Clustering Dynamic Textures with the Hierarchical EM Algorithm for Modeling Video

Adeel Mumtaz, Emanuele Coviello, Gert. R. G. Lanckriet, Antoni B. Chan

Abstract—The dynamic texture (DT) is a probabilistic generative model, defined over space and time, that represents a video as the output of a linear dynamical system (LDS). The DT model has been applied to a wide variety of computer vision problems, such as motion segmentation, motion classification, and video registration. In this paper, we derive a new algorithm for clustering DT models that is based on the hierarchical EM algorithm. The proposed clustering algorithm is capable of both clustering DTs and learning *novel* DT cluster centers that are representative of the cluster members, in a manner that is consistent with the underlying generative probabilistic model of the DT. We also derive an efficient recursive algorithm for sensitivity analysis of the discrete-time Kalman smoothing filter, which is used as the basis for computing expectations in the E-step of the HEM algorithm. Finally, we demonstrate the efficacy of the clustering algorithm on several applications in motion analysis, including hierarchical motion clustering, semantic motion annotation, and learning bag-of-systems codebooks for dynamic texture recognition.

Index Terms—Dynamic Textures, Expectation Maximization, Kalman Filter, Bag of Systems, Video Annotation, Sensitivity Analysis.

1 Introduction

Modeling motion as a spatio-temporal texture has shown promise in a wide variety of computer vision problems, which have otherwise proven challenging for traditional motion representations, such as optical flow [1, 2]. In particular, the *dynamic texture model*, proposed in [3], has demonstrated a surprising ability to abstract a wide variety of complex global patterns of motion and appearance into a *simple* spatio-temporal model. The dynamic texture (DT) is a probabilistic generative model, defined over space and time, that represents a video (*i.e.*, spatio-temporal volume) as the output of a linear dynamical system (LDS). The model includes a hidden-state process, which encodes the motion of the video over time, and an observation variable that determines the appearance of each video frame, conditioned on the current hidden state. Both the hidden-state vector and the observation vector are representative of the entire image, enabling a holistic characterization of the motion for the entire sequence. The DT model has been applied to a wide variety of computer vision problems, including video texture synthesis [3], video registration [4, 5], motion and video texture segmentation [6, 7, 8, 9, 10], human activity recognition [11], human gait recognition [12], and motion classification [13, 14, 15, 16, 17, 18, 19, 20]. These successes illustrate both the modeling capabilities of the DT representation, and the robustness of the underlying probabilistic framework.

In this paper, we address the problem of clustering dynamic texture models, *i.e.*, clustering linear dynamical systems. Given a set of DTs (*e.g.*, each learned from a small video cube extracted from a large set of videos), the goal is to group similar DTs into K clusters, while also learning a representative DT "center" that can sufficiently summarize each group. This is analogous to standard K-means clustering, except that the datapoints are dynamic textures, instead of real vectors. A robust DT clustering algorithm has several potential applications in video analysis, including: 1) hierarchical clustering of motion; 2) video indexing for fast video retrieval; 3) DT codebook generation for the bag-of-systems motion representation; 4) semantic video annotation via weakly-supervised learning. Finally, DT clustering can also serve as an effective method for learning DTs from a large dataset of video via hierarchical estimation.

The parameters of the LDS lie on a non-Euclidean space (non-linear manifold), and hence cannot be clustered directly with the K-means algorithm, which operates on real vectors in Euclidean space. One solution, proposed in [18], first embeds the DTs into a Euclidean space using non-linear dimensionality reduction (NLDR), and then performs K-means on the low-dimensional space to obtain the clustering. While this performs the task of grouping the DTs into similar clusters, [18] is not able to generate *novel* DTs as cluster centers. These limitations could be addressed by clustering the DTs' parameters directly on the non-linear manifold, *e.g.*, using intrinsic mean-shift [21] or LLE [22]. However, these methods require analytic expressions for the log and exponential map on the manifold, which are difficult to compute for the DT parameters.

An alternative to clustering with respect to the manifold structure is to directly cluster the probability distributions of the DTs. One method for clustering probability distributions, in particular, Gaussians, is the hierarchical expectation-maximization (HEM) algorithm for Gaussian mixture models (GMMs), first proposed in [23]. The HEM algorithm of [23] takes a Gaussian mixture model (GMM) with K_b mixture components and reduces it to another GMM with K_r components ($K_r < K_b$), where each of the new Gaussian components represents a group of the original Gaussians (*i.e.*,

- *A. Mumtaz and A. B. Chan are with the Department of Computer Science, City University of Hong Kong.*
  E-mail: adeelmumtaz@gmail.com, abchan@cityu.edu.hk.
- *E. Coviello and G. R. G. Lanckriet are with the Department of Electrical and Computer Engineering, University of California, San Diego.*
  E-mail: emanuetre@gmail.com, gert@ece.ucsd.edu.

forming a cluster of Gaussians). HEM proceeds by generating *virtual* samples from each of the Gaussian components in the base GMM. Using these virtual samples, the reduced GMM is then estimated using the standard EM algorithm. The key insight of [23] is that, by applying the law of large numbers, a sum over virtual samples can be replaced by an expectation over the base Gaussian components, yielding a clustering algorithm that depends only on the parameters of the base GMM. The components of the reduced GMM are the Gaussian cluster centers, while the base components that contributed to these centers are the cluster members.

In this paper, we propose an HEM algorithm for *clustering dynamic textures* through their probability distributions [24]. The resulting algorithm is capable of both clustering DTs and learning *novel* DT cluster centers that are representative of the cluster members, in a manner that is consistent with the underlying generative probabilistic model of the DT. Besides clustering dynamic textures, the HEM algorithm can be used to efficiently learn a DT mixture from large datasets of video, using a hierarchical estimation procedure. In particular, intermediate DT mixtures are learned on small portions of the large dataset, and the final model is estimated by running HEM on the intermediate models. Because HEM is based on maximum-likelihood principles, it drives model estimation towards similar optimal parameter values as performing maximum-likelihood estimation on the full dataset.

we demonstrate the efficacy of the HEM clustering algorithm for DTs on several computer visions problems. First, we perform hierarchical clustering of video textures, showing that HEM groups perceptually similar motion together. Second, we use HEM to learn DT mixture models for semantic motion annotation, based on the supervised multi-class labeling (SML) framework [25]. DT annotation models are learned efficiently from weakly-labeled videos, by aggregating over large amounts of data using the HEM algorithm. Third, we generate codebooks with novel DT codewords for the bag-of-systems motion representation, and demonstrate improved performance on the task of dynamic texture recognition.

The contributions of this paper are three-fold. First, we propose and derive the HEM algorithm for clustering dynamic textures (linear dynamical systems). This involves extending the original HEM algorithm [23] to handle mixture components with hidden states (which are distinct from the hidden assignments of the overall mixture). Second, we derive an efficient recursive algorithm for calculating the E-step of this HEM algorithm, which makes a novel contribution to the subfield of "suboptimal filter analysis" or "sensitivity analysis" [26]. In particular, we derive expressions for the behavior (mean, covariance, and cross-covariance) of the Kalman smoothing filter when a mismatched source is applied. Third, we demonstrate the applicability of our HEM algorithm on a wide variety of tasks, including hierarchical DT clustering, DTM density estimation from large amounts of data, and estimating DT codebooks for BoS representations.

The remainder of this paper is organized as follows. Section 2 discusses related work, and we review dynamic texture models in Section 3. In Section 4, we derive the HEM algorithm for DT mixture models, and in Appendix A we derive an efficient algorithm for sensitivity analysis of the Kalman smoothing filter. Finally, Section 5 concludes the paper by presenting three applications of HEM with experimental evaluations.

2 Related Work

[18] proposes to cluster DT models using non-linear dimensionality reduction (NLDR). First, the DTs are embedded into a Euclidean space using multidimensional scaling (MDS) and the Martin distance function. Next, the DTs are grouped together by applying K-means clustering on the low-dimensional embedded points. Generating representative DTs corresponding to the K-means cluster centers is challenging, due to the pre-image and out-of-sample limitations of kernelized NLDR techniques. [18] works around this problem by selecting the DT whose low-dimensional embedding is closest to the low-dimensional cluster center as the representative DT for the cluster.

The HEM algorithm for GMMs, proposed in [23], has been employed in [27] to build GMM hierarchies for efficient image indexing, and in [25] to estimate GMMs from large image datasets for semantic annotation. In this paper, we extend the HEM algorithm to dynamic texture mixtures (DTMs), where each mixture component is an LDS. In contrast to GMMs, the E-step inference of HEM for DTMs requires a substantial derivation to obtain an efficient algorithm, due to the hidden state variables of the LDS.

Other approaches to clustering probability distributions have also been proposed in the literature. [28] introduces a generic clustering algorithm based on Bregman divergences. Setting the Bregman divergence to the discrete KL divergence yields an algorithm for clustering multinomials. When the Bregman divergence is the sum of the Mahalonobis distance and the Burg matrix divergence, the result is a clustering algorithm for multivariate Gaussians [29], which uses the covariance and means of the base Gaussians. Similarly, [30] minimizes the weighted sum of the Kullback-Leibler (KL) divergence between the cluster center and each probability distribution, yielding an alternating minimization procedure identical to [29]. While this approach could also be applied to clustering dynamic textures, it would require calculating prohibitively large (if not infinite) covariance matrices.

Previous works on sensitivity analysis [31, 32] focus on the actual covariance matrix of the error, i.e., the covariance of the error between the state estimate of the Kalman smoothing filter and the true state when a mismatched source LDS is applied. In contrast, the HEM E-step requires the expectation, covariance, and cross-covariance of the smoothed state estimator under a different LDS, i.e., the actual expected behavior of the Kalman smoothing filter when a mismatched source is applied. Some of these quantities are related to the actual error covariance matrix, and some are not. Hence, the results from [31, 32] cannot be directly used to obtain our HEM E-step, or vice versa.

With respect to our previous work, the HEM algorithm for DTMs was originally proposed in [24]. In contrast to [24], this paper presents a more complete analysis of HEM-DTM and significantly more experimental results: 1) a complete derivation of the HEM algorithm for DT mixtures; 2) a complete

derivation of the sensitivity analysis of the Kalman smoothing filter, used for the E-step in HEM-DTM; 3) a new experiment on semantic motion annotation using a video dataset of real scenes; 4) new experiments on dynamic texture recognition with the bag-of-systems representation using different datasets, as well as comparisons with other state-of-the-art methods. Finally, we have also applied HEM-DTM to music annotation in [33], which mainly focuses on large-scale experiments and interpreting the parameters of the learned DT annotation models.

3 Dynamic Texture Models

A dynamic texture [3] (DT) is a generative model for both the appearance and the dynamics of video sequences. The model consists of a random process containing an *observation variable* y_t, which encodes the appearance component (vectorized video frame at time t), and a *hidden state variable* x_t, which encodes the dynamics (evolution of the video over time). The appearance component is drawn at each time instant, conditionally on the current hidden state. The state and observation variables are related through the *linear dynamical system* (LDS) defined by

$$x_t = Ax_{t-1} + v_t, \tag{1}$$

$$y_t = Cx_t + w_t + \bar{y}, \tag{2}$$

where $x_t \in \mathbb{R}^n$ and $y_t \in \mathbb{R}^m$ are real vectors (typically $n \ll m$). The matrix $A \in \mathbb{R}^{n\times n}$ is a *state transition matrix*, which encodes the dynamics or evolution of the hidden state variable (*i.e.*, the motion of the video), and the matrix $C \in \mathbb{R}^{m\times n}$ is an *observation matrix*, which encodes the appearance component of the video sequence. The vector $\bar{y} \in \mathbb{R}^n$ is the mean of the dynamic texture (*i.e.*, the mean video frame). v_t is a *driving noise process*, and is zero-mean Gaussian distributed, *i.e.*, $v_t \sim \mathcal{N}(0, Q)$, where $Q \in \mathbb{R}^{n\times n}$ is a covariance matrix. w_t is the *observation noise* and is also zero-mean Gaussian, *i.e.*, $w_t \sim \mathcal{N}(0, R)$, where $R \in \mathbb{R}^{m\times m}$ is a covariance matrix (typically, it is assumed the observation noise is i.i.d. between the pixels, and hence $R = rI_m$ is a scaled identity matrix). Finally, the *initial condition* is specified as $x_1 \sim \mathcal{N}(\mu, S)$, where $\mu \in \mathbb{R}^n$ is the mean of the initial state, and $S \in \mathbb{R}^{n\times n}$ is the covariance. The dynamic texture is specified by the parameters $\Theta = \{A, Q, C, R, \mu, S, \bar{y}\}$. A number of methods are available to learn the parameters of the dynamic texture from a training video sequence, including maximum-likelihood (*e.g.*, expectation-maximization [34]), or a suboptimal, but computationally efficient, greedy least-squares procedure [3].

While a dynamic texture models a time-series as a single sample from a linear dynamical system, the dynamic texture mixture (DTM), proposed in [8], models multiple time-series as samples from a set of K dynamic textures. The DTM model introduces an assignment random variable $z \sim \text{multinomial}(\pi_1, \cdots, \pi_K)$, which selects the parameters of one of the K dynamic texture components for generating a video observation, resulting in system equations

$$\begin{cases} x_t = A_z x_{t-1} + v_t \\ y_t = C_z x_t + w_t + \bar{y}_z, \end{cases} \tag{3}$$

where each mixture component is parameterized by $\Theta_z = \{A_z, Q_z, C_z, R_z, \mu_z, S_z, \bar{y}_z\}$, and the DTM model is parameterized by $\Theta = \{\pi_z, \Theta_z\}_{z=1}^K$. Given a set of video samples, the maximum-likelihood parameters of the DTM can be estimated with recourse to the expectation-maximization (EM) algorithm [8]. The EM algorithm for DTM alternates between estimating first and second-order statistics of the hidden states, conditioned on each video, with the Kalman smoothing filter (E-step), and computing new parameters given these statistics (M-step).

4 The HEM Algorithm for Dynamic Textures

The hierarchical expectation-maximization (HEM) algorithm was proposed in [23] to reduce a Gaussian mixture model (GMM) with a large number of components into a representative GMM with fewer components. In this section we derive the HEM algorithm when the mixture components are *dynamic textures*.

4.1 Formulation

Let $\Theta^{(b)} = \{\pi_i^{(b)}, \Theta_i^{(b)}\}_{i=1}^{K^{(b)}}$ denote the base DT mixture model with $K^{(b)}$ components. The likelihood of the observed random variable $y_{1:\tau} \sim \Theta^{(b)}$ is given by

$$p(y_{1:\tau}|\Theta^{(b)}) = \sum_{i=1}^{K^{(b)}} \pi_i^{(b)} p(y_{1:\tau}|z^{(b)} = i, \Theta^{(b)}), \tag{4}$$

where $y_{1:\tau}$ is the video, τ is the video length, and $z \sim \text{multinomial}(\pi_1^{(b)}, \cdots \pi_{K^{(b)}}^{(b)})$ is the hidden variable that indexes the mixture components. $p(y_{1:\tau}|z^{(b)} = i, \Theta^{(b)})$ is the likelihood of the video $y_{1:\tau}$ under the ith DT mixture component, and $\pi_i^{(b)}$ is the prior weight for the ith component. The goal is to find a reduced DT mixture model, $\Theta^{(r)}$, which represents (4) using fewer mixture components. The likelihood of the observed video random variable $y_{1:\tau} \sim \Theta^{(r)}$ is

$$p(y_{1:\tau}|\Theta^{(r)}) = \sum_{j=1}^{K^{(r)}} \pi_j^{(r)} p(y_{1:\tau}|z^{(r)} = j, \Theta^{(r)}), \tag{5}$$

where $K^{(r)}$ is the number of DT components in the reduced model ($K^{(r)} < K^{(b)}$), and $z^{(r)} \sim \text{multinomial}(\pi_1^{(r)}, \cdots, \pi_{K^{(r)}}^{(r)})$ is the hidden variable for indexing components in $\Theta^{(r)}$. Note that we will always use i and j to index the components of the base model $\Theta^{(b)}$ and the reduced model $\Theta^{(r)}$, respectively. We will also use the short-hand $\Theta_i^{(b)}$ and $\Theta_j^{(r)}$ to denote the ith component of $\Theta^{(b)}$ and the jth component of $\Theta^{(r)}$, respectively. For example, we denote $p(y_{1:\tau}|z^{(b)} = i, \Theta^{(b)}) = p(y_{1:\tau}|\Theta_i^{(b)})$.

4.2 Parameter estimation

To obtain the reduced model, HEM [23] considers a set of N *virtual* samples drawn from the base model $\Theta^{(b)}$, such that $N_i = N\pi_i^{(b)}$ video samples are drawn from the ith component. The DT, however, has both observable Y and hidden state X variables (which are distinct from the hidden assignments of the overall mixture). To adapt HEM to DT models with hidden state variables, the most straightforward approach is to draw virtual samples from both X and Y according to their joint distribution. However, when computing the parameters of a new DT of the reduced model, there is no guarantee

that the virtual hidden states from the base models live in the same basis (equivalent DTs can be formed by scaling, rotating, or permuting A, C, and X). This basis mismatch will cause problems when estimating parameters from the virtual samples of the hidden states. The key insight is that, in order to remove ambiguity caused by multiple equivalent hidden state representations, we must only generate virtual samples from the observable Y, while treating the hidden states X as *additional* missing information in HEM.

We denote the set of N_i virtual video samples for the ith component as $Y_i = \{y_{1:\tau}^{(i,m)}\}_{m=1}^{N_i}$, where $y_{1:\tau}^{(i,m)} \sim \Theta_i^{(b)}$ is a single video sample and τ is the length of the virtual video (a parameter we can choose). The entire set of N samples is denoted as $Y = \{Y_i\}_{i=1}^{K^{(b)}}$. To obtain a consistent hierarchical clustering, we also assume that all the samples in a set Y_i are eventually assigned to the same reduced component $\Theta_j^{(r)}$, as in [23]. The parameters of the reduced model can then be computed using maximum likelihood estimation with the virtual video samples,

$$\Theta^{(r)^*} = \underset{\Theta^{(r)}}{\operatorname{argmax}} \log p(Y|\Theta^{(r)}), \tag{6}$$

where

$$\begin{aligned}\log p(Y|\Theta^{(r)}) &= \log \prod_{i=1}^{K^{(b)}} p(Y_i|\Theta^{(r)}) \\ &= \log \prod_{i=1}^{K^{(b)}} \sum_{j=1}^{K^{(r)}} \pi_j^{(r)} p(Y_i|z_i^{(r)}=j,\Theta^{(r)}) \\ &= \log \prod_{i=1}^{K^{(b)}} \sum_{j=1}^{K^{(r)}} \pi_j^{(r)} \int p(Y_i,X_i|\Theta_j^{(r)})dX_i \end{aligned} \tag{7}$$

and $X_i = \{x_{1:\tau}^{(i,m)}\}_{m=1}^{N_i}$ are the hidden-state variables corresponding to Y_i, and $z_i^{(r)}$ is the hidden variable assigning Y_i to a mixture component in $\Theta^{(r)}$. (7) requires marginalizing over hidden states $\{X, Z\}$, and hence (6) can be solved using the EM algorithm [35], which is an iterative optimization method that alternates between estimating the hidden variables with the current parameters, and computing new parameters given the estimated hidden variables (the "complete data"), given by

$$\begin{aligned}&\text{E-Step:} \quad \mathcal{Q}(\Theta^{(r)}, \hat{\Theta}^{(r)}) = \mathbb{E}_{X,Z|Y,\hat{\Theta}^{(r)}}[\log p(X,Y,Z|\Theta^{(r)})], \\ &\text{M-Step:} \quad \hat{\Theta}^{(r)^*} = \underset{\Theta^{(r)}}{\operatorname{argmax}}\, \mathcal{Q}(\Theta^{(r)}, \hat{\Theta}^{(r)}),\end{aligned}$$

where $\hat{\Theta}^{(r)}$ is the current estimate of the parameters, $p(X,Y,Z|\Theta^{(r)})$ is the "complete-data" likelihood, and $\mathbb{E}_{X,Z|Y,\hat{\Theta}^{(r)}}$ is the conditional expectation with respect to the current model parameters.

As is common with the EM formulation with mixture models, we introduce a hidden assignment variable $\mathbf{z}_{i,j}$, which is an indicator variable for when the video sample set Y_i is assigned to the jth component of $\Theta^{(r)}$, *i.e.*, when $z_i^{(r)} = j$. The complete-data log-likelihood is then

$$\begin{aligned}&\log p(X,Y,Z|\Theta^{(r)}) \\ &= \log \prod_{i=1}^{K^{(b)}} \prod_{j=1}^{K^{(r)}} \left(\pi_j^{(r)} p(Y_i,X_i|\Theta_j^{(r)})\right)^{\mathbf{z}_{i,j}} \\ &= \sum_{i=1}^{K^{(b)}} \sum_{j=1}^{K^{(r)}} \mathbf{z}_{i,j} \log \pi_j^{(r)} + \mathbf{z}_{i,j} \log p(Y_i,X_i|\Theta_j^{(r)}).\end{aligned} \tag{8}$$

We next derive the $\mathcal{Q}$ function, E-step, and M-step.

4.3 $\mathcal{Q}$ function for HEM-DTM

In the E-step, the $\mathcal{Q}$ function is obtained by taking the conditional expectation, with respect to the hidden variables $\{X, Z\}$, of the complete-data likelihood in (8)

$$\begin{aligned}\mathcal{Q}(\Theta^{(r)}, \hat{\Theta}^{(r)}) &= \sum_{i=1}^{K^{(b)}} \sum_{j=1}^{K^{(r)}} \mathbb{E}_{X,Z|Y,\hat{\Theta}^{(r)}} \Big[\mathbf{z}_{i,j} \log \pi_j^{(r)} \\ &\quad + \mathbf{z}_{i,j} \log p(Y_i,X_i|\Theta_j^{(r)})\Big] \\ &= \sum_{i=1}^{K^{(b)}} \sum_{j=1}^{K^{(r)}} \mathbb{E}_{Z|Y,\hat{\Theta}^{(r)}}[\mathbf{z}_{i,j}] \log \pi_j^{(r)} \\ &\quad + \mathbb{E}_{Z|Y,\hat{\Theta}^{(r)}}[\mathbf{z}_{i,j}] \mathbb{E}_{X|Y,\hat{\Theta}_j^{(r)}}[\log p(Y_i,X_i|\Theta_j^{(r)})]\end{aligned} \tag{9}$$

$$\begin{aligned}&= \sum_{i=1}^{K^{(b)}} \sum_{j=1}^{K^{(r)}} \hat{\mathbf{z}}_{i,j} \log \pi_j^{(r)} \\ &\quad + \hat{\mathbf{z}}_{i,j} \mathbb{E}_{X_i|Y_i,\hat{\Theta}_j^{(r)}} \left[\log p(Y_i,X_i|\Theta_j^{(r)})\right],\end{aligned} \tag{10}$$

where (9) follows from

$$\begin{aligned}&\mathbb{E}_{X,Z|Y,\hat{\Theta}^{(r)}}[\mathbf{z}_{i,j} \log p(Y_i,X_i|\Theta_j^{(r)})] \\ &= \mathbb{E}_{Z|Y,\hat{\Theta}^{(r)}} \mathbb{E}_{X|Z,Y,\hat{\Theta}^{(r)}}[\mathbf{z}_{i,j} \log p(Y_i,X_i|\Theta_j^{(r)})] \\ &= \mathbb{E}_{Z|Y,\hat{\Theta}^{(r)}}[\mathbf{z}_{i,j}] \mathbb{E}_{X|Y,\mathbf{z}_{i,j}=1,\hat{\Theta}^{(r)}}[\log p(Y_i,X_i|\Theta_j^{(r)})] \\ &= \hat{\mathbf{z}}_{i,j} \mathbb{E}_{X|Y,\hat{\Theta}_j^{(r)}}[\log p(Y_i,X_i|\Theta_j^{(r)})],\end{aligned}$$

and $\hat{\mathbf{z}}_{i,j}$ is the probability that sample set Y_i is assigned to component j in $\Theta^{(r)}$, obtained with Bayes' rule,

$$\begin{aligned}\hat{\mathbf{z}}_{i,j} &= \mathbb{E}_{Z|Y,\hat{\Theta}^{(r)}}[\mathbf{z}_{i,j}] = p(z_i^{(r)}=j|Y_i,\hat{\Theta}^{(r)}) \\ &= \frac{\pi_j^{(r)} p(Y_i|\hat{\Theta}_j^{(r)})}{\sum_{j'=1}^{K^{(r)}} \pi_{j'}^{(r)} p(Y_i|\hat{\Theta}_{j'}^{(r)})}.\end{aligned} \tag{11}$$

For the likelihood of the virtual samples, $p(Y_i|\hat{\Theta}_j^{(r)})$, we can obtain an approximation that only depends on the model parameters $\Theta_i^{(b)}$ that generated the samples,

$$\begin{aligned}\log p(Y_i|\hat{\Theta}_j^{(r)}) &= \sum_{m=1}^{N_i} \log p(y_{1:\tau}^{(i,m)}|\hat{\Theta}_j^{(r)}) \\ &= N_i \left[\frac{1}{N_i} \sum_{m=1}^{N_i} \log p(y_{1:\tau}^{(i,m)}|\hat{\Theta}_j^{(r)})\right] \\ &\approx N_i \mathbb{E}_{y|\Theta_i^{(b)}} \left[\log p(y_{1:\tau}|\hat{\Theta}_j^{(r)})\right],\end{aligned} \tag{12}$$

where (12) follows from the law of large numbers [23] (as $N_i \to \infty$). Substituting into (11), we get the expression for $\hat{\mathbf{z}}_{i,j}$, similar to the one derived in [23],

$$\hat{\mathbf{z}}_{i,j} = \frac{\pi_j^{(r)} \exp\left(N_i \mathbb{E}_{y|\Theta_i^{(b)}}[\log p(y_{1:\tau}|\hat{\Theta}_j^{(r)})]\right)}{\sum_{j'=1}^{K^{(r)}} \pi_{j'}^{(r)} \exp\left(N_i \mathbb{E}_{y|\Theta_i^{(b)}}[\log p(y_{1:\tau}|\hat{\Theta}_{j'}^{(r)})]\right)}. \quad (13)$$

For the last term in (10), we have

$$\begin{aligned}
&\mathbb{E}_{X_i|Y_i,\hat{\Theta}_j^{(r)}}[\log p(Y_i, X_i|\Theta_j^{(r)})] \\
&= \sum_{m=1}^{N_i} \mathbb{E}_{x_{1:\tau}^{(i,m)}|y_{1:\tau}^{(i,m)},\hat{\Theta}_j^{(r)}}[\log p(y_{1:\tau}^{(i,m)}, x_{1:\tau}^{(i,m)}|\Theta_j^{(r)})] \\
&\approx N_i \mathbb{E}_{y|\Theta_i^{(b)}}\left[\mathbb{E}_{x|y,\hat{\Theta}_j^{(r)}}[\log p(y_{1:\tau}, x_{1:\tau}|\Theta_j^{(r)})]\right], \quad (14)
\end{aligned}$$

where, again, (14) follows from the law of large numbers. Hence, the $\mathcal{Q}$ function is given by

$$\begin{aligned}
\mathcal{Q}(\Theta^{(r)}, \hat{\Theta}^{(r)}) &= \sum_{i=1}^{K^{(b)}} \sum_{j=1}^{K^{(r)}} \hat{\mathbf{z}}_{i,j} \log \pi_j^{(r)} \quad (15) \\
&+ \hat{\mathbf{z}}_{i,j} N_i \mathbb{E}_{y|\Theta_i^{(b)}}\left[\mathbb{E}_{x|y,\hat{\Theta}_j^{(r)}}[\log p(y_{1:\tau}, x_{1:\tau}|\Theta_j^{(r)})]\right].
\end{aligned}$$

Note that the form of the $\mathcal{Q}$ function in (15) is similar to that of the EM algorithm for DTM [8]. The first difference is the *additional* expectation w.r.t. $\Theta_i^{(b)}$. In HEM, each base DT $\Theta_i^{(b)}$ takes role of a "data-point" in standard EM, where an *additional* expectation w.r.t. $\Theta_i^{(b)}$ averages over the possible values of the "data-point", yielding the double expectation $\mathbb{E}_{y|\Theta_i^{(b)}}[\mathbb{E}_{x|y,\hat{\Theta}_j^{(r)}}[\cdot]]$. The second difference is the additional weighting of N_i on the second term, which accounts for the prior probabilities of each base DT.

Given these two differences with EM-DTM, the $\mathcal{Q}$ function for HEM-DTM will have the same form as that of EM [8, eqn. 16], but with two modifications: 1) conditional statistics of the hidden state will be computed using a double expectation, $\mathbb{E}_{y|\Theta_i^{(b)}}[\mathbb{E}_{x|y,\hat{\Theta}_j^{(r)}}[\cdot]]$; 2) an additional weight N_i will be applied when aggregating these expectations. Therefore, it can be shown that the HEM-DTM $\mathcal{Q}$ function is

$$\begin{aligned}
\mathcal{Q}(\Theta^{(r)}; \hat{\Theta}^{(r)}) &= \sum_j \hat{N}_j \log \pi_j \quad (16) \\
&- \frac{1}{2} \sum_j \mathrm{tr}(R_j^{-1}(\hat{\Lambda}_j - \hat{\Gamma}_j C_j^T - C_j \hat{\Gamma}_j^T + C_j \hat{\Phi}_j C_j^T)) \\
&\quad + \mathrm{tr}(S_j^{-1}(\hat{\eta}_j - \hat{\xi}_j \mu_j^T - \mu_j \hat{\xi}_j^T + \hat{M}_j \mu_j \mu_j^T)) \\
&\quad + \mathrm{tr}(Q_j^{-1}(\hat{\varphi}_j - \hat{\Psi}_j A_j^T - A_j \hat{\Psi}_j^T + A_j \hat{\phi}_j A_j^T)) \\
&\quad + \hat{M}_j(\tau \log|R_j| + (\tau - 1)\log|Q_j| + \log|S_j|),
\end{aligned}$$

where we define the aggregate statistics,

$$\begin{aligned}
&\hat{N}_j = \textstyle\sum_i \hat{\mathbf{z}}_{i,j}, && \hat{\Phi}_j = \textstyle\sum_i \hat{\mathbf{w}}_{i,j} \sum_{t=1}^{\tau} \hat{P}_{t,t|j}^{(i)}, \\
&\hat{M}_j = \textstyle\sum_i \hat{\mathbf{w}}_{i,j}, && \hat{\Psi}_j = \textstyle\sum_i \hat{\mathbf{w}}_{i,j} \sum_{t=2}^{\tau} \hat{P}_{t,t-1|j}^{(i)}, \\
&\hat{\xi}_j = \textstyle\sum_i \hat{\mathbf{w}}_{i,j} \hat{x}_{1|j}^{(i)}, && \hat{\varphi}_j = \textstyle\sum_i \hat{\mathbf{w}}_{i,j} \sum_{t=2}^{\tau} \hat{P}_{t,t|j}^{(i)}, \\
&\hat{\eta}_j = \textstyle\sum_i \hat{\mathbf{w}}_{i,j} \hat{P}_{1,1|j}^{(i)}, && \hat{\phi}_j = \textstyle\sum_i \hat{\mathbf{w}}_{i,j} \sum_{t=2}^{\tau} \hat{P}_{t-1,t-1|j}^{(i)}, \\
&\hat{\gamma}_j = \textstyle\sum_i \hat{\mathbf{w}}_{i,j} \sum_{t=1}^{\tau} \hat{u}_t^{(i)}, && \hat{\Lambda}_j = \textstyle\sum_i \hat{\mathbf{w}}_{i,j} \sum_{t=1}^{\tau} \hat{U}_{t|j}^{(i)}, \\
&\hat{\beta}_j = \textstyle\sum_i \hat{\mathbf{w}}_{i,j} \sum_{t=1}^{\tau} \hat{x}_{t|j}^{(i)}, && \hat{\Gamma}_j = \textstyle\sum_i \hat{\mathbf{w}}_{i,j} \sum_{t=1}^{\tau} \hat{W}_{t|j}^{(i)},
\end{aligned}$$

with $\hat{\mathbf{w}}_{i,j} = \hat{\mathbf{z}}_{i,j} N_i = \hat{\mathbf{z}}_{i,j} \pi_i^{(b)} N$. The individual conditional state expectations are

$$\hat{x}_{t|j}^{(i)} = \mathbb{E}_{y|\Theta_i^{(b)}}\left[\mathbb{E}_{x|y,\hat{\Theta}_j^{(r)}}[x_t]\right], \quad (17)$$

$$\hat{P}_{t|j}^{(i)} = \mathbb{E}_{y|\Theta_i^{(b)}}\left[\mathbb{E}_{x|y,\hat{\Theta}_j^{(r)}}[x_t x_t^T]\right], \quad (18)$$

$$\hat{P}_{t,t-1|j}^{(i)} = \mathbb{E}_{y|\Theta_i^{(b)}}\left[\mathbb{E}_{x|y,\hat{\Theta}_j^{(r)}}[x_t x_{t-1}^T]\right], \quad (19)$$

$$\hat{W}_{t|j}^{(i)} = \mathbb{E}_{y|\Theta_i^{(b)}}\left[(y_t - \bar{y}_j)\mathbb{E}_{x|y,\hat{\Theta}_j^{(r)}}[x_t]^T\right], \quad (20)$$

$$\hat{U}_{t|j}^{(i)} = \mathbb{E}_{y|\Theta_i^{(b)}}\left[(y_t - \bar{y}_j)(y_t - \bar{y}_j)^T\right], \quad (21)$$

$$\hat{u}_t^{(i)} = \mathbb{E}_{y|\Theta_i^{(b)}}\left[y_t\right], \quad (22)$$

where $\hat{\Theta}_j^{(r)}$ is the current parameter estimate for the jth component of the reduced model. Note that the expectations of the hidden state, conditioned on each component $\Theta_i^{(b)}$, are computed through a common DT model $\hat{\Theta}_j^{(r)}$. Hence, the potential problem with mismatches between the hidden-state bases of $\Theta^{(b)}$ is avoided. We next derive an efficient algorithm for computing the E-step expectations.

4.4 E-step expectations

To simplify notation, we denote the parameters of a given base mixture component $\Theta_i^{(b)}$ as $\Theta_b = \{A_b, Q_b, C_b, R_b, \mu_b, S_b, \bar{y}_b\}$, and likewise for a reduced mixture component $\hat{\Theta}_j^{(r)}$ as $\Theta_r = \{A_r, Q_r, C_r, R_r, \mu_r, S_r, \bar{y}_r\}$. We denote the corresponding expectations in (17-22) by dropping the i and j indices, $\{\hat{x}_t, \hat{P}_t, \hat{P}_{t|t-1}, \hat{W}_t, \hat{U}_t, \hat{u}_t\}$.

The inner expectations in (17-20), $\mathbb{E}_{x|y,\Theta_r}[\cdot]$, are related to the conditional state estimator of the Kalman smoothing filter of Θ_r, when given an observation $y_{1:\tau}$ [34, 26],

$$\begin{aligned}
\tilde{x}_{t|\tau}^{(r)} &= \mathbb{E}_{x_t|y_{1:\tau},\Theta_r}[x_t], \\
\tilde{V}_{t|\tau}^{(r)} &= \mathrm{cov}_{x_t|y_{1:\tau},\Theta_r}(x_t), \quad (23) \\
\tilde{V}_{t,t-1|\tau}^{(r)} &= \mathrm{cov}_{x_{t-1},t|y_{1:\tau},\Theta_r}(x_t, x_{t-1}),
\end{aligned}$$

where $\tilde{a}_{t|s}^{(r)}$ denotes the expectation at time t, conditioned on sequence $y_{1:s}$, w.r.t. Θ_r. Rewriting (17-20) in terms of the Kalman smoothing filter in (23),

$$\begin{aligned}
\hat{x}_t &= \mathbb{E}_{y|\Theta_b}\left[\tilde{x}_{t|\tau}^{(r)}\right], \\
\hat{P}_t &= \mathbb{E}_{y|\Theta_b}\left[\tilde{V}_{t|\tau}^{(r)} + \tilde{x}_{t|\tau}^{(r)}(\tilde{x}_{t|\tau}^{(r)})^T\right] \\
&= \hat{V}_t + \hat{\chi}_t + \hat{x}_t(\hat{x}_t)^T, \\
\hat{P}_{t,t-1} &= \mathbb{E}_{y|\Theta_b}\left[\tilde{V}_{t,t-1|\tau}^{(r)} + \tilde{x}_{t|\tau}^{(r)}(\tilde{x}_{t-1|\tau}^{(r)})^T\right] \quad (24) \\
&= \hat{V}_{t,t-1} + \hat{\chi}_{t,t-1} + \hat{x}_t(\hat{x}_{t-1})^T, \\
\hat{W}_t &= \mathbb{E}_{y|\Theta_b}\left[(y_t - \bar{y}_r)(\tilde{x}_{t|\tau}^{(r)})^T\right] \\
&= \hat{\kappa}_t + (\hat{u}_t - \bar{y}_r)(\hat{x}_t)^T,
\end{aligned}$$

where we define the double expectations,

$$\begin{aligned}
&\hat{V}_t = \mathbb{E}_{y|\Theta_b}[\tilde{V}_{t|\tau}^{(r)}], && \hat{V}_{t,t-1} = \mathbb{E}_{y|\Theta_b}[\tilde{V}_{t,t-1|\tau}^{(r)}], \\
&\hat{\kappa}_t = \mathrm{cov}_{y|\Theta_b}(y_t, \tilde{x}_{t|\tau}^{(r)}), && \hat{\chi}_t = \mathrm{cov}_{y|\Theta_b}(\tilde{x}_{t|\tau}^{(r)}), \quad (25) \\
&\hat{\chi}_{t,t-1} = \mathrm{cov}_{y|\Theta_b}(\tilde{x}_{t|\tau}^{(r)}, \tilde{x}_{t-1|\tau}^{(r)}).
\end{aligned}$$

Note that $\hat{x}_t$ is the output of the state estimator from a Kalman smoothing filter for Θ_r when the observation y is generated from a different model Θ_b. This is also known as "suboptimal filter analysis" or "sensitivity analysis" [26, 36], where the goal is to analyze filter performance when an optimal filter, according to some source distribution, is run on a different source distribution. Hence, the expectations in (24) and (25) can be calculated by sensitivity analysis of the Kalman smoothing filter for model Θ_r and source Θ_b. This procedure is summarized here, with the derivation appearing in Appendix A. First, given the Kalman filter for Θ_b and Θ_r, which calculates the statistics of the state x_t given the previous observations $y_{1:t-1}$,

$$\begin{aligned}\tilde{x}^{(b)}_{t|t-1} &= \mathbb{E}_{x_t|y_{1:t-1},\Theta_b}[x_t],\ \tilde{V}^{(b)}_{t|t-1} = \mathrm{cov}_{x_t|y_{1:t-1},\Theta_b}(x_t),\\ \tilde{x}^{(r)}_{t|t-1} &= \mathbb{E}_{x_t|y_{1:t-1},\Theta_r}[x_t],\ \tilde{V}^{(r)}_{t|t-1} = \mathrm{cov}_{x_t|y_{1:t-1},\Theta_r}(x_t),\end{aligned} \tag{26}$$

sensitivity analysis of the Kalman filter consists of marginalizing over the distribution of partial observations, $y_{1:t-1} \sim \Theta_b$, and computing the mean and covariance,

$$\hat{\mathbf{x}}_t = \mathbb{E}_{\Theta_b}\begin{bmatrix} x^{(b)}_t \\ \tilde{x}^{(b)}_{t|t-1} \\ \tilde{x}^{(r)}_{t|t-1}\end{bmatrix}, \hat{\mathbf{V}}_t = \mathrm{cov}_{\Theta_b}\Big(\begin{bmatrix} x^{(b)}_t \\ \tilde{x}^{(b)}_{t|t-1} \\ \tilde{x}^{(r)}_{t|t-1}\end{bmatrix}\Big), \tag{27}$$

of the true state $x^{(b)}_t$ and the state estimators $\hat{x}^{(b)}_{t|t-1}$ and $\hat{x}^{(r)}_{t|t-1}$. Second, using these results for the Kalman filter, sensitivity analysis of the Kalman smoothing filter consists of marginalizing (23) over the distribution of full observations, $y_{1:\tau} \sim \Theta_b$, yielding the expectations in (25) and (24).

The remaining two expectations in (21-22) are calculated from the marginal statistics of Θ_b,

$$\begin{aligned}\hat{U}_t &= \mathbb{E}_{y|\Theta_b}\left[(y_t - \bar{y}_r)(y_t - \bar{y}_r)^T\right]\\ &= \mathrm{cov}_{y|\Theta_b}(y_t, y_t) + (\hat{u}_t - \bar{y}_r)(\hat{u}_t - \bar{y}_r)^T\\ &= C_b\hat{\mathbf{V}}^{[1,1]}_t C_b^T + R_b + (\hat{u}_t - \bar{y}_r)(\hat{u}_t - \bar{y}_r)^T, \qquad (28)\\ \hat{u}_t &= \mathbb{E}_{y|\Theta_b}[y_t] = C_b\hat{\mathbf{x}}^{[1]}_t + \bar{y}_b, \qquad (29)\end{aligned}$$

Finally, for the soft assignments $\hat{\mathbf{z}}_{i,j}$, the expected log-likelihood term, $\mathbb{E}_{y|\Theta_b}[\log p(y|\Theta_r)]$, is calculated efficiently by expressing the observation log-likelihood of the DT in "innovation" form and marginalizing over $y \sim \Theta_b$, resulting in (35-37). This is derived in Appendix A.

Algorithm 1 summarizes the procedure for calculating the E-step expectations in (17-22). First, the Kalman filter and Kalman smoothing filter are run on Θ_b and Θ_r, using Algorithm 2. Next, sensitivity analysis is performed on the Kalman filter and Kalman smoothing filter via Algorithms 3 and 4, where Θ_r is the model and Θ_b is the source. Finally, the expectations and expected log-likelihood are calculated according to (30-37).

4.5 M-step

In the M-step of HEM for DTM, the paramters $\Theta^{(r)}$ are updated by maximizing the $\mathcal{Q}$ function. The form of the HEM $\mathcal{Q}$ function in (16) is identical to that of EM for DTM [8]. Hence, the equations for updating the parameters are identical

Algorithm 1 Expectations for HEM-DTM

1: **Input**: DT parameters Θ_b and Θ_r, length τ.
2: Run Kalman smoothing filter (Algorithm 2) on Θ_b and Θ_r to obtain $\{\tilde{V}^{(b)}_{t|t-1}, \tilde{V}^{(b)}_{t|\tau}, \tilde{V}^{(b)}_{t,t-1|\tau}\}$ and $\{\tilde{V}^{(r)}_{t|t-1}, \tilde{V}^{(r)}_{t|\tau}, \tilde{V}^{(r)}_{t,t-1|\tau}\}$.
3: Run sensitivity analysis on the Kalman filters, Θ_b and Θ_r, (Algorithm 3) to obtain $\{\hat{\mathbf{x}}_t, \hat{\mathbf{V}}_t\}$.
4: Run sensitivity analysis for the Kalman smoothing filters, Θ_b and Θ_r, (Algorithm 4) to obtain $\{\hat{x}_t, \hat{\chi}_t, \hat{\chi}_{t,t-1}, \hat{\kappa}_t\}$.
5: Compute E-step expectations, for $t = \{1, \cdots, \tau\}$:

$$\hat{u}_t = C_b\hat{\mathbf{x}}^{[1]}_t + \bar{y}_b, \tag{30}$$
$$\hat{U}_t = C_b\hat{\mathbf{V}}^{[1,1]}_t C_b^T + R_b + (\hat{u}_t - \bar{y}_r)(\hat{u}_t - \bar{y}_r)^T, \tag{31}$$
$$\hat{P}_t = \tilde{V}^{(r)}_{t|\tau} + \hat{\chi}_t + \hat{x}_t(\hat{x}_t)^T, \tag{32}$$
$$\hat{P}_{t,t-1} = \tilde{V}^{(r)}_{t,t-1|\tau} + \hat{\chi}_{t,t-1} + \hat{x}_t(\hat{x}_{t-1})^T, \tag{33}$$
$$\hat{W}_t = \hat{\kappa}_t + (\hat{u}_t - \bar{y}_r)(\hat{x}_t)^T. \tag{34}$$

6: Compute expected log-likelihood ℓ:

$$\hat{\Sigma}_t = C_r\tilde{V}^{(r)}_{t|t-1}C_r^T + R_r, \quad \hat{\Lambda}_t = \hat{\mathbf{V}}^{[3,3]}_t + \hat{\mathbf{x}}^{[3]}_t(\hat{\mathbf{x}}^{[3]}_t)^T, \tag{35}$$
$$\hat{\lambda}_t = C_b\hat{\mathbf{V}}^{[2,3]}_t + (C_b\hat{\mathbf{x}}^{[1]}_t + \bar{y}_b - \bar{y}_r)(\hat{\mathbf{x}}^{[3]}_t)^T, \tag{36}$$
$$\begin{aligned}\ell = \sum_{t=1}^{\tau} &\tfrac{-1}{2}\mathrm{tr}\left[\hat{\Sigma}_t^{-1}(\hat{U}_t - \hat{\lambda}_t C_r^T - C_r\hat{\lambda}_t^T + C_r\hat{\Lambda}_t C_r^T)\right]\\ &- \tfrac{1}{2}\log\left|\hat{\Sigma}_t\right| - \tfrac{m}{2}\log(2\pi).\end{aligned} \tag{37}$$

7: **Output**: $\{\hat{x}_t, \hat{P}_t, \hat{P}_{t,t-1}, \hat{W}_t, \hat{U}_t, \hat{u}_t\}, \ell$.

to EM for DTM, although the aggregate expectations are different. Each DT component $\Theta^{(r)}_j$ is updated as

$$\begin{aligned} C^*_j &= \hat{\Gamma}_j\hat{\Phi}_j^{-1}, \quad R^*_j = \tfrac{1}{\hat{N}_j}(\hat{\Lambda}_j - C^*_j\hat{\Gamma}_j),\\ A^*_j &= \hat{\Psi}_j\hat{\phi}_j^{-1}, \quad Q^*_j = \tfrac{1}{\hat{N}_j}(\hat{\varphi}_j - A^*_j\hat{\Psi}_j^T),\\ \mu^*_j &= \tfrac{1}{\hat{N}_j}\hat{\xi}_j, \qquad S^*_j = \tfrac{1}{\hat{N}_j}\hat{\eta}_j - \mu^*_j(\mu^*_j)^T,\\ \pi^*_j &= \tfrac{\hat{N}_j}{K^{(b)}}, \qquad \bar{y}^*_j = \tfrac{1}{\hat{N}_j}(\hat{\gamma}_j - C^*_j\hat{\beta}_j).\end{aligned} \tag{38}$$

5 Applications and Experiments

In this section, we discuss several novel applications of HEM-DTM to video and motion analysis, including hierarchical motion clustering, semantic motion annotation, and DT codebook generation for the *bag-of-systems* video representation, which are illustrated in Figure 1. These applications exploit several desirable properties of HEM to obtain promising results. First, given a set of input DTs, HEM estimates a novel set of fewer DTs that represents the input in a manner that is consistent with the underlying generative probabilistic models, by maximizing the log-likelihood of "virtual" samples generated from the input DTs. As a result, the clusters formed by HEM are also consistent with the probabilistic framework. Second, HEM can estimate models on large datasets, by breaking the learning problem into smaller pieces. In particular, intermediate models are learned on small non-overlapping portions of a large dataset, and the final model is estimated by running HEM on the intermediate models. Because HEM is based on maximum-likelihood principles, it drives model estimation towards similar optimal parameter values as performing maximum-likelihood estimation on the full dataset. However, the computer memory requirements are significantly less, since we no longer have to store the entire dataset during parameter estimation. In addition, the intermediate models are estimated independently of each other, so the task can be easily

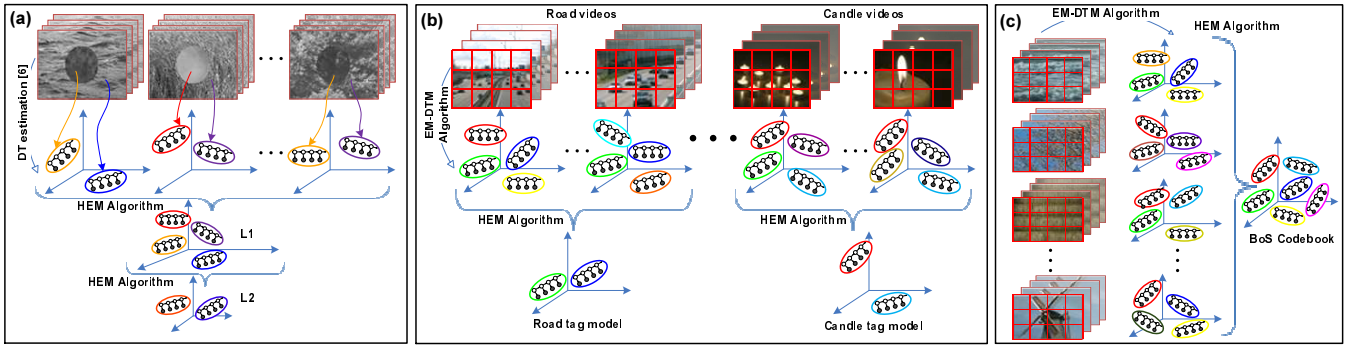

Fig. 1: Applications of the HEM-DTM algorithm: a) hierarchical clustering of video textures; b) learning DT annotation models; c) training a bag-of-systems (BoS) codebook.

parallelized. In the remainder of the section, we present three applications of HEM-DTM to video and motion analysis.

5.1 Implementation notes

In the following experiments, the EM-DTM algorithm is first used to learn *video-level* DTMs from overlapping vectorized video patches (spatio-temporal cubes) extracted from the video. We initialize EM-DTM using an iterative "component splitting" procedure suggested in [8], where EM is run repeatedly with an increasing number of mixture components. Specifically, we start by estimating a DTM with $K = 1$ components by running EM-DTM to convergence. Next, we select the DT component, and duplicate it to form two components (this is the "splitting"), followed by slightly perturbing the DT parameters. This new DTM with $K = 2$ components serves as the initialization for EM-DTM, which is again run until convergence. The process is repeated until the desired number of components is reached. We use a growing schedule of $K = \{1, 2, 4, 8, 16\}$, and perturb the observation matrix C when creating new DT components. We use a similar procedure to initialize the reduced DTM when running HEM-DTM. We set the virtual sample parameters to $\tau = 20$ and $N = 1000$. The state-space dimension is set to $n = 10$. The likelihood of a video under a DT, $p(y_{1:\tau}|\Theta)$, is calculated efficiently using the innovation form of the likelihood in (74). Finally, we make a standard i.i.d. assumption on the observation noise of the DT, *i.e.*, $R = rI$. In this case, the inversion of large $m \times m$ covariance matrices, *e.g.*, in (37) and (48), is calculated efficiently using the matrix inversion lemma.

5.2 Hierarchical clustering of video textures

We first consider hierarchical motion clustering of video textures, by successively clustering DTs with the HEM algorithm, as illustrated in Figure 1a. Given a set of K_1 video textures, spatio-temporal cubes are extracted from the video and a DT is learned for each video texture. This forms the first level of the hierarchy (the video-level DT). The next level in the hierarchy is formed by clustering the DTs from the previous level into K_2 groups with the HEM algorithm ($K_2 < K_1$). The DT cluster centers are selected as the representative models at this level, and the process is continued with each level in the hierarchy learned from the preceding level. The result is a tree representation of the video dataset, with similar textures grouped together in the hierarchy. Note that this type of hierarchy could not be built in a straightforward manner using the EM algorithm on the original spatio-temporal cubes. While it is possible to learn several DTMs with successively smaller values of K, there is no guarantee that the resulting mixtures, or the cluster memberships of the video patches, will form a tree.

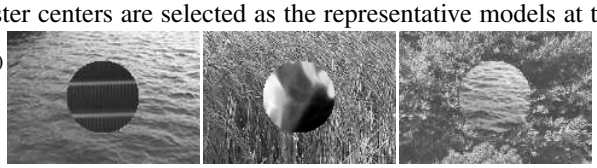

Fig. 2: Video texture examples: a) video with 2 textures; b) ground-truth labels.

5.2.1 Experimental setup

We illustrate hierarchical motion clustering on the video texture dataset from [8]. This dataset is composed of 99 video sequences, each containing 2 distinct video textures (see Figure 2 for examples). There are 12 texture classes in total, ranging from water (sea, river, pond) to plants (grass and trees), to fire and steam. To obtain the first level of the hierarchy, we learn one DT for each texture in each video (the locations of the textures are known), and pool these DTs together to form a DTM with $K_1 = 198$ components. Each DT is learned using [6] on 100 spatio-temporal cubes ($5 \times 5 \times 60$ pixels) sampled from the texture segment. The second level of the hierarchy is obtained by running HEM on the level-1 DT mixture to reduce the number of components to $K_2 = 12$. Finally, the

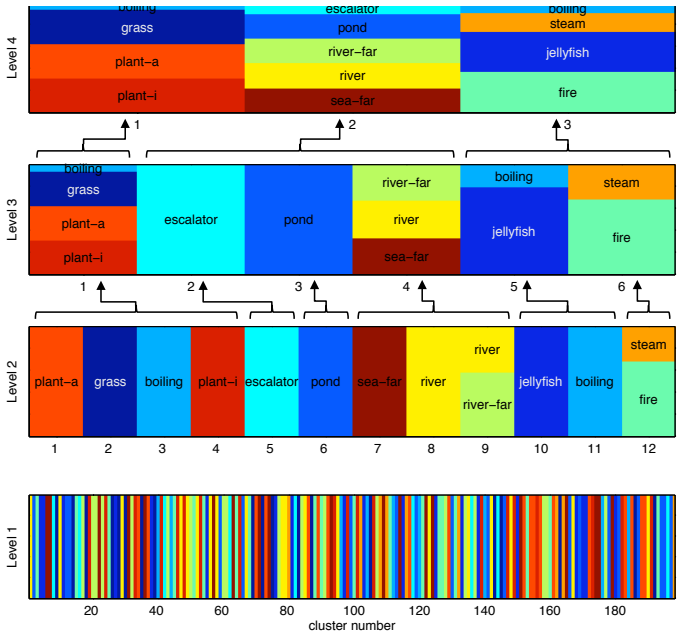

Fig. 3: Hierarchical clustering of video textures: The arrows and brackets show the cluster membership from the preceding level (the groupings between Levels 1 and 2 are omitted for clarity).

third and fourth levels are obtained by running HEM on the previous level for $K_3 = 6$ and $K_4 = 3$ clusters, respectively.

5.2.2 Clustering Results

Figure 3 shows the hierarchical clustering that is obtained with HEM. The first level contains the DTs that represent each texture segment in the database. Each vertical bar represents one DT, where the color indicates the ground-truth cluster label (texture name). In the second level, the 12 DT components are shown as vertical bars, where the colors indicate the proportion of the cluster membership with a particular ground-truth cluster label. In most cases, each cluster corresponds to a single texture (*e.g.*, *grass*, *escalator*, *pond*), which illustrates that HEM is capable of clustering DTs into similar motions. The Rand index for the level-2 clustering using HEM is 0.973 (for comparison, clustering histograms-of-oriented-optical-flow using K-means yields a Rand index of 0.958). One error is seen in the HEM cluster with both the *river* and *river-far* textures, which is reasonable considering that the *river-far* texture contains both near and far perspectives of water. Moving up to the third level of the hierarchy, HEM forms two large clusters containing the plant textures (*plant-i*, *plant-a*, *grass*) and water textures (*river-far*, *river*, *sea-far*). Finally, in the fourth level, the video textures are grouped together according to broad categories: plants (*grass*, *plant-a*, *plant-i*), water (*pond*, *river-far*, *river*, *sea-far*), and rising textures (*fire*, *jellyfish*, and *steam*). These results illustrate that HEM for DT is capable of extracting meaningful clusters in a hierarchical manner.

5.3 Semantic video texture annotation

In this section, we formulate the annotation of video sequences as a supervised multi-class labeling (SML) problem [25] using DTM models.

5.3.1 Video Annotation Framework

A video sequence is first decomposed into spatio-temporal cubes as $\mathcal{Y} = \{y_{1:\tau}^{(i)}\}_{i=1}^{N}$ where each $y_{1:\tau}^{(i)}$ is a vectorized video patch of length τ. The number of video cubes N depends on the size and length of the video. Semantic content of the video can be represented with a vocabulary $\mathcal{V} = \{w_1, \cdots, w_{|\mathcal{V}|}\}$ of unique tags (*e.g.*, trees, river, and directed motion), with size $|\mathcal{V}|$. Each video is represented with an annotation vector of the form $c = \{c_1, \ldots, c_{|\mathcal{V}|}\}$, where a particular entry $c_k > 0$ if there is some association of the video with the kth tag in the vocabulary.

Each tag w_k is modeled as a probability distribution over the video cubes, *i.e.*, $p(y_{1:\tau}^{(i)}|w_k)$, which in our case will be a DTM model. The annotation task is then to find a subset $\mathcal{W} = \{w_1, \ldots, w_A\} \subseteq \mathcal{V}$ of A tags, that best describes a novel video $\mathcal{Y}$. Given the novel video, the most relevant tags are those with highest posterior probability according to Bayes' rule,

$$p(w_k|\mathcal{Y}) = \frac{p(\mathcal{Y}|w_k)p(w_k)}{p(\mathcal{Y})}, \tag{39}$$

where $p(w_k)$ is the kth tag prior, $p(\mathcal{Y})$ is the prior for the video and $p(\mathcal{Y}|w_k) = \prod_{i=1}^{N} p(y_{1:\tau}^{(i)}|w_k)$. The video can then be represented as a semantic multinomial $\mathbf{p} = [p(w_1|\mathcal{Y}), \ldots, p(w_{|\mathcal{V}|}|\mathcal{Y})]$. The top A tags according to the semantic multinomial $\mathbf{p}$ are then selected as the annotations of the video. To promote annotation using a diverse set of tags, we also assume a uniform prior, $p(w_k) = 1/|\mathcal{V}|$.

5.3.2 Learning tag models with HEM

For each tag w_k, the tag distribution $p(y_{1:\tau}^{(i)}|w_k)$ is modeled with a DTM model, which is estimated from the set of training videos associated with the particular tag. One approach to estimation is to extract all the video fragments from the relevant training videos for the tag, and then run the EM algorithm [8] directly on this data to learn the tag-level DTM. This approach, however, requires storing many video fragments in memory (RAM) for running the EM algorithm. For even modest-sized databases, the memory requirements can exceed the RAM capacity of most computers.

To allow efficient training in computation time and memory requirements, the learning procedure is split into two steps. First, a video-level DTM model is learned for each video in the training set using the standard EM algorithm [8]. Next, a tag-level model is formed by pooling together all the video level DTMs associated with a tag, to form a large mixture (*i.e.*, each DT component in a relevant video-level DTM becomes a component in the large mixture). However, a drawback of this model aggregation approach is that the number of DTs in the DTM tag model grows linearly with the size of the training data, making inference computationally inefficient when using large training sets. To alleviate this problem, the DTM tag models formed by model aggregation are reduced to a representative DTM with fewer components by using the HEM algorithm. The HEM algorithm clusters together similar DTs in the video-level DTMs, thus summarizing the common information in videos associated with a particular tag. The new DTM tag model allows for more efficient inference, due to fewer mixture components, while maintaining a reliable representation of the tag-level model. The process for learning a tag-level DTM model from video level DTMs is illustrated in Figure 1b.

5.3.3 Experimental setup

For the annotation experiment we use the DynTex dataset [37], which consists of over 650 videos, mostly in everyday

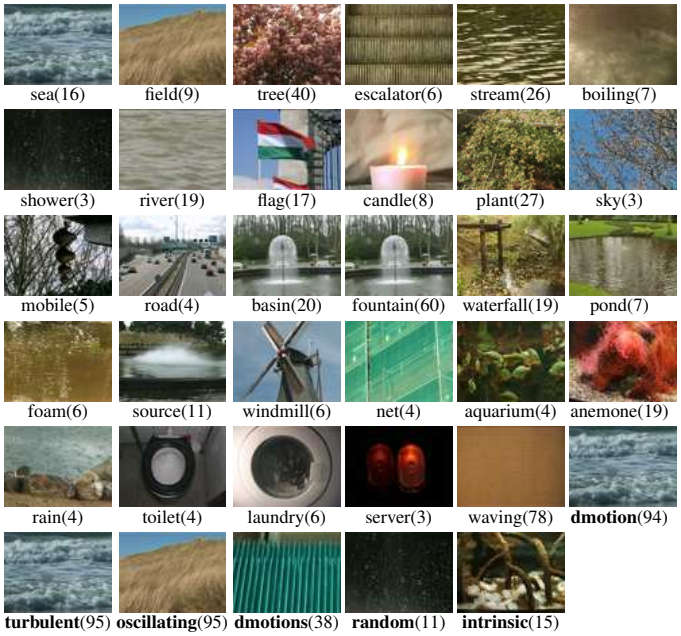

Fig. 4: List of tags with example thumbnails and video count for the DynText dataset. "Structural" tags are in bold.

TABLE 1: Annotation Results for different methods on the DynTex dataset.

	Average Precision	Average Recall	Average F-Measure	Tags with Recall > 0
DTM-HEM	**0.420**	**0.507**	**0.397**	33
GMM-HEM-DCT	0.275	0.461	0.292	**34**
GMM-HEM-OPF	0.191	0.201	0.144	30

surroundings. Ground truth annotation information is present for 385 sequences (called the "golden set"), based on a detailed analysis of the physical processes underlying the dynamic textures. We select the 35 most frequent tags in DynTex for annotation comprising of 337 sequences. The tags are also grouped into two categories: 1) *process* tags, which describe the physical texture process (*e.g.*, sea, field, and tree), and are mainly based on the appearance; 2) *structural* tags, which describe only the motion characteristics (*e.g.*, turbulent and oscillating), and are largely independent of appearance. Note that videos with a particular *structural* tag can have a wide range of appearances, since the tag only applies to underlying motion. Each video has an average of 2.34 tags [1]. Figure 4 shows an example of each tag alongside the number of sequences in the dataset.

Each video is truncated to 50 frames, converted to grayscale and downsampled 3 times using bicubic interpolation, resulting in a size of $192 \times 240 \times 50$. Overlapping spatio-temporal cubes of size $7\times7\times20$ (step: $4\times4\times10$) are extracted from the videos. We only consider patches with significant motion, by ignoring a patch if any pixel has variance < 5 in time. Video-level DTMs are learned with $K = 16$ components to capture enough of the temporal diversity present in each sequence, while tag-level DTMs use $K^{(r)} = 8$ components.

Annotation performance is measured following the procedure described in [25]. Annotation accuracy is reported by computing precision, recall and F-score for each tag, and then averaging over all tags. Per-tag precision is the probability that the model correctly uses the tag when annotating a video. Per-tag recall is the probability that the model annotates a video that should have been annotated with the tag. Precision, recall and F-score measure for a tag w are defined as:

$$P = \frac{|W_C|}{|W_A|}, R = \frac{|W_C|}{|W_H|}, F = 2((P)^{-1} + (R)^{-1})^{-1}, \quad (40)$$

where $|W_H|$ is the number of sequences that have tag w in the ground truth, $|W_A|$ is the number of times the annotation system uses W when automatically tagging a video, and $|W_C|$ is the number of times w is correctly used. In case a tag is never selected for annotation, the corresponding precision (that otherwise would be undefined) is set to the tag prior from the training set, which equals the performance of a random classifier.

To investigate the advantage of the DTM's temporal representation, we compare the annotation performance of HEM-DTM to the hierarchically-trained Gaussian mixture models using DCT features [25] (GMM-HEM-DCT) and using optical flow features [9] (GMM-HEM-OPF). The dataset is split into 50% training and 50% test sets, with each video appearing exactly once in either set. Results are averaged over 5 trials, using different random training/test sets.

1. Details of the data set and more results can be found at: http://visal.cs.cityu.edu.hk/research/hemdtm/

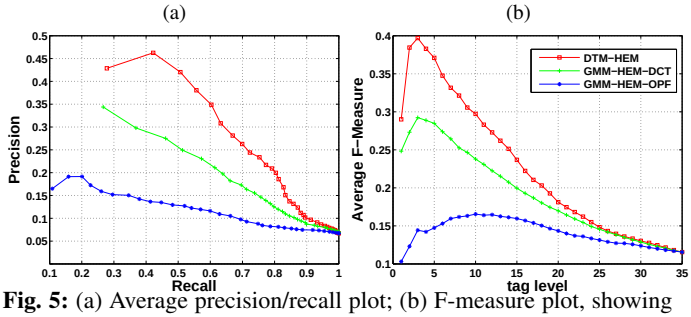

Fig. 5: (a) Average precision/recall plot; (b) F-measure plot, showing all tag-levels, for different methods on DynTex data set.

5.3.4 Annotation Results

Table 1 shows the average precision, recall, and F-measure for annotation with $A = 3$ tags, while Figure 5 shows these values for all 35 tag levels. Video annotation using DTMs outperforms using DCT and optical flow features, with an F-score of 0.397 versus 0.292 and 0.144. Overall, this suggests that the DTM can better capture both the appearance and dynamics of the video texture processes.

TABLE 2: Per Tag performance on DynTex data set.

		Precision			Recall			F-Measure		
		DTM	DCT	OPF	DTM	DCT	OPF	DTM	DCT	OPF
process tags	*anemone*	0.291	0.297	0.103	0.751	0.773	0.606	0.410	**0.421**	0.173
	aquarium	0.140	0.090	0.026	0.167	0.633	0.500	0.150	**0.152**	0.050
	basin	0.150	0.216	0.056	0.353	0.263	0.067	0.191	**0.220**	0.054
	boiling	0.235	0.421	0.120	1.000	0.720	0.390	0.372	**0.528**	0.175
	candle	0.482	0.117	0.022	1.000	0.860	0.050	**0.631**	0.204	0.031
	escalator	0.733	0.583	0.800	0.450	0.617	0.300	0.513	**0.598**	0.420
	field	0.486	0.328	0.072	0.312	0.761	0.460	0.350	**0.457**	0.118
	flag	0.226	0.380	0.311	0.575	0.501	0.365	0.308	**0.417**	0.320
	foam	0.362	0.125	0.082	0.667	0.733	0.300	**0.431**	0.204	0.126
	fountain	0.437	0.383	0.627	0.501	0.222	0.390	0.463	0.277	**0.476**
	laundry	0.138	0.085	0.061	0.800	0.520	0.267	**0.232**	0.141	0.096
	mobile	0.800	0.080	0.300	0.217	0.217	0.200	**0.340**	0.108	0.233
	net	0.700	0.600	0.020	0.400	0.733	0.267	0.480	**0.628**	0.037
	plant	0.261	0.255	0.086	0.781	0.571	0.378	**0.390**	0.352	0.137
	pond	0.273	0.196	0.058	0.720	0.640	0.133	**0.367**	0.273	0.081
	rain	1.000	0.390	0.083	0.733	0.533	0.133	**0.800**	0.326	0.102
	river	0.287	0.256	0.243	0.456	0.349	0.145	**0.351**	0.292	0.176
	road	1.000	0.238	1.000	0.600	0.600	0.600	**0.700**	0.320	0.700
	sea	0.539	0.474	0.072	0.896	0.745	0.096	**0.661**	0.572	0.082
	server	1.000	0.056	0.000	0.700	0.300	0.000	**0.800**	0.094	0.000
	shower	0.000	0.017	0.000	0.000	0.100	0.000	0.000	**0.029**	0.000
	sky	0.000	0.000	0.000	0.000	0.000	0.000	**0.000**	0.000	0.000
	source	0.323	0.057	0.025	0.350	0.267	0.033	**0.316**	0.093	0.029
	stream	0.335	0.331	0.186	0.249	0.282	0.097	0.249	**0.279**	0.128
	toilet	0.600	0.157	0.000	0.600	0.567	0.000	**0.527**	0.234	0.000
	tree	0.503	0.594	0.140	0.827	0.741	0.183	0.624	**0.656**	0.154
	waterfall	0.119	0.153	0.089	0.187	0.153	0.197	0.133	**0.140**	0.111
	windmill	0.179	0.059	0.000	0.400	0.567	0.000	**0.217**	0.106	0.000
structural tags	*dmotion*	0.468	0.457	0.510	0.335	0.152	0.123	**0.385**	0.221	0.189
	dmotions	0.212	0.254	0.235	0.269	0.247	0.091	0.227	**0.230**	0.114
	interinsic	0.491	0.174	0.064	0.725	0.578	0.106	**0.560**	0.263	0.077
	oscillating	0.681	0.796	0.214	0.722	0.609	0.104	**0.692**	0.689	0.138
	random	0.383	0.169	0.072	0.251	0.157	0.157	**0.229**	0.161	0.092
	turbulent	0.498	0.427	0.409	0.429	0.268	0.131	**0.456**	0.326	0.198
	waving	0.374	0.409	0.616	0.313	0.152	0.150	**0.339**	0.221	0.235
	Process	**0.414**	**0.248**	**0.164**	**0.525**	**0.499**	**0.220**	***0.393***	**0.290**	**0.143**
	Structural	**0.444**	**0.384**	**0.303**	**0.435**	**0.309**	**0.123**	***0.412***	**0.302**	**0.149**


55fc410
(truth)boiling,turbulent
(DTM)**boiling**,**turbulent**,laundry,flag,waving
(DCT)**boiling**,candle,interinsic,**turbulent**,flag
(OPF)candle,laundry,**turbulent**,interinsic,flag


644b510
(truth)plant,oscillating
(DTM)**plant**,**oscillating**,tree,anemone,fountain
(DCT)foam,anemone,**oscillating**,**plant**,basin
(OPF)field,**plant**,toilet,anemone,**oscillating**


645c610
(truth)road,dmotions
(DTM)**road**,**dmotions**,foam,waterfall,waving
(DCT)**road**,waving,**dmotions**,turbulent,dmotion
(OPF)**road**,waterfall,turbulent,net,dmotion


646a320
(truth)flag,waving
(DTM)**flag**,**waving**,laundry,turbulent,boiling
(DCT)**flag**,**waving**,windmill,laundry,candle
(OPF)**flag**,**waving**,laundry,windmill,turbulent


6483510
(truth)fountain,foam,turbulent,dmotions
(DTM)**foam**,stream,**dmotions**,**fountain**,**turbulent**
(DCT)waterfall,**foam**,stream,oscillating,dmotion
(OPF)toilet,plant,field,oscillating,anemone


6487910
(truth)plant,oscillating
(DTM)**plant**,**oscillating**,tree,aquarium,stream
(DCT)waterfall,stream,**oscillating**,foam,**plant**
(OPF)toilet,**plant**,field,**oscillating**,anemone

Fig. 6: Annotation examples from the DynTex database, showing ground truth, DTM, GMM-DCT, and GMM-OPF annotations. Automatic annotations that match the ground-truth annotations are in bold.

Table 2 presents the annotation performance for the individual tags, as well as averages over the *process* and *structural* categories. For the *process* category, DTM outperforms DCT on average F-score (0.393 versus 0.290), although the performance on individual tags is mixed. In some cases, appearance (via DCT features) is sufficient to identify the relevant texture (*e.g.*. net). For the *structural* category, DTM also outperforms DCT with an average F-score of 0.412 versus 0.302, while also dominating DCT on all but one individual structural tags. In these cases, appearance features cannot sufficiently model the *structural* tags, since these tags contain significant variation in appearance. On the other hand, DTM is able to learn the common motion characteristics, in spite of the variation in appearance. Finally, Figure 6 presents some example annotations for different videos using the top-5 tags. To give a sense of the computational cost of these annotation experiments, the average runtime using a standard PC (3.16 Ghz, C++, OpenCV) was 3.3 minutes to learn a video-level DTM, 2.4 minutes to learn a tag model from video-level DTMs, and 2.3 minutes to annotate a single video.

5.3.5 Effect of various training parameters

We further investigated the effect of varying the number of states, number of components, and training set size. Figures 7(a) and 7(b) show the F-score when varying the number of video-level components and tag-level components. In general, increasing the number of components at the video- and tag-level improves performance, since the DTM can better capture the variations in underlying dynamics of the video sequence. Figure 7(c) shows the annotation performance while varying the dimension of the state space n. Increasing n tends to improve performance. Finally, Figure 7(d) presents the average F-score while changing the size of the training set, by selecting a subset of the training set. The performance improves consistently with the increase in number of videos.

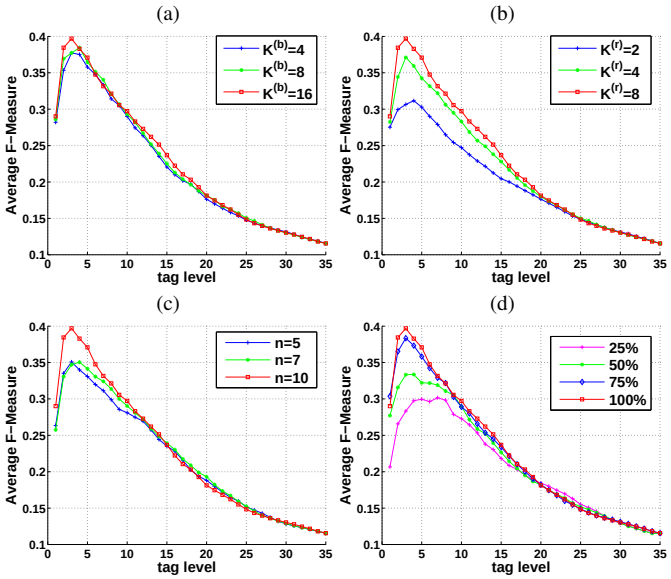

Fig. 7: Effect on annotation performance when varying the number of: (a) base components; (b) tag-level components; (c) states; (d) training videos.

5.4 HEM-trained bag-of-systems codebook for dynamic texture recognition

The bag-of-systems (BoS) representation [18] is a descriptor of motion in a video, where dynamic texture (DT) codewords represent the typical motion patterns in spatio-temporal patches extracted from the video. The BoS representation of videos is analogous to the bag-of-visual-words representation of images, where images are represented by counting the occurrences of visual codewords in the image. Specifically, in the BoS framework the codebook is formed by generative time-series models instead of words, each of them compactly characterizing typical textures and dynamics patterns of pixels in a spatio-temporal patch. Hence, each video is represented by a BoS histogram with respect to the codebook, by assigning individual spatio-temporal patches to the most likely codeword, and then counting the frequency with which each codeword is selected.

To learn the DT codebook, [18, 38] first estimate individual DTs, learned from spatio-temporal cubes extracted at spatio-temporal interest points [18],or from non-overlapping samples from the video [38]. Codewords are then generated by clustering the individual DTs using a combination of non-linear dimensionality reduction (NLDR) and K-means clustering. Due to the pre-image problem of kernelized NLDR, this clustering method is not capable of producing *novel* DT codewords, as discussed in Section 2. In this section, we use the HEM-DTM algorithm to generate *novel* DT codewords for the bag-of-systems representation, thus improving the robustness of the motion descriptor. We validate the HEM-trained BoS codebook on the task of dynamic texture recognition, while comparing with existing state-of-the-art methods [14, 39, 40, 18, 20].

5.4.1 Learning a BoS Codebook with HEM-DTM

The procedure for learning a BoS codebook is illustrated in Figure 1c. First, for each video in the training corpus, a dense sampling of spatio-temporal cubes is extracted, and a DTM

is learned with the EM algorithm [8]. Next, these DTMs are pooled together to form one large DTM, and the number of mixture components is reduced using the HEM-DTM algorithm. Finally, the novel DT cluster centers are selected as the BoS codewords. Note that this method of codebook generation is able to exploit all the training data, as opposed to only a subset selected via interest-point operators as in [18], or non-overlapping samples as in [38]. This is made possible through the efficient hierarchical learning of the codebook model, as discussed in the previous sections.

Given the BoS codebook, the BoS representation of a video is formed by first counting the number of occurrences of each codeword in the video, where each spatio-temporal cube is assigned to the codeword with largest likelihood. Next, a BoS histogram (weight vector w) is formed using the standard term frequency (TF) or term frequency inverse document frequency (TFIDF) representations,

$$\text{TF: } w_{ik} = \frac{N_{ik}}{N_i}, \quad \text{TFIDF: } w_{ik} = \frac{N_{ik}}{N_i} \log\left(\frac{V}{V_k}\right), \qquad (41)$$

where w_{ik} is the kth codeword entry for the ith video, N_{ik} is the number of times codeword k appears in video i, $N_i = \sum_k N_{ik}$ is the total number of codewords for video i, V is the total number of training videos, and V_k is the number of training videos in which codeword k occurs.

5.4.2 Related Work and Datasets

Current approaches to dynamic texture recognition use DT models [13, 14, 20, 18] or aggregations of local descriptors [39, 40]. [13, 14] represent each video as a DT model, and then leverage nearest neighbors or support vector machine (SVM) classifiers, by adopting an appropriate distance functions between dynamic textures, *e.g.*., Martin distance [13] or Kullback-Leibler divergence [14]. The resulting classifiers are largely dependent on the *appearance* of the video, *i.e.*, the particular viewpoint of each texture. Subsequent methods address this issue by proposing translation-invariant or viewpoint-independent approaches: [20] proposes distances between DTs based only on the spectrum or cepstrum of the hidden-state process x_t, while ignoring the appearance component of the model; [18] proposes a *bag-of-systems* representation for videos, formed by assigning spatio-temporal patches, which are selected by interest-point operators, to DT codewords. The patch-based framework of BoS is less sensitive to changes in viewpoint than the approaches based on holistic appearance [13, 14].

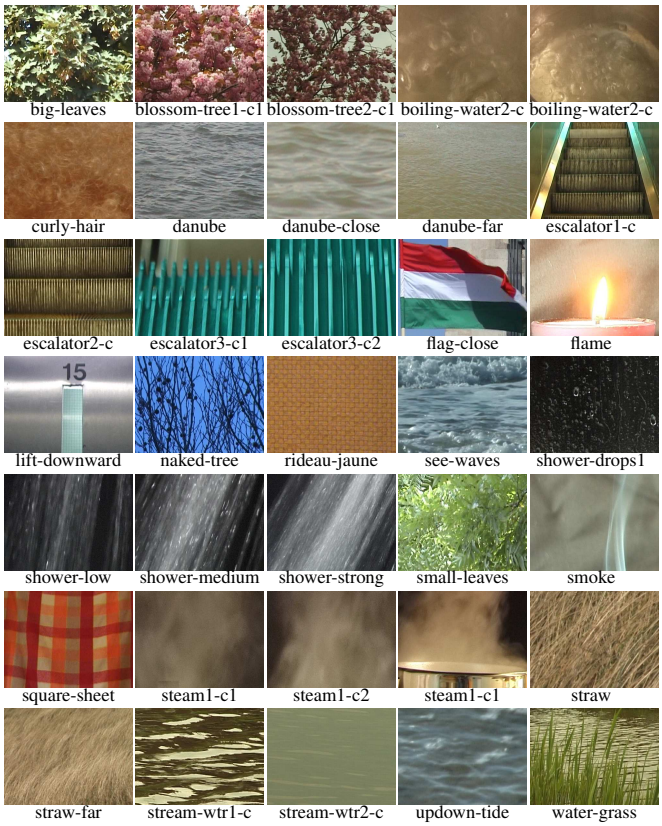

Fig. 8: Examples from DynTex35.

In contrast to using DT models, [39, 40] aggregate local descriptors to form a video descriptor. [39] uses distributions of local space-time oriented structures, while [40] concatenates local binary pattern (LBP) histograms extracted from three orthogonal planes in space-time (XY, XT, YT). While these two descriptors are less sensitive to viewpoint, they both ignore the underlying long-term motion dynamics of the texture process.

The datasets used by the above papers are either based on the UCLA [13] or DynTex [37] video textures datasets, with modifications in order to test viewpoint-invariance:

- **UCLA50**: the original UCLA dataset [13] consists of 50 classes, with 4 videos per class. The original videos are grayscale with a frame size of 160×110. [13, 14] crop the videos to a representative 48×48 video patch so that the texture is from the same viewpoint. In our BoS experiments, we use the original *uncropped* versions.
- **UCLA39**: [20] considers 39 classes from UCLA, which do not violate the assumption of spatial stationarity. Each video is cropped into a left subvideo and a right subvideo (both 48×48), where one side is used for training and the other for testing. This classification task is significantly more challenging than UCLA50, since the appearances of the training videos are quite different than those of the test videos.
- **UCLA9**: [18] groups related classes in UCLA into 9 super-classes, where each super-class contains different viewpoints of the same texture process. Experiments are conducted on subsets of these 9 super-classes: water vs fountain (UCLA9wf), fountain vs waterfall (UCLA9wff), 4 classes (UCLA9c4), and 8 classes (UCLA9c8). The original *uncropped* videos are used.
- **UCLA7**: [39] also groups similar classes from UCLA into 7-super classes, using the *uncropped* video.
- **DynTex35**: [40] uses the old DynTex dataset[2], consisting of 35 sequences. Each sequence is decomposed into 10 subvideos, by splitting spatially and temporally, resulting in 35 classes, with 10 videos per class. Example frames from each class in DynTex35 are presented in Figure 8.

In this paper, we validate our proposed HEM-trained BoS on each of these datasets, following the protocols established by their respective papers and comparing to their published results.

2. This is an old version of the DynTex dataset used in the previous section.

TABLE 3: Distances and kernels used for classification.

square-root distance	(SR)	$d_s(w_1, w_2) = \arccos(\sum_k \sqrt{w_{1k} w_{2k}})$
χ^2 distance	(CS)	$d_{\chi^2}(w_1, w_2) = \frac{1}{2} \sum_k \frac{\lvert w_{1k} - w_{2k} \rvert}{w_{1k} + w_{2k}}$
χ^2 kernel	(CSK)	$K(w_1, w_2) = 1 - \sum_k \frac{(w_{1k} - w_{2k})^2}{\frac{1}{2}(w_{1k} + w_{2k})}$
Exponentiated χ^2 kernel	(ECS)	$K(w_1, w_2) = \exp(-\gamma d_{\chi^2}(w_1, w_2))$
Bhattacharyya kernel	(BCK)	$K(w_1, w_2) = \sum_k \sqrt{w_{1k} w_{2k}}$

5.4.3 Experimental setup

For the UCLA-based datasets, overlapping spatio-temporal cubes with size $5 \times 5 \times 75$ (step: $2 \times 2 \times 75$) pixels are extracted densely from the grayscale video. For the DynTex35 dataset, the videos are converted to grayscale, and overlapping spatio-temporal cubes with size $7 \times 7 \times 50$ (step: $5 \times 5 \times 30$) pixels are extracted. We ignore patches without significant motion by only selecting patches with overall pixel variance > 1. For all datasets, we learn video-level DTMs with $K = 4$ components. The BoS codebook is then learned by running HEM with $K = 64$ on the mixture formed from all the training video DTMs. For UCLA9, we also consider a codebook size of $K = 8$ in order to obtain a fair comparison with [18]. Each video is then represented by its TF and TFIDF vectors using the BoS codebook.

We mainly follow the protocol of [18] to train dynamic texture classifiers, using the various distances and kernel functions listed in Table 3 and the BoS representation. First, we use k-nearest neighbor classifiers using χ^2 and square-root distances, denoted as CS1 and SR1 for $k = 1$ and CS3 and SR3 for $k = 3$. Second, we consider support vector machines (SVM) using kernel functions related to the CS and SR distances, such as the χ^2 kernel (CSK), exponentiated χ^2 kernel (ECS), and Bhattacharyya kernel (BCK). SVM training and testing is performed with libSVM [41], with kernel and SVM parameters selected using 10-fold cross-validation on the training set. Finally, a generative classification approach, namely a naive Bayes (NB) classifier, was also tested, as in [18]. All classification results are averaged over a number of trials using different training and test sets, depending on the protocol of the dataset.

5.4.4 Classification results

Table 4 presents the video classification results for the various classifiers using the HEM-BoS codebook and either TF or TFIDF representations, and existing state-of-the-art reference methods for each dataset. Reference results (Ref) are those provided in the respective papers. The row labeled "Best" refers to the best accuracy among the various classifiers using the HEM-BoS codebook. First, looking at $K = 64$ codewords, the best classifier using the HEM-BoS codebook *consistently outperforms* the reference methods of [14, 39, 40, 18, 20]. To identify the best-performing (*i.e.*, most consistent) classifier, we rank all the HEM-BoS classifiers on each individual dataset, and then calculate the average ranking over the 5 datasets. The best ranking classifier is the 1-NN classifier using the square-root distance (TF-SR1). TF-SR1 is also consistently more accurate than the reference methods. These results demonstrate the efficacy of the HEM-BoS codebook for representing of a wide range of dynamic textures, while maintaining viewpoint and translation invariance.

Among the datasets, accuracy on UCLA39 is the most improved, from 20% [20] or 42.3% [39] to 56.4% for HEM-BoS. In contrast to [20], which is based solely on motion dynamics, and [39], which models local appearance and instantaneous motion, the BoS representation is able to leverage both the local appearance (for translation invariance) and motion dynamics of the video to improve the overall accuracy.

Next, we compare the two methods of learning a BoS codebook, the HEM algorithm and NLDR/clustering [18], using $K = 8$ as in [18]. On both the 4- and 8-class UCLA9 datasets, the accuracy using HEM-BoS improves significantly over NLDR-BoS, from 89% to 97.92% on the 4-class problem, and from 80% [18] or 84% [38] to 92.83% on the 8-class problem[3]. The improvement in performance is due to both the

3. Using the same patch sizes as in [38], we get similar performances on the 8-class problem: 92.83% accuracy for $20 \times 20 \times 25$ patches, and 90.10% for $30 \times 30 \times 25$.

TABLE 4: BoS Classification Results. Average Rank is calculated from the individual ranks on each dataset for $K = 64$ (shown in parenthesis). [A/B] refers to "method A as reported in B".

Method			K=8				K=64					Average Rank
			UCLA9wf	UCLA9wff	UCLA9c4	UCLA9c8	UCLA9c8	UCLA7	UCLA39	UCLA50	DynTex35	
TF		*CS1*	99.17	**98.75**	97.08	91.20	95.87 (9.0)	98.75 (8.5)	46.79 (12.0)	92.30 (10.0)	97.56 (3.0)	8.5
	N	*CS3*	98.75	97.50	92.50	85.43	90.65 (13.0)	97.75 (13.0)	46.47 (13.0)	86.00 (15.0)	95.12 (12.0)	13.2
	N	*SR1*	**100.00**	98.12	96.46	92.28	96.63 (5.0)	99.00 (6.0)	52.88 (8.5)	96.45 (2.0)	97.99 (2.0)	**4.7**
		SR3	98.75	95.62	93.33	83.37	89.89 (14.0)	98.50 (11.0)	52.88 (8.5)	89.15 (12.0)	96.27 (9.0)	10.9
	S	*CSK*	**100.00**	97.19	92.50	86.41	97.28 (4.0)	99.00 (6.0)	52.88 (8.5)	93.90 (7.0)	97.29 (4.0)	5.9
	V	*INT*	**100.00**	96.56	92.29	75.54	95.22 (10.0)	99.50 (3.0)	57.69 (2.0)	**96.55** (1.0)	95.71 (10.0)	5.2
	M	*BCK*	99.58	95.31	93.12	70.87	**97.39** (1.5)	**99.75** (1.5)	52.88 (8.5)	94.40 (5.5)	93.84 (14.0)	6.2
TFIDF		*CS1*	99.17	96.56	97.71	92.07	96.09 (7.0)	98.75 (8.5)	45.83 (14.0)	91.70 (11.0)	96.84 (5.0)	9.1
	N	*CS3*	99.17	96.25	93.33	85.98	90.98 (12.0)	97.25 (14.0)	44.55 (15.0)	86.30 (14.0)	95.56 (11.0)	13.2
	N	*SR1*	99.17	95.94	**97.92**	**92.83**	96.52 (6.0)	98.50 (11.0)	56.09 (4.0)	95.60 (4.0)	**98.57** (1.0)	5.2
		SR3	99.17	95.00	95.42	85.00	89.46 (15.0)	98.50 (11.0)	56.41 (3.0)	89.10 (13.0)	96.70 (6.0)	9.6
	S	*CSK*	97.50	93.12	93.33	83.91	**97.39** (1.5)	99.00 (6.0)	53.53 (6.0)	93.10 (8.5)	96.42 (8.0)	6.0
	V	*INT*	97.50	87.50	87.08	69.78	94.89 (11.0)	99.25 (4.0)	**58.01** (1.0)	96.00 (3.0)	94.84 (13.0)	6.4
	M	*BCK*	97.50	87.19	90.21	67.39	97.28 (3.0)	**99.75** (1.5)	55.13 (5.0)	93.10 (8.5)	92.97 (15.0)	6.6
NB			**100.00**	96.56	93.54	79.78	96.09 (8.0)	94.75 (15.0)	50.96 (11.0)	94.40 (5.5)	96.56 (7.0)	9.3
BEST			**100**	**98.75**	**97.92**	**92.83**	**97.39**	**99.75**	**58.01**	**96.55**	**98.57**	***TFSR1***
Ref			**100** [18]	98 [18]	89 [18]	80 [18]		92.30 [39]	42.3 [39]	96 [14]	97.14 [40]	
			100 [13/18]	56.25 [13/18]	83 [13/18]	52.27 [13/18], 84 [38]			20 [20],15 [14/20]	89 [13/14], 81 [39]		

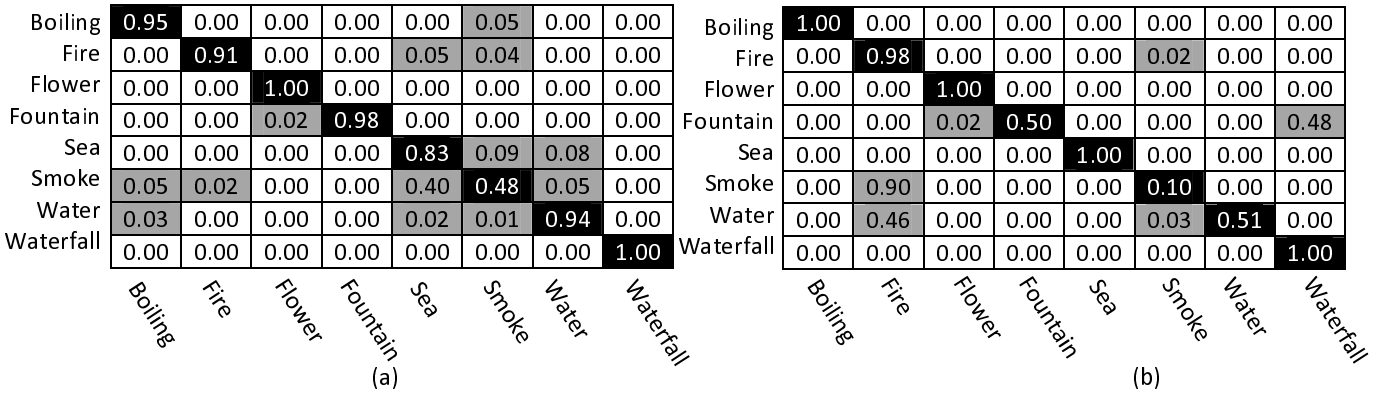

Fig. 9: Confusion matrix on UCLA9c8 using: (a) HEM-trained BoS (b) BoS from [18].

generation of novel DT codewords, and the ability to learn these codewords efficiently from more data, *i.e.*, from a dense sampling of spatio-temporal cubes, rather than those selected by interest point operators. Figure 9 shows the confusion matrix for UCLA9c8, using the HEM-BoS and the NLDR-BoS from [18], respectively. HEM-BoS removes the misclassifications of "water" to "fire", and "fountain" to "waterfall". Again, this illustrates the robustness of the BoS learned with HEM. Figure 10 shows several examples of test videos with the generated class labels. The average runtime was 1.5 hours to learn a codebook from video-level DTMs for UCLA39, and 20 seconds to calculate the BoS representation for a single video.

Finally, we investigate the effect of increasing the codebook size for the BoS representation. Figure 11 plots the accuracy on UCLA$\{7, 39, 50\}$ and DynTex35, versus a codebook size of $K = \{8, 16, 32, 64\}$. In general, increasing the number of codewords also increases the classifier accuracy, with accuracy saturating for UCLA50 and UCLA7. Also, increasing the codebook size increases the computational cost of projecting to the codebook. A codebook size of $K = 64$ represents a good tradeoff between speed and performance for BoS classification on these datasets.

6 Conclusions

In this paper, we have derived a hierarchical EM algorithm that both clusters DTs and learns novel DT cluster centers that are representative of the cluster members, in a manner that is consistent with the underlying probabilistic models. The clustering is achieved by generating "virtual" samples from the input DTs, and maximizing the log-likelihood of these virtual samples with respect to the DT cluster centers. Using the law-of-large-numbers, the sum over virtual samples can be replaced by an expectation over the input DTs, resulting in a clustering algorithm that depends only on the input DT model parameters. For the E-step inference of HEM, we also derive a novel efficient algorithm for sensitivity analysis of the Kalman smoothing filter. Besides clustering, the HEM algorithm for DTs can also be used for hierarchical model estimation from large datasets, where DTMs are first learned on subsets of the data (*e.g.*, individual videos), and the resulting DTMs are then aggregated using the HEM algorithm. This formulation provides a significant increase in computational and memory efficiency, in comparison to running standard EM on the full dataset.

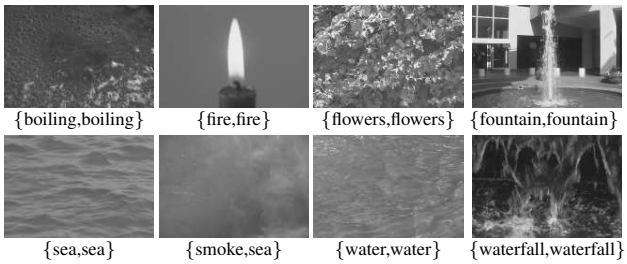

Fig. 10: Classification examples from UCLA9c8 {ground truth, classifier prediction}.

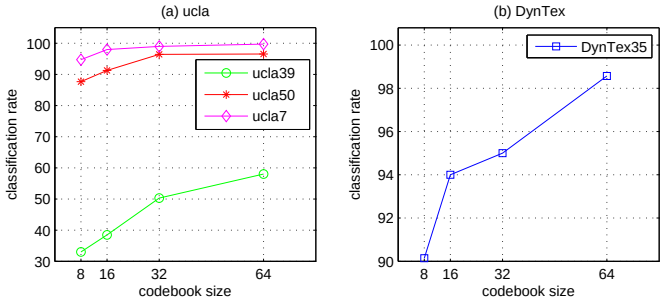

Fig. 11: Effect of increasing the codebook size on BoS classification using different data sets.

We apply the HEM algorithm to a variety of motion analysis problems. First, we apply HEM to hierarchically cluster video textures, and demonstrate that the algorithm produces consistent clusters based on video motion. Second, we use HEM to estimate motion annotation models using the SML framework, where each annotation model is a DTM learned with weakly-labeled training data. Third, we use HEM to learn BoS codebooks and demonstrate state-of-the-art results in dynamic texture recognition. Future work will be directed at extending HEM to general graphical models, allowing a wide variety of generative models to be clustered or used as codewords in a bag-of-X representation. Finally, in this work we have not addressed the model selection problem, i.e., selecting the number of reduced mixture components. Since HEM is based on maximum likelihood principles, it is possible to apply standard statistical model selection approaches, such as Akaike information criterion (AIC) and Bayesian information criterion (BIC) [42]. Alternatively, inspired by Bayesian non-parametric statistics, the HEM formulation could be extended to include a Dirichlet process prior [43], with the number of components adapting to the data.

ACKNOWLEDGMENTS

The authors thank R. Péteri for the DynTex dataset, and G. Doretto for the UCLA dataset. AM and ABC were supported by the Research Grants Council of the Hong Kong Special Administrative Region, China [CityU 110610]. EC and GRGL acknowledge support from Qualcomm, Inc., Yahoo! Inc., the Hellman Fellowship Program, the Alfred P. Sloan Foundation, NSF Grants CCF-0830535 and IIS-1054960, and the UCSD FWGrid Project, NSF Research Infrastructure Grant Number EIA-0303622. ABC, EC and GRGL also received support from a Google Research Award.

REFERENCES

[1] B. Horn and B. Schunk, "Determining optical flow," *Artificial Intelligence*, vol. 17, pp. 185–204, 1981.
[2] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. DARPA Image Understanding Workshop*, 1981, pp. 121–130.
[3] G. Doretto, A. Chiuso, Y. N. Wu, and S. Soatto, "Dynamic textures," *Intl. J. Computer Vision*, vol. 51, no. 2, pp. 91–109, 2003.
[4] A. W. Fitzgibbon, "Stochastic rigidity: image registration for nowhere-static scenes," in *ICCV*, vol. 1, 2001, pp. 662–70.
[5] A. Ravichandran and R. Vidal, "Dynamic texture registration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009.
[6] G. Doretto, D. Cremers, P. Favaro, and S. Soatto, "Dynamic texture segmentation," in *ICCV*, vol. 2, 2003, pp. 1236–42.
[7] A. Ghoreyshi and R. Vidal, "Segmenting dynamic textures with Ising descriptors, ARX models and level sets," in *Dynamical Vision Workshop in the European Conf. on Computer Vision*, 2006.
[8] A. B. Chan and N. Vasconcelos, "Modeling, clustering, and segmenting video with mixtures of dynamic textures," *IEEE TPAMI*, vol. 30, no. 5, pp. 909–926, May 2008.
[9] R. Vidal and A. Ravichandran, "Optical flow estimation & segmentation of multiple moving dynamic textures," in *IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2, 2005, pp. 516–21.
[10] A. B. Chan and N. Vasconcelos, "Layered dynamic textures," *IEEE Trans. on Pattern Analysis and Machine Intelligence: Special Issue on Probabilistic Graphical Models in Computer Vision*, vol. 31, no. 10, pp. 1862–1879, October 2009.
[11] R. Chaudry, A. Ravichandran, G. Hager, and R. Vidal, "Histograms of oriented optical flow and Binet-Cauchy kernels on nonlinear dynamical systems for the recognition of human actions," in *CVPR*, 2009.
[12] A. Bissacco, A. Chiuso, and S. Soatto, "Classification and recognition of dynamical models: The role of phase, independent components, kernels and optimal transport." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, pp. 1958–1972, 2007.
[13] P. Saisan, G. Doretto, Y. Wu, and S. Soatto, "Dynamic texture recognition," in *CVPR*, vol. 2, 2001, pp. 58–63.
[14] A. B. Chan and N. Vasconcelos, "Probabilistic kernels for the classification of auto-regressive visual processes," in *CVPR*, vol. 1, 2005, pp. 846–851.
[15] 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," *Intl. J. Computer Vision*, vol. 73, no. 1, pp. 95–119, 2007.
[16] R. Vidal and P. Favaro, "Dynamicboost: Boosting time series generated by dynamical systems," in *IEEE Intl. Conf. on Computer Vision*, 2007.
[17] A. B. Chan and N. Vasconcelos, "Classifying video with kernel dynamic textures," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2007.
[18] A. Ravichandran, R. Chaudhry, and R. Vidal, "View-invariant dynamic texture recognition using a bag of dynamical systems," in *CVPR*, 2009.
[19] B. Ghanem and N. Ahuja, "Phase based modelling of dynamic textures," in *IEEE Intl. Conf. on Computer Vision*, 2007.
[20] F. Woolfe and A. Fitzgibbon, "Shift-invariant dynamic texture recognition," in *ECCV*, 2006.
[21] H. Cetingul and R. Vidal, "Intrinsic mean shift for clustering on Stiefel and Grassmann manifolds," in *CVPR*, 2009.
[22] A. Goh and R. Vidal, "Clustering and dimensionality reduction on Riemannian manifolds," in *CVPR*, 2008.
[23] N. Vasconcelos and A. Lippman, "Learning mixture hierarchies," in *Neural Information Processing Systems*, 1998.
[24] A. B. Chan, E. Coviello, and G. Lanckriet, "Clustering dynamic textures with the hierarchical EM algorithm," in *Intl. Conference on Computer Vision and Pattern Recognition*, 2010.
[25] G. Carneiro, A. B. Chan, P. J. Moreno, and N. Vasconcelos, "Supervised learning of semantic classes for image annotation and retrieval," *IEEE TPAMI*, vol. 29, no. 3, pp. 394–410, March 2007.
[26] A. Gelb, *Applied Optimal Estimation*. MIT Press, 1974.
[27] N. Vasconcelos, "Image indexing with mixture hierarchies," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2001.
[28] A. Banerjee, S. Merugu, I. Dhillon, and J. Ghosh, "Clustering with bregman divergences," *Journal of Machine Learning Research (JMLR)*, vol. 6, pp. 1705–1749, 2005.
[29] J. V. Davis and I. Dhillon, "Differential entropic clustering of multivariate gaussians," in *Adv. in Neural Inf. Proc. Sys. (NIPS*, 2006.
[30] J. Goldberger and S. Roweis, "Hierarchical clustering of a mixture model," in *In NIPS*. MIT Press, 2005, pp. 505–512.
[31] R. E. Griffin and A. P. Sage, "Sensitivity analysis of discrete filtering and smoothing algorithms," *AIAA Journal*, vol. 7, pp. 1890–1897, Oct. 1969.
[32] J. Wall, A. Willsky, and N. Sandell, "On the fixed-interval smoothing problem." *Stochastics.*, vol. 5, pp. 1–41, 1981.
[33] E. Coviello, A. Chan, and G. Lanckriet, "Time series models for semantic music annotation," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 19, no. 5, pp. 1343 –1359, july 2011.
[34] R. H. Shumway and D. S. Stoffer, "An approach to time series smoothing and forecasting using the EM algorithm," *Journal of Time Series Analysis*, vol. 3, no. 4, pp. 253–264, 1982.
[35] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society B*, vol. 39, pp. 1–38, 1977.
[36] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice-Hall, 1993.
[37] R. Péteri, S. Fazekas, and M. J. Huiskes, "DynTex: A comprehensive database of dynamic textures," *Pattern Recognition Letters*, vol. 31, no. 12, pp. 1627–32, 2010. [Online]. Available: http://www.cwi.nl/projects/dyntex
[38] A. Ravichandran, R. Chaudhry, and R. Vidal, "Categorizing dynamic textures using a bag of dynamical systems," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PP, no. 99, p. 1, 2012.
[39] K. G. Derpanis and R. P. Wildes, "Dynamic texture recognition based on distributions of spacetime oriented structure," in *CVPR*, 2010.
[40] G. Zhao and M. Pietikainen, "Dynamic texture recognition using local binary patterns with an application to facial expressions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007.
[41] C. Chang and C. Lin, "Libsvm: a library for support vector machines," *ACM TIST*, 2011.
[42] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference and prediction*, 2nd ed. Springer, 2009. [Online]. Available: http://www-stat.stanford.edu/ tibs/ElemStatLearn/
[43] D. M. Blei and M. I. Jordan, "Variational inference for dirichlet process mixtures," *Bayesian Analysis*, vol. 1, pp. 121–144, 2005.

Adeel Mumtaz received the B.S. degree in computer science from Pakistan Institute of Engineering and Applied Sciences and the M.S. degree in computer system engineering from Ghulam Ishaq Khan Institute of Engineering Sciences and Technology, Pakistan, in 2004 and 2006, respectively. He is currently working toward the PhD degree in Computer Science at the City University of Hong Kong. He is currently with the Video, Image, and Sound Analysis Laboratory , Department of Computer Science, CityU. His research interests include Computer Vision, Machine Learning and Pattern recognition.

Emanuele Coviello received the "Laurea Triennale" degree in information engineering and the "Laurea Specialistica" degree in telecommunication engineering from the Università degli Studi di Padova, Padova, Italy, in 2006 and 2008, respectively. He is currently pursuing the Ph.D. degree in the Department of Electrical and Computer Engineering, University of California at San Diego (UCSD), La Jolla, where he has joined the Computer Audition Laboratory. Mr. Coviello received the "Premio Guglielmo Marconi Junior 2009" award, from the Guglielmo Marconi Foundation (Italy), and won the "2010 Yahoo! Key Scientific Challenge Program," sponsored by Yahoo!. His main interest is machine learning applied to content based information retrieval and multimedia data modeling, and automatic information extraction from the Internet.

Gert Lanckriet received the M.S. degree in electrical engineering from the Katholieke Universiteit Leuven, Leuven, Belgium, in 2000 and the M.S. and Ph.D. degrees in electrical engineering and computer science from the University of California, Berkeley, in 2001 and 2005, respectively. In 2005, he joined the Department of Electrical and Computer Engineering, University of California, San Diego, where he heads the Computer Audition Laboratory. His research focuses on the interplay of convex optimization, machine learning, and signal processing, with applications in computer audition and music information retrieval. Prof. Lanckriet was awarded the SIAM Optimization Prize in 2008 and is the recipient of a Hellman Fellowship, an IBM Faculty Award, an NSF CAREER Award and an Alfred P. Sloan Foundation Research Fellowship. In 2011, MIT Technology Review named him one of the 35 top young technology innovators in the world (TR35).

Antoni B. Chan received the B.S. and M.Eng. degrees in electrical engineering from Cornell University, Ithaca, NY, in 2000 and 2001, respectively, and the Ph.D. degree in electrical and computer engineering from the University of California, San Diego (UCSD), San Diego, in 2008. From 2001 to 2003, he was a Visiting Scientist with the Vision and Image Analysis Laboratory, Cornell University, Ithaca, NY, and in 2009, he was a Postdoctoral Researcher with the Statistical Visual Computing Laboratory, UCSD. In 2009, he joined the Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong, as an Assistant Professor. His research interests include computer vision, machine learning, pattern recognition, and music analysis. Dr. Chan was the recipient of an NSF IGERT Fellowship from 2006 to 2008, and an Early Career Award in 2012 from the Research Grants Council of the Hong Kong SAR, China.