial}{\partial X}\frac{-1}{2}\mathrm{tr}\left(X^{-1}A\right) - \frac{b}{2}\log|X| = 0 \tag{55}$$

$$= \frac{1}{2}X^{-T}A^TX^{-T} - \frac{b}{2}X^{-T} \;\Rightarrow\; X^* = \frac{1}{b}A. \tag{56}$$

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

$$X^* = \arg\max_X -\frac{1}{2}\mathrm{tr}\left[D(-BX^T - XB^T + XCX^T)\right], \tag{57}$$

where $D$ and $C$ are the symmetric and invertible matrices. The solution is

$$\begin{aligned}\frac{\partial}{\partial X}\frac{-1}{2}\mathrm{tr}\left[D(-BX^T - XB^T + XCX^T)\right] &= 0\\ = -\frac{1}{2}(-DB - D^TB + D^TXC^T + DXC)\end{aligned} \tag{58}$$

$$= DB - DXC = 0 \quad \Rightarrow \quad X^* = BC^{-1}. \tag{59}$$

The optimal parameters are found by collecting the relevant terms in (21) and maximizing. This leads to a number of problems of the form of (21), namely,

$$A^{(j)*} = \arg\max_{A^{(j)}} -\frac{1}{2}\mathrm{tr}\Big[{Q^{(j)}}^{-1}\Big( -\psi^{(j)}{A^{(j)}}^T - A^{(j)}{\psi^{(j)}}^T + A^{(j)}\phi_1^{(j)}{A^{(j)}}^T\Big)\Big], \tag{60}$$

$$\xi^{(j)*} = \arg\max_{\xi^{(j)}} -\frac{1}{2}\mathrm{tr}\Big[{Q^{(j)}}^{-1}\Big(-\hat{x}_1^{(j)}{\xi^{(j)}}^T - \xi^{(j)}{\hat{x}_1^{(j)}}^T + \xi^{(j)}{\xi^{(j)}}^T\Big)\Big], \tag{61}$$

$$C_i^{(j)*} = \arg\max_{C_i^{(j)}} -\frac{1}{2}\frac{1}{r^{(j)}}\hat{z}_i^{(j)}\Big(-2C_i^{(j)}\Gamma_i^{(j)} + C_i^{(j)}\Phi_i^{(j)}{C_i^{(j)}}^T\Big), \tag{62}$$

$$\bar{y}_i^{(j)*} = \arg\max_{\bar{y}_i^{(j)}} -\frac{\hat{z}_i^{(j)}}{2r^{(j)}}\Bigg( -2\bar{y}_i^{(j)}\Bigg(\sum_{t=1}^{\tau} y_{i,t} - C_i^{(j)}\gamma_i^{(j)}\Bigg) + \tau(\bar{y}_i^{(j)})^2\Bigg). \tag{63}$$

Using (59) leads to the solutions of (14) in (22). The remaining problems are of the form of (54)

$$\begin{aligned} Q^{(j)*} = \arg\max_{Q^{(j)}} & -\frac{1}{2}\mathrm{tr}\Big[{Q^{(j)}}^{-1}\Big(\hat{P}_{1,1}^{(j)} - \hat{x}_1^{(j)}{\xi^{(j)}}^T \\ & - \xi^{(j)}{\hat{x}_1^{(j)}}^T + \xi^{(j)}{\xi^{(j)}}^T + \phi_2^{(j)} - \psi^{(j)}{A^{(j)}}^T \\ & - A^{(j)}{\psi^{(j)}}^T + A^{(j)}\phi_1^{(j)}{A^{(j)}}^T\Big)\Big] - \frac{\tau}{2}\log\big|Q^{(j)}\big|, \\ r^{(j)*} = \arg\max_{r^{(j)}} & \frac{-1}{2r^{(j)}}\sum_{i=1}^{m}\hat{z}_i^{(j)}\Bigg(\sum_{t=1}^{\tau}\big(y_{i,t} - \bar{y}_i^{(j)}\big)^2 \\ & - 2C_i^{(j)}\Gamma_i^{(j)} + C_i^{(j)}\Phi_i^{(j)}{C_i^{(j)}}^T\Bigg) - \frac{\tau}{2}\hat{N}_j\log r^{(j)}. \end{aligned}$$

In the first case, it follows from (56) that

$$Q^{(j)*} = \frac{1}{\tau}\Big(\hat{P}_{1,1}^{(j)} - \hat{x}_1^{(j)}{\xi^{(j)}}^T - \xi^{(j)}{\hat{x}_1^{(j)}}^T + \xi^{(j)}{\xi^{(j)}}^T + \phi_2^{(j)} - \psi^{(j)}{A^{(j)}}^T - A^{(j)}{\psi^{(j)}}^T + A^{(j)}\phi_1^{(j)}{A^{(j)}}^T\Big) \tag{64}$$

$$= \frac{1}{\tau}\Big(\hat{P}_{1,1}^{(j)} - \xi^{(j)*}{\xi^{(j)*}}^T + \phi_2^{(j)} - A^{(j)*}{\psi^{(j)}}^T\Big). \tag{65}$$

In the second case,

$$\begin{aligned} r^{(j)*} &= \frac{1}{\tau\hat{N}_j}\sum_{i=1}^{m}\hat{z}_i^{(j)}\Bigg(\sum_{t=1}^{\tau}\big(y_{i,t} - \bar{y}_i^{(j)}\big)^2 - 2C_i^{(j)}\Gamma_i^{(j)} + C_i^{(j)}\Phi_i^{(j)}{C_i^{(j)}}^T\Bigg) \\ &= \frac{1}{\tau\hat{N}_j}\sum_{i=1}^{m}\hat{z}_i^{(j)}\Bigg(\sum_{t=1}^{\tau}\big(y_{i,t} - \bar{y}_i^{(j)}\big)^2 - C_i^{(j)*}\Gamma_i^{(j)}\Bigg). \end{aligned}$$

## APPENDIX B

## SAMPLING A STATE SEQUENCE FROM AN LDS CONDITIONED ON THE OBSERVATION

In this appendix, we present an algorithm to efficiently sample a state sequence $x_{1:\tau} = \{x_1, \cdots, x_\tau\}$ from an LDS with parameters $\Theta = \{A, Q, C, R, \xi, \bar{y}\}$, conditioned on the observed sequence $y_{1:\tau} = \{y_1, \cdots, y_\tau\}$. The sampling algorithm first runs the Kalman filter [59] to compute state estimates conditioned on the current observations

$$\begin{aligned} \hat{x}_t^{t-1} &= \mathbb{E}(x_t|y_{1:t-1}), & \hat{V}_t^{t-1} &= \mathrm{cov}(x_t|y_{1:t-1}), \\ \hat{x}_t^{t} &= \mathbb{E}(x_t|y_{1:t}), & \hat{V}_t^{t} &= \mathrm{cov}(x_t|y_{1:t}), \end{aligned} \tag{66}$$

via the recursions

$$\begin{aligned} \hat{V}_t^{t-1} &= A\hat{V}_{t-1}^{t-1}A^T + Q, \\ K_t &= \hat{V}_t^{t-1}C^T\big(C\hat{V}_t^{t-1}C^T + R\big)^{-1}, \\ \hat{V}_t^t &= \hat{V}_t^{t-1} - K_tC\hat{V}_t^{t-1}, \\ \hat{x}_t^{t-1} &= A\hat{x}_{t-1}^{t-1}, \quad \hat{x}_t^t = \hat{x}_t^{t-1} + K_t\big(y_t - \bar{y} - C\hat{x}_t^{t-1}\big), \end{aligned} \tag{67}$$

where $t = 1, \dots, \tau$ and the initial conditions are $\hat{x}_1^0 = \xi$ and $\hat{V}_1^0 = Q$. From the Markovian structure of the LDS (Fig. 1a), $p(x_{1:\tau}|y_{1:\tau})$ can be factored in reverse order

$$p(x_{1:\tau}|y_{1:\tau}) = p(x_\tau|y_{1:\tau})\prod_{t=1}^{\tau-1}p(x_t|x_{t+1}, y_{1:\tau}) \tag{68}$$

$$= p(x_\tau|y_{1:\tau})\prod_{t=1}^{\tau-1}p(x_t|x_{t+1}, y_{1:t}), \tag{69}$$

where $p(x_\tau|y_{1:\tau})$ is a Gaussian with parameters already computed by the Kalman filter, $x_\tau \sim \mathcal{N}(\hat{x}_\tau^\tau, \hat{V}_\tau^\tau)$. The remaining distributions $p(x_t|x_{t+1}, y_{1:t})$ are Gaussian with mean and covariance given by the conditional Gaussian theorem [45]:

$$\begin{aligned} \mu_t &= \mathbb{E}[x_t|x_{t+1}, y_{1:t}] \\ &= \mathbb{E}[x_t|y_{1:t}] + \mathrm{cov}(x_t, x_{t+1}|y_{1:t})\mathrm{cov}(x_{t+1}|y_{1:t})^{-1} \\ &\quad \cdot (x_{t+1} - \mathbb{E}[x_{t+1}|y_{1:t}]) \\ &= \hat{x}_t^t + \hat{V}_t^tA^T\big(\hat{V}_{t+1}^t\big)^{-1}\big(x_{t+1} - \hat{x}_{t+1}^t\big), \end{aligned} \tag{70}$$

$$\begin{aligned} \Sigma_t &= \mathrm{cov}(x_t|x_{t+1}, y_{1:t}) \\ &= \mathrm{cov}(x_t|y_{1:t}) - \mathrm{cov}(x_t, x_{t+1}|y_{1:t})\mathrm{cov}(x_{t+1}|y_{1:t})^{-1} \\ &\quad \cdot \mathrm{cov}(x_{t+1}, x_t|y_{1:t}) \\ &= \hat{V}_t^t - \hat{V}_t^tA^T\big(\hat{V}_{t+1}^t\big)^{-1}A\hat{V}_t^t. \end{aligned} \tag{71}$$

where we have used $\mathrm{cov}(x_t, x_{t+1}|y_{1:t}) = \mathrm{cov}(x_t, Ax_t|y_{1:t}) = \hat{V}_t^tA^T$. A state sequence $\{x_1, \cdots, x_\tau\}$ can thus be sampled in reverse order, with $x_\tau \sim \mathcal{N}(\hat{x}_\tau^\tau, \hat{V}_\tau^\tau)$ and $x_t \sim \mathcal{N}(\mu_t, \Sigma_t)$ for $0 < t < \tau$.

## APPENDIX C

## DERIVATION OF THE VARIATIONAL APPROXIMATION FOR LDT

In this appendix, we derive a variational approximation for the LDT. The $\mathcal{L}$ function of (37) is minimized by sequentially optimizing each of the factors $q(x^{(j)})$ and $q(z_i)$, while holding the remaining constant [51]. For convenience, we define the variable $W = \{X, Z\}$. Rewriting (37) in terms of a single factor $q(w_l)$, while holding all others constant,

$$\mathcal{L}(q(W)) \propto \int q(w_l)\log q(w_l)dw_l - \int q(w_l)\int \prod_{k\neq l} q(w_k)\log p(W,Y)dW \tag{72}$$

$$= \int q(w_l)\log q(w_l)dw_l - \int q(w_l)\log \hat{p}(w_l,Y)dw_l = \mathrm{KL}(q(w_l)\|\hat{p}(w_l,Y)), \tag{73}$$

where in (72), we have dropped terms that do not depend on $q(w_l)$ (and hence, do not affect the optimization), and defined $\hat{p}(w_l,Y)$ as

$$\log \hat{p}(w_l,Y) \propto \mathbb{E}_{W_{k\neq l}}[\log p(W,Y)], \tag{74}$$

where

$$\mathbb{E}_{W_{k\neq l}}[\log p(W,Y)] = \int \prod_{k\neq l} q(w_k)\log p(W,Y)dW_{k\neq l}. \tag{75}$$

Since (73) is minimized when $q^*(w_l) = \hat{p}(w_l,Y)$, the optimal factor $q(w_l)$ is equal to the expectation of the joint log-likelihood with respect to the other factors $W_{k\neq l}$. We next derive the forms of the optimal factors $q(x^{(j)})$ and $q(z_i)$. For convenience, we ignore normalization constants during the derivation and reinstate them after the forms of the factors are known.

### C.1 Optimization of $q(x^{(j)})$

Rewriting (74) with $w_l = x^{(j)}$,

$$\begin{aligned}\log q^*\big(x^{(j)}\big) &\propto \log \hat{p}\big(x^{(j)},Y\big) = \mathbb{E}_{Z,X_{k\neq j}}[\log p(X,Y,Z)]\\ &\propto \mathbb{E}_{Z,X_{k\neq j}}\left[\sum_{i=1}^{m} z_i^{(j)}\log p\big(y_i|x^{(j)},z_i=j\big) + \log p\big(x^{(j)}\big)\right]\\ &= \sum_{i=1}^{m}\mathbb{E}_{z_i}\big[z_i^{(j)}\big]\log p\big(y_i|x^{(j)},z_i=j\big) + \log p\big(x^{(j)}\big),\end{aligned} \tag{76}$$

where we have dropped the terms of the complete data log-likelihood (15) that are not a function of $x^{(j)}$. Defining $h_i^{(j)} = \mathbb{E}_{z_i}[z_i^{(j)}] = \int q(z_i)z_i^{(j)}dz_i$, and the normalization term

$$\mathcal{Z}_q^{(j)} = \int p(x^{(j)})\prod_{i=1}^{m} p\big(y_i|x^{(j)},z_i=j\big)^{h_i^{(j)}}dx^{(j)}, \tag{77}$$

the optimal $q(x^{(j)})$ is given by (38).

### C.2 Optimization of $q(z_i)$

Rewriting (74) with $w_l = z_i$ and dropping terms that do not depend on $z_i$,

$$\log q^*(z_i) \propto \log \hat{p}(z_i,Y) = \mathbb{E}_{X,Z_{k\neq i}}[\log p(X,Y,Z)] \tag{78}$$

$$\propto \mathbb{E}_{X,Z_{k\neq i}}\left[\sum_{j=1}^{K} z_i^{(j)}\log p\big(y_i|x^{(j)},z_i=j\big) + \log\Big(V_i(z_i)\prod_{(i,i')\in\mathcal{E}} V_{i,i'}(z_i,z_{i'})\Big)\right] \tag{79}$$

$$= \sum_{j=1}^{K} z_i^{(j)}\mathbb{E}_{x^{(j)}}\big[\log p\big(y_i|x^{(j)},z_i=j\big)\big] + \sum_{(i,i')\in\mathcal{E}}\mathbb{E}_{z_{i'}}[\log V_{i,i'}(z_i,z_{i'})] + \log V_i(z_i). \tag{80}$$

Looking at the last two terms, we have

$$\sum_{(i,i')\in\mathcal{E}}\mathbb{E}_{z_{i'}}[\log V_{i,i'}(z_i,z_{i'})] + \log V_i(z_i) \tag{81}$$

$$= \sum_{(i,i')\in\mathcal{E}}\mathbb{E}_{z_{i'}}\left[\sum_{j=1}^{K} z_i^{(j)}z_{i'}^{(j)}\log\frac{\gamma_1}{\gamma_2} + \log\gamma_2\right] + \sum_{j=1}^{K} z_i^{(j)}\log\alpha_i^{(j)} \tag{82}$$

$$= \sum_{j=1}^{K} z_i^{(j)}\left(\sum_{(i,i')\in\mathcal{E}} h_{i'}^{(j)}\log\frac{\gamma_1}{\gamma_2} + \log\alpha_i^{(j)}\right). \tag{83}$$

Hence, $\log q^*(z_i) \propto \sum_{j=1}^{K} z_i^{(j)}\log(g_i^{(j)}\alpha_i^{(j)})$, where $g_i^{(j)}$ is defined in (41). This is a multinomial distribution of normalization constant $\sum_{j=1}^{K}(\alpha_i^{(j)}g_i^{(j)})$, leading to (39) with $h_i^{(j)}$ as given in (40).

### C.3 Normalization Constant for $q(x^{(j)})$

Taking the log of (77),

$$\log \mathcal{Z}_q^{(j)} = \log\int p(x^{(j)})\prod_{i=1}^{m}\prod_{t=1}^{\tau} p\big(y_{i,t}|x_t^{(j)},z_i=j\big)^{h_i^{(j)}}dx^{(j)}.$$

Note that the term $p(y_{i,t}|x_t^{(j)}z_i=j)^{h_i^{(j)}}$ does not affect the integral when $h_i^{(j)}=0$. Defining $\mathcal{I}_j$ as the set of indices with nonzero $h_i^{(j)}$, i.e., $\mathcal{I}_j = \{i|h_i^{(j)}>0\}$, we have

$$\log \mathcal{Z}_q^{(j)} = \log\int p(x^{(j)})\prod_{i\in\mathcal{I}_j}\prod_{t=1}^{\tau} p\big(y_{i,t}|x_t^{(j)},z_i=j\big)^{h_i^{(j)}}dx^{(j)}, \tag{84}$$

where

$$p\big(y_{i,t}|x_t^{(j)},z_i=j\big)^{h_i^{(j)}} = G\big(y_{i,t},C_i^{(j)}x_t^{(j)}+\bar{y}_i^{(j)},r^{(j)}\big)^{h_i^{(j)}} \tag{85}$$

$$= \big(2\pi r^{(j)}\big)^{-\frac{1}{2}h_i^{(j)}}\left(\frac{2\pi r^{(j)}}{h_i^{(j)}}\right)^{\frac{1}{2}} \cdot G\left(y_{i,t},C_i^{(j)}x_t^{(j)}+\bar{y}_i^{(j)},\frac{r^{(j)}}{h_i^{(j)}}\right). \tag{86}$$

For convenience, we define an LDS over the subset $\mathcal{I}_j$ parameterized by $\hat{\Theta}_j = \{A^{(j)},Q^{(j)},\hat{C}^{(j)},\hat{R}_j,\xi^{(j)},\hat{y}^{(j)}\}$, where $\hat{C}^{(j)} = [C_i^{(j)}]_{i\in\mathcal{I}_j}$, $\hat{y}^{(j)} = [\bar{y}_i^{(j)}]_{i\in\mathcal{I}_j}$ and $\hat{R}_j$ is a diagonal with entries $\hat{r}_i^{(j)} = \frac{r^{(j)}}{h_i^{(j)}}$ for $i\in\mathcal{I}_j$. Noting that this LDS has conditional observation likelihood

$$\hat{p}\big(y_{i,t}|x_t^{(j)},z_i=j\big) = G\big(y_{i,t},C_i^{(j)}x_t^{(j)}+\bar{y}_i^{(j)},\hat{r}_i^{(j)}\big), \tag{87}$$

we can rewrite

$$p\big(y_{i,t}|x_t^{(j)}, z_i = j\big)^{h_i^{(j)}} = \big(2\pi r^{(j)}\big)^{\frac{1}{2}(1-h_i^{(j)})}\big(h_i^{(j)}\big)^{-\frac{1}{2}} \cdot \hat{p}\big(y_{i,t}|x_t^{(j)}, z_i = j\big), \tag{88}$$

and from (84),

$$\log \mathcal{Z}_q^{(j)} = \log \int p\big(x^{(j)}\big) \prod_{i\in\mathcal{I}_j}\prod_{t=1}^{\tau} \Big[\big(2\pi r^{(j)}\big)^{\frac{1}{2}(1-h_i^{(j)})} \cdot \big(h_i^{(j)}\big)^{-\frac{1}{2}} \hat{p}\big(y_{i,t}|x_t^{(j)}, z_i = j\big)\Big] dx^{(j)}. \tag{89}$$

Under the LDS $\hat{\Theta}_j$, the likelihood of $Y_j = [y_i]_{i\in\mathcal{I}_j}$ is

$$\hat{p}_j(Y_j) = \int p\big(x^{(j)}\big) \prod_{i\in\mathcal{I}_j}\prod_{t=1}^{\tau} \hat{p}\big(y_{i,t}|x_t^{(j)}, z_i = j\big) dx^{(j)}, \tag{90}$$

and hence, it follows that

$$\log \mathcal{Z}_q^{(j)} = \frac{\tau}{2}\sum_{i\in\mathcal{I}_j}\big(1 - h_i^{(j)}\big)\log(2\pi r^{(j)}) - \frac{\tau}{2}\sum_{i\in\mathcal{I}_j}\log h_i^{(j)} + \log \hat{p}_j(Y_j). \tag{91}$$

### C.4 Lower Bound on $p(Y)$

To lower bound $p(Y)$ as in (46), we compute the $\mathcal{L}$ function of (34). We start with

$$\log \frac{q(X,Z)}{p(X,Y,Z)} = \log q(X,Z) - \log p(X,Y,Z) \tag{92}$$

$$= \Big[\sum_j \log q\big(x^{(j)}\big) + \sum_i \log q(z_i)\Big] - \Big[\sum_{j,i} z_i^{(j)} \cdot \log p\big(y_i|x^{(j)}, z_i = j\big) + \sum_j \log p\big(x^{(j)}\big) + \log p(Z)\Big]. \tag{93}$$

Substituting the optimal $q^*$ of (38) and (39),

$$\log \frac{q(X,Z)}{p(X,Y,Z)} = \Big[\sum_{j,i} h_i^{(j)} \log p\big(y_i|x^{(j)}, z_i = j\big) + \sum_j \log p\big(x^{(j)}\big) - \sum_j \log \mathcal{Z}_q^{(j)} + \sum_{j,i} z_i^{(j)} \log h_i^{(j)}\Big] - \Big[\sum_{j,i} z_i^{(j)} \cdot \log p\big(y_i|x^{(j)}, z_i = j\big) + \sum_j \log p\big(x^{(j)}\big) + \log p(Z)\Big] \tag{94}$$

$$= \sum_{j,i}\big(h_i^{(j)} - z_i^{(j)}\big)\log p\big(y_i|x^{(j)}, z_i = j\big) - \sum_j \log \mathcal{Z}_q^{(j)} + \sum_{j,i} z_i^{(j)} \log h_i^{(j)} - \log p(Z). \tag{95}$$

From (10),

$$\log p(Z) = \sum_{j,i} z_i^{(j)} \log \alpha_i^{(j)} + \sum_{(i,i')\in\mathcal{E}}\Big[\log \gamma_2 + \sum_j z_i^{(j)} z_{i'}^{(j)} \log \frac{\gamma_1}{\gamma_2}\Big] - \log \mathcal{Z}_Z.$$

Substituting this into (95) and taking the expectation with respect to $q^*(X,Z)$ yields the KL divergence:

$$\mathrm{KL}(q(X,Z)\|p(X,Y,Z)) = -\sum_j \log \mathcal{Z}_q^{(j)} + \sum_{j,i} h_i^{(j)} \cdot \log \frac{h_i^{(j)}}{\alpha_i^{(j)}} - \sum_{(i,i')\in\mathcal{E}}\Big[\log \gamma_2 + \sum_j h_i^{(j)} h_{i'}^{(j)} \log \frac{\gamma_1}{\gamma_2}\Big] + \log \mathcal{Z}_Z. \tag{96}$$

Substituting into (46) yields the log-likelihood lower bound (47).

## Acknowledgments

The authors thank Rene Vidal for the code from [11], [12], Mubarak Shah for the crowd videos [32], [57], Renaud Péteri, Mark Huiskes, and Sándor Fazekas for the windmill video [58], and the anonymous reviewers for insightful comments. This work was funded by a US National Science Foundation (NSF) award IIS-0534985 and an NSF IGERT award DGE-0333451.

## References

[1] B.K.P. Horn, *Robot Vision.* McGraw-Hill Book Company, 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," *Proc. DARPA Image Understanding Workshop,* pp. 121-130, 1981.
[4] J. Barron, D. Fleet, and S. Beauchemin, "Performance of Optical Flow Techniques," *Int'l J. Computer Vision,* vol. 12, pp. 43-77, 1994.
[5] J. Wang and E. Adelson, "Representing Moving Images with Layers," *IEEE Trans. Image Processing,* vol. 3, no. 5, pp. 625-638, Sept. 1994.
[6] B. Frey and N. Jojic, "Estimating Mixture Models of Images and Inferring Spatial Transformations Using the EM Algorithm," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 416-422, 1999.
[7] G. Doretto, A. Chiuso, Y.N. Wu, and S. Soatto, "Dynamic Textures," *Int'l J. Computer Vision,* vol. 51, no. 2, pp. 91-109, 2003.
[8] G. Doretto, D. Cremers, P. Favaro, and S. Soatto, "Dynamic Texture Segmentation," *Proc. Int'l Conf. Computer Vision,* vol. 2, pp. 1236-1242, 2003.
[9] P. Saisan, G. Doretto, Y. Wu, and S. Soatto, "Dynamic Texture Recognition," *Proc. IEEE. Conf. Computer Vision and Pattern Recognition,* vol. 2, pp. 58-63, 2001.
[10] A.B. Chan and N. Vasconcelos, "Probabilistic Kernels for the Classification of Auto-Regressive Visual Processes," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 1, pp. 846-851, 2005.
[11] R. Vidal and A. Ravichandran, "Optical Flow Estimation & Segmentation of Multiple Moving Dynamic Textures," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 2, pp. 516-521, 2005.
[12] A. Ghoreyshi and R. Vidal, "Segmenting Dynamic Textures with Ising Descriptors, ARX Models and Level Sets," *Proc. Dynamical Vision Workshop in the European Conf. Computer Vision,* 2006.
[13] A.B. Chan and N. Vasconcelos, "Layered Dynamic Textures," *Advances in Neural Information Processing Systems,* vol. 18, pp. 203-210, 2006.
[14] S. Soatto, G. Doretto, and Y.N. Wu, "Dynamic Textures," *Proc. IEEE Int'l Conf. Computer Vision,* pp. 439-446, 2001.
[15] L. Yuan, F. Wen, C. Liu, and H.-Y. Shum, "Synthesizing Dynamic Textures with Closed-Loop Linear Dynamic Systems," *Proc. European Conf. Computer Vision,* pp. 603-616, 2004.
[16] B. Ghanem and N. Ahuja, "Phase Based Modelling of Dynamic Textures," *Proc. IEEE Int'l Conf. Computer Vision,* 2007.

[17] A.B. Chan and N. Vasconcelos, "Modeling, Clustering, and Segmenting Video with Mixtures of Dynamic Textures," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 30, no. 5, pp. 909-926, May 2008.
[18] A.W. Fitzgibbon, "Stochastic Rigidity: Image Registration for Nowhere-Static Scenes," *Proc. Int'l Conf. Computer Vision,* vol. 1, pp. 662-670, 2001.
[19] S.V.N. Vishwanathan, A.J. Smola, and R. Vidal, "Binet-cauchy Kernels on Dynamical Systems and Its Application to the Analysis of Dynamic Scenes," *Int'l J. Computer Vision,* vol. 73, no. 1, pp. 95-119, 2007.
[20] A.B. Chan and N. Vasconcelos, "Classifying Video with Kernel Dynamic Textures," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2007.
[21] E. Cetingul, R. Chaudhry, and R. Vidal, "A System Theoretic Approach to Synthesis and Classification of Lip Articulation," *Proc. Int'l Workshop Dynamical Vision,* 2007.
[22] R. Vidal and P. Favaro, "Dynamicboost: Boosting Time Series Generated by Dynamical Systems," *Proc. Int'l Conf. Computer Vision,* 2007.
[23] S.M. Siddiqi, B. Boots, and G.J. Gordon, "A Constraint Generation Approach to Learning Stable Linear Dynamical Systems," *Advances in Neural Information Processing Systems,* 2007.
[24] R. Costantini, L. Sbaiz, and S. Süsstrunk, "Higher Order SVD Analysis for Dynamic Texture Synthesis," *IEEE Trans. Image Processing,* vol. 17, no. 1, pp. 42-52, Jan. 2008.
[25] M. Szummer and R. Picard, "Temporal Texture Modeling," *Proc. IEEE Conf. Image Processing,* vol. 3, pp. 823-826, 1996.
[26] G. Doretto, E. Jones, and S. Soatto, "Spatially Homogeneous Dynamic Textures," *Proc. European Conf. Computer Vision,* 2004.
[27] C.-B. Liu, R.-S. Lin, and N. Ahuja, "Modeling Dynamic Textures Using Subspace Mixtures," *Proc. Int'l Conf. Multimedia and Expo,* pp. 1378-1381, 2005.
[28] C.-B. Liu, R.-S. Lin, N. Ahuja, and M.-H. Yang, "Dynamic Texture Synthesis as Nonlinear Manifold Learning and Traversing," *Proc. British Machine Vision Conf.,* vol. 2, pp. 859-868, 2006.
[29] G. Doretto and S. Soatto, "Dynamic Shape and Appearance Models," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 28, no. 12, pp. 2006-2019, Dec. 2006.
[30] R. Vidal, "Online Clustering of Moving Hyperplanes," *Advances in Neural Information Processing Systems,* 2006.
[31] L. Cooper, J. Liu, and K. Huang, "Spatial Segmentation of Temporal Texture Using Mixture Linear Models," *Proc. Dynamical Vision Workshop in the IEEE Intl. Conf. Computer Vision,* 2005.
[32] S. Ali and M. Shah, "A Lagrangian Particle Dynamics Approach for Crowd Flow Segmentation and Stability Analysis," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2007.
[33] R. Shumway and D. Stoffer, "Dynamic Linear Models with Switching," *J. Am. Statistical Assoc.,* vol. 86, pp. 763-769, 1991.
[34] Y. Wu, G. Hua, and T. Yu, "Switching Observation Models for Contour Tracking in Clutter," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 295-302, 2003.
[35] M. Isard and A. Blake, "A Mixed-State Condensation Tracker with Automatic Model-Switching," *Proc. Int'l Conf. Computer Vision,* pp. 107-112, 1998.
[36] V. Pavlović, B.J. Frey, and T.S. Huang, "Time-Series Classification Using Mixed-State Dynamic Bayesian Networks," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 1999.
[37] V. Pavlović, J. Rehg, and J. MacCormick, "Learning Switching Linear Models of Human Motion," *Advances in Neural Information Processing Systems,* vol. 13, 2000.
[38] C.-J. Kim, "Dynamic Linear Models with Markov-Switching," *J. Econometrics,* vol. 60, pp. 1-22, 1994.
[39] S.M. Oh, J.M. Rehg, T. Balch, and F. Dellaert, "Learning and Inferring Motion Patterns Using Parametric Segmental Switching Linear Dynamic Systems," *Int'l J. Computer Vision,* special issue on learning for vision, vol. 77, nos. 1-3, pp. 103-124, 2008.
[40] Z. Ghahramani and G.E. Hinton, "Variational Learning for Switching State-Space Models," *Neural Computation,* vol. 12, no. 4, pp. 831-864, 2000.
[41] R.H. Shumway and D.S. Stoffer, "An Approach to Time Series Smoothing and Forecasting Using the EM Algorithm," *J. Time Series Analysis,* vol. 3, no. 4, pp. 253-264, 1982.
[42] 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.
[43] D. Bauer, "Comparing the CCA Subspace Method to Pseudo Maximum Likelihood Methods in the Case of No Exogenous Inputs," *J. Time Series Analysis,* vol. 26, pp. 631-668, 2005.
[44] N. Vasconcelos and A. Lippman, "Empirical Bayesian Motion Segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 23, no. 2, pp. 217-221, Feb. 2001.
[45] S.M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory.* Prentice-Hall, 1993.
[46] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *J. Royal Statistical Soc. B,* vol. 39, pp. 1-38, 1977.
[47] Z. Ghahramani and G. Hinton, "Parameter Estimation for Linear Dynamical Systems," Technical Report CRG-TR-96-2, Dept. of Computer Science, Univ. of Toronto, 1996.
[48] R.M. Gray, "Vector Quantization" *IEEE Trans. Acoustics, Speech, and Signal Processing Magazine,* vol. 1, no. 2, pp. 4-29, Apr. 1984.
[49] D.J.C. MacKay, "Introduction to Monte Carlo Methods," *Learning in Graphical Models,* pp. 175-204, MIT Press, 1999.
[50] S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distribution, and the Bayesian Restoration of Images," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 6, no. 6, pp. 721-741, Nov. 1984.
[51] C.M. Bishop, *Pattern Recognition and Machine Learning.* Springer, 2006.
[52] J. Besag, "Spatial Interaction and the Statistical Analysis of Lattice Systems," *J. Royal Statistical Soc., Series B (Methodological),* vol. 36, no. 2, pp. 192-236, 1974.
[53] A. Gunawardana and W. Byrne, "Convergence Theorems for Generalized Alternating Minimization Procedures," *J. Machine Learning Research,* vol. 6, pp. 2049-2073, 2005.
[54] L. Hubert and P. Arabie, "Comparing Partitions," *J. Classification,* vol. 2, pp. 193-218, 1985.
[55] "Layered Dynamic Textures," http://www.svcl.ucsd.edu/projects/layerdytex, 2009.
[56] J. Shi and J. Malik, "Motion Segmentation and Tracking Using Normalized Cuts," *Proc. IEEE Int'l Conf. Computer Vision,* pp. 1154-1160, 1999.
[57] "UCF Crowd Motion Database," http://www.cs.ucf.edu/~sali/Projects/CrowdSegmentation, 2009.
[58] "Dyntex: A Comprehensive Database of Dynamic Textures," http://www.cwi.nl/projects/dyntex, 2009.
[59] A. Gelb, *Applied Optimal Estimation.* MIT Press, 1974.

**Antoni B. Chan** received the BS and MEng degrees in electrical engineering from Cornell University in 2000 and 2001, respectively, and the PhD degree in electrical and computer engineering from the University of California, San Diego (UCSD), in 2008. He is a postdoctoral researcher in the Statistical Visual Computing Lab at UCSD. From 2001 to 2003, he was a visiting scientist in the Vision and Image Analysis Lab at Cornell University. From 2006 to 2008, he was the recipient of a US National Science Foundation (NSF) IGERT Fellowship. His research interests are in computer vision and machine learning. He is a member of the IEEE.

**Nuno Vasconcelos** received the Licenciatura degree 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 and a Hellman Fellowship. He has authored more than 75 peer-reviewed publications. His work spans various areas, including computer vision, machine learning, signal processing and compression, and multimedia systems. He is a senior member of the IEEE.