Categorizing Dynamic Textures Using a Bag of Dynamical Systems

Avinash Ravichandran, *Member*, *IEEE*, Rizwan Chaudhry, *Member*, *IEEE*, and René Vidal, *Senior Member*, *IEEE*

Abstract—We consider the problem of categorizing video sequences of dynamic textures, i.e., nonrigid dynamical objects such as fire, water, steam, flags, etc. This problem is extremely challenging because the shape and appearance of a dynamic texture continuously change as a function of time. State-of-the-art dynamic texture categorization methods have been successful at classifying videos taken from the same viewpoint and scale by using a Linear Dynamical System (LDS) to model each video, and using distances or kernels in the space of LDSs to classify the videos. However, these methods perform poorly when the video sequences are taken under a different viewpoint or scale. In this paper, we propose a novel dynamic texture categorization framework that can handle such changes. We model each video sequence with a collection of LDSs, each one describing a small spatiotemporal patch extracted from the video. This Bag-of-Systems (BoS) representation is analogous to the Bag-of-Features (BoF) representation for object recognition, except that we use LDSs as feature descriptors. This choice poses several technical challenges in adopting the traditional BoF approach. Most notably, the space of LDSs is not euclidean; hence, novel methods for clustering LDSs and computing codewords of LDSs need to be developed. We propose a framework that makes use of nonlinear dimensionality reduction and clustering techniques combined with the Martin distance for LDSs to tackle these issues. Our experiments compare the proposed BoS approach to existing dynamic texture categorization methods and show that it can be used for recognizing dynamic textures in challenging scenarios which could not be handled by existing methods.

Index Terms—Dynamic textures, categorization, linear dynamical systems

1 Introduction

Dynamic textures are video sequences of complex nonrigid dynamical objects such as fire, flames, water on the surface of a lake, a flag fluttering in the wind, etc. The development of algorithms for the analysis of such video sequences is important in several applications such as surveillance, where, for example, one wants to detect fires or pipe ruptures. However, the continuous change in the shape and appearance of a dynamic texture makes the application of traditional computer vision algorithms very challenging.

Over the years, several approaches have been proposed for modeling and synthesizing video sequences of dynamic textures [31], [28], [36], [1], [10], [19]. Among them, the generative model proposed by Doretto et al. [10], where a dynamic texture is modeled using a Linear Dynamical System (LDS), has been shown to be very versatile. The LDS-based model has been successfully used for various vision tasks such as synthesis, editing, segmentation, registration, and categorization. In this paper, we are primarily interested in the problem of categorization of dynamic textures. That is, given a video sequence of a single dynamic texture, we want to identify which class (e.g., water, fire, etc.) the video sequence belongs to.

Most of the existing dynamic texture categorization methods model the video sequence (or a manually selected image region) as the output of an LDS. Then, a distance or a kernel between the model parameters of two dynamical systems is defined. Once such a distance or kernel has been defined, classifiers such as Nearest Neighbors (NNs) or Support Vector Machines (SVMs) [12] can be used to categorize a query video sequence based on the training data. Among these methods, Saisan et al. [27] used distances based on the principal angles between the observability subspaces associated with the LDSs. Vishwanathan et al. [34] used Binet-Cauchy kernels to compare the parameters of two LDSs. Further, Chan and Vasconcelos used both the KL divergence [5] and the Martin distance [4] as a metric between dynamical systems. Finally, Woolfe and Fitzgibbon [39] used the family of Chernoff distances and distances between cepstrum coefficients as a metric between LDSs. Other types of approaches for dynamic texture categorization, such as Fujita and Nayar [13], divide the video sequences into blocks and compare the trajectories of the states in order to perform the inference. Alternatively, Vidal and Favaro [33] extended boosting to LDSs by using dynamical systems as weak classifiers.

However, existing approaches to dynamic texture categorization suffer from two major drawbacks.

1. Rather than using an LDS to model the whole video sequence, existing approaches are typically applied to a manually extracted "region of interest" containing the most dynamic content in the original video. In

- *A. Ravichandran is with the UCLA Vision Lab, University of California, Los Angeles, Boelter Hall # 3811A, 405 Hilgard Avenue, Los Angeles, CA 90095. E-mail: avinash@cs.ucla.edu.*
- *R. Chaudhry and R. Vidal are with the Center for Imaging Science, The Johns Hopkins University, 3400 N. Charles St., Baltimore, MD 21218. E-mail: rizwanch@cis.jhu.edu, rvidal@jhu.edu .*

Manuscript received 16 June 2011; revised 30 Jan. 2012; accepted 18 Mar. 2012; published online 2 Apr. 2012.
Recommended for acceptance by D. Cremers.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2011-06-0390.
Digital Object Identifier no. 10.1109/TPAMI.2012.83.

practice, automatically identifying a region of interest for each video sequence might not always be possible.

2. Existing approaches are unable to handle videos taken under different viewpoint and scale. Indeed, most existing approaches have been validated on the database from [27], where videos corresponding to the same semantic category, e.g., "water," but taken from different viewpoints, e.g., close or far, are assigned to different categories such as "water close" and "water far." Hence, the database is interpreted as having 50 classes, while in reality there are only nine different semantic categories. As a consequence, existing methods address a very specific scenario, which is not very useful for practical applications.

One way to address these drawbacks is to leverage recent works on image-based object categorization that can handle variations in viewpoint and scale. One such work is the Bag of Features (BoF) approach [30], [8], where an image is hypothesized to be identifiable by the distribution of certain key features extracted from the image. Specifically, a collection of feature descriptors is extracted from each image in the training set. These feature descriptors are quantized into a dictionary of visual words or codewords and the distribution of codewords is used to represent each image. New images are categorized by comparing their distribution of codewords to those of images in the training set using classifiers such as NNs or SVMs.

A simple way of extending the BoF approach to videos is to replace traditional image features by spatiotemporal video features. This requires methods for detecting points of interest in space and time as well as methods for describing a spatiotemporal volume around each of these points. Several feature descriptors [9], [20], [38], [37], [18] have been proposed in the existing literature. Laptev [20] extends the Harris corner detector used for images to video sequences by considering a second moment matrix of the spatiotemporal gradients. Willems et al. [37] use the spatiotemporal Hessian. Dollár et al. [9] convolve the video sequence spatially with a Gaussian filter and temporally with a quadrature pair of 1D Gabor filters. Local maxima of this response function are then selected as feature points. Wong and Cipolla [38] model the video sequence with an LDS and detect interest points using the parameters of the LDS, as opposed to directly from the video sequence. Kläser et al. [18] extend the Histogram of oriented Gradient (HoG)-based approach from 2D to 3D. A detailed comparison of these methods can be found in [37] and an empirical evaluation of these descriptors for action recognition in videos can be found in [35]. Once the spatiotemporal features have been extracted, the BoF approach for videos is identical to that for images.

We contend that simple extensions of the BoF approach to videos based on spatiotemporal feature descriptors will not be sufficient for dynamic texture categorization. Our rationale is that the aforementioned spatiotemporal feature descriptors are based exclusively on spatial and temporal image gradients. As a consequence, they fail to capture a critical property of a dynamic texture: their dynamics. Although Wong and Cipolla [38] used the LDS representation to detect interest points in video sequences, they described the video sequences using the DOLLAR [9] features, which do not explicitly capture the dynamics.

In this paper, we propose a Bag-of-Systems (BoSs) approach to categorization of dynamic textures which extends the BoF approach from images to videos.

Our first contribution is to replace traditional spatiotemporal feature descriptors by LDSs. This choice explicitly models the temporal evolution of the intensities of the spatiotemporal patch it describes as opposed to implicitly modeling them using spatiotemporal gradients. Indeed, local LDS models have been successfully used for dynamic texture segmentation in [11] and [14]. However, they have not been used for dynamic texture categorization. The reason for this in the context of BoF methods is that the parameters of an LDS live in a non-euclidean space and this makes traditional methods for forming a codebook based on clustering euclidean feature descriptors no longer applicable.

Our second contribution is to propose two different methods for computing a codebook of LDSs. The first method uses a combination of nonlinear dimensionality reduction followed by K-Means clustering in the low-dimensional space. The second method uses the K-Medoid algorithm applied to the Martin distances among the LDSs. Given a codebook of LDSs, we use various histogram schemes for representing each video sequence and several classification techniques for categorizing novel sequences. We compare the BoS approach to traditional dynamic texture classification algorithms and to BoF approaches based on spatiotemporal features. Our experiments show that our BoS approach outperforms existing methods and that it can handle videos of dynamic textures taken under different viewpoint and scale conditions.

A simpler version of the proposed BoS approach was first introduced in [24], where we used an interest point detector to extract patches and a K-Means approach to compute codewords. In this paper, we use a dense sampling approach to extract interest points and show that this improves the performance of BoS. We further extend the method of forming codebooks by considering two alternative approaches for clustering the data. We also present numerous new experiments that evaluate the performance of BoS as a function of the patch and codebook size. Finally, we compare BoS against numerous state of the art methods and show that our proposed method outperforms them.

2 Preliminaries

In this section, we introduce the necessary concepts that are required for our categorization framework. We first introduce the model for dynamic textures and show how video sequences can be modeled using LDSs. We then introduce the Martin distance between two LDSs. This distance will serve as our (dis)similarity metric for comparing LDS in the BoS framework.

2.1 Dynamic Texture Model

Given F frames of a video or a spatiotemporal patch of p pixels, $\{I(t) \in \mathbb{R}^p\}_{t=1}^{F}$, we model the pixel intensities of each frame, $I(t)$, as the output of an LDS, i.e.,

$$\mathbf{z}(t+1) = A\mathbf{z}(t) + B\mathbf{v}(t), \tag{1}$$

$$I(t) = C^0 + C\mathbf{z}(t) + \mathbf{w}(t), \tag{2}$$

where $\mathbf{z}(t) \in \mathbb{R}^n$ is the hidden state at time t, $A \in \mathbb{R}^{n \times n}$ models the dynamics of the hidden state, $C \in \mathbb{R}^{p \times n}$ maps the hidden state to the output of the system, $C^0 \in \mathbb{R}^p$ is the mean of the video sequence, and $\mathbf{w}(t) \sim \mathcal{N}(0, R)$ and $B\mathbf{v}(t) \sim \mathcal{N}(0, Q)$ are the measurement and process noise, respectively. As is often the case, the transformation $Q = BB^\top$ is used to incorporate the B matrix in the noise covariance and $B\mathbf{v}_t$ is replaced by $\mathbf{v}'_t$. The dimension of the hidden state, n, is the order of the system and p is the number of pixels in one frame of the sequence or patch.

Given a video sequence, learning the parameters of this dynamical model, i.e., learning (C^0, A, C, Q, R) from $\{I(t)\}_{t=1}^F$, is called system identification and it is a well-studied problem in the controls community. Several optimal methods such as N4SID [23] and EM [29] can be used to learn these parameters. However, due to the high dimensionality of the output (number of pixels), these procedures become computationally intractable.

To overcome this difficulty, Doretto et al. [10] introduced a fast but suboptimal method for computing the system parameters using a Principal Component Analysis (PCA)-based approach. In this approach, given the frames of the video sequence, $\{I(t)\}_{t=1}^F$, a compact, rank n, singular value decomposition of the matrix

$$Y = [I(1) - C^0, \ldots, I(F) - C^0] = U\Sigma V^\top, \tag{3}$$

is performed where

$$C^0 = \frac{1}{N}\sum_{t=1}^{F} I(t).$$

The system parameters and the state parameters are then estimated as

$$C = U \quad \text{and} \quad Z = \Sigma V^\top, \tag{4}$$

where $Z = [\mathbf{z}(1), \mathbf{z}(2), \ldots, \mathbf{z}(F)]$ are the estimated states of the system. Notice that in computing Z, the state (1) is not enforced. This is what makes this method suboptimal. Given the state sequence, the matrix A can be computed using least-squares as

$$A = [\mathbf{z}(2), \mathbf{z}(3), \ldots, \mathbf{z}(F)][\mathbf{z}(1), \mathbf{z}(2), \ldots, \mathbf{z}(F-1)]^\dagger, \tag{5}$$

where $Z^\dagger$ represents the pseudoinverse of Z. Also,

$$Q = \frac{1}{N-1}\sum_{t=1}^{F-1} \mathbf{v}'_t (\mathbf{v}'_t)^\top, \tag{6}$$

where $\mathbf{v}'_t = B\mathbf{v}(t) = \mathbf{z}(t+1) - A\mathbf{z}(t)$.

The advantage of this model is that it decouples the appearance of the spatiotemporal patch, which is modeled by C, from the dynamics, which are represented by A. Hence, given a spatiotemporal patch, we describe it using the tuple $M = (A, C)$. Such a feature descriptor models both the dynamics and the appearance in the spatiotemporal patch as opposed to image gradients that only model local texture.

2.2 Subspace Angles-Based Distance between LDSs

In any categorization algorithm, features from a new query video sequence need to be compared with the features from the training set. In general these features lie in a euclidean space and the euclidean distance between the features is used as a metric for comparison. In our case, however, the feature descriptors are LDSs. Therefore, given two LDSs, $M_1 = (A_1, C_1)$ and $M_2 = (A_2, C_2)$, we need to define a notion of similarity between these two descriptors.

One family of distances between two LDSs is based on the *subspace angles* between the two systems. The subspace angles are defined as the principal angles between the *observability subspaces* associated with the two model parameters. The observability subspace is the range-space of the extended observability matrix of the LDS defined by

$$\mathcal{O}_\infty(M) = [C^\top, (CA)^\top, (CA^2)^\top, \ldots]^\top \in \mathbb{R}^{\infty \times n}. \tag{7}$$

The calculation of the subspace angles between the two models is performed by first solving for $\mathcal{P}$ from the Lyapunov equation $\mathcal{A}^\top \mathcal{P}\mathcal{A} - \mathcal{P} = -\mathcal{C}^\top \mathcal{C}$, where

$$\mathcal{P} = \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} \in \mathbb{R}^{2n \times 2n}, \tag{8}$$

$$\mathcal{A} = \begin{bmatrix} A_1 & \mathbf{0} \\ \mathbf{0} & A_2 \end{bmatrix} \in \mathbb{R}^{2n \times 2n}, \tag{9}$$

$$\mathcal{C} = [\, C_1 \quad C_2 \,] \in \mathbb{R}^{p \times 2n}. \tag{10}$$

The cosine of the subspace angles $\{\theta_i\}_{i=1}^n$ is calculated as

$$\cos^2 \theta_i = i\text{th eigenvalue}(P_{11}^{-1} P_{12} P_{22}^{-1} P_{21}). \tag{11}$$

Using the subspace angles, the Martin distance between M_1 and M_2 is defined as

$$d_M(M_1, M_2)^2 = -\ln \prod_{i=1}^{n} \cos^2 \theta_i. \tag{12}$$

The Martin distance can be easily extended to deal with two systems of different orders n_1 and n_2 by solving for $\mathcal{P} \in \mathbb{R}^{(n_1+n_2)\times(n_1+n_2)}$ from $\mathcal{A} \in \mathbb{R}^{(n_1+n_2)\times(n_1+n_2)}$ and $\mathcal{C} \in \mathbb{R}^{p \times (n_1+n_2)}$. However, one limitation of the Martin distance is that the number of pixels p for both of the models must be the same. This limitation can be overcome by simply resampling the images to be the same size. For more details regarding subspace angle-based distances between LDSs, we refer the reader to [6].

3 Categorizing Video Sequences Using a Bag of Dynamical Systems

Our approach to solving the dynamic texture categorization problem is inspired by the BoF approach for categorizing images. The typical steps followed in the BoF framework are the following:

1. Features and their corresponding descriptors are extracted from all the images in the training set.
2. A codebook is formed using clustering methods such as K-Means, where the cluster centers represent codewords in the codebook.
3. Each image in the training set is represented by the distribution of codewords in the image.

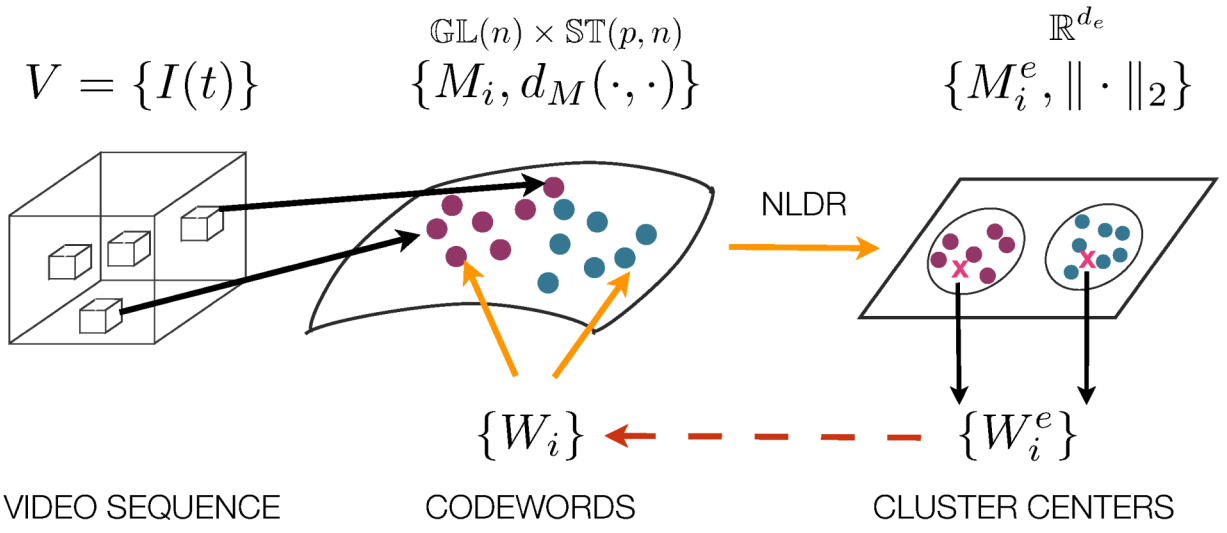

Fig. 1. Outline of the bag-of-systems framework for dynamic texture recognition.

4. A classifier is chosen to compare the distribution of codewords of a new query image to the distribution of codewords of the images in the training set and thus infer its category.

In this paper, we propose to follow a similar approach for categorizing video sequences of dynamic textures. Our key contributions are in the first two steps of the BoF approach, where we replace traditional spatiotemporal features by LDSs and introduce a novel method for computing a codebook of LDSs. This last step is particularly challenging because it requires clustering in the space of LDSs. An overview of our framework is shown in Fig. 1.

In the following sections, we will describe the corresponding BoS analogues to each one of the BoF steps. While we will restrict our description of the BoS analogues corresponding to the most popular methods of each step in the BoF framework, notice that one could replace each step with any alternative approach using the concepts we will introduce here. For example, in the clustering of LDSs, step 1 could use a hierarchical clustering scheme as outlined in [22] or use a different representation and classification scheme such as Latent Dirichlet Allocation [3] or probabilistic latent semantic analysis [15].

3.1 Feature Extraction and Descriptors

The first step in our framework, and in fact in any BoF approach, is to extract feature descriptors from the video sequences. Toward this end, there exist two popular methods: the interest point approach and the dense sampling approach. In the *interest point approach*, certain pixels of the entire video sequence are selected as "interesting" whenever they satisfy a particular criterion employed. For example, the 3D Harris detector looks for points that are like corners in space and time, i.e., points whose spatiotemporal derivatives have large magnitudes. Once the feature points are detected, the salient properties such as intensity and optical flow and their gradients of a neighborhood around each of these points are described using a descriptor. In the *dense sampling approach*, the feature detection step is skipped. Instead, the given video is divided into regular sized spatiotemporal volumes with the possibility that these volumes could overlap. Each of these volumes is then described using a feature descriptor.

In our framework, we will employ the dense sampling approach. There are several reasons for this choice. First, interest point approaches have been shown to be very useful for describing points of an object that are relevant for recognition (see, e.g., [8]). As shown in [9], [20], [38], [37], and [18], this is also the case for recognition of activities in videos, where spatiotemporal points represent points on the human body. In our case, however, we are describing the motion of objects such as water, where every point is as interesting as any other point. Consequently, we would like to use this information and not ignore certain regions in the video sequence. Second, there is support from existing literature on both image categorization [21] and activity recognition from videos [35] that the dense sampling approach works better than the interest point approach.

The main difference between our approach and the standard BoF approach for videos lies in our choice of feature descriptors. More specifically, we describe each spatiotemporal patch using an LDS, as opposed to using existing spatiotemporal descriptors. In this way, we can explicitly model both the appearance and the dynamics of each spatiotemporal patch. As our experiments in Section 4 will show, using LDS feature descriptors almost always improves the categorization performance with respect to using regular spatiotemporal descriptors. In the next section, we will describe how we form codebooks for LDSs in order to apply the BoF framework.

3.2 Codebook Formation

In the traditional BoF framework, once feature points and their corresponding descriptors are extracted, the descriptors are clustered using an algorithm such as K-Means to form a codebook. In our case the descriptors are the parameters of LDSs, which lie in a non-euclidean manifold. Hence, in order to cluster the LDSs to form codewords, we need to generalize K-Means to the non-euclidean space.

In order to perform clustering on the space of dynamical systems, we would first need to define a Riemannian metric. The LDS parameters tuple (A, C) belongs to a subset of the

outer-product space $GL(n) \times ST(p,n)$, where $GL(n)$ is the group of all invertible matrices of size n and $ST(p,n)$ is the Stiefel manifold of $p \times n$ matrices with orthonormal columns ($p \geq n$). This subset is characterized by several other constraints that the tuple (A, C) needs to satisfy. In particular, A must be a stable matrix, i.e., $|\lambda_i(A)| < 1$, where $\lambda_i(A)$ is the ith eigenvalue of A. In addition, the system must be observable. Specifically, the rank of the observability matrix in (7) must be equal to n. Furthermore, any Riemannian metric for the space of LDS must be invariant to a change of basis of the state space. Specifically, if we apply a linear transformation $P \in GL(n)$ to the state $\mathbf{z}_t$, the dynamical system parameters (A, C) are transformed to (PAP^{-1}, CP^{-1}). Therefore, the dynamical systems with parameters (A, C) and (PAP^{-1}, CP^{-1}) are equivalent and the distance between them must be zero. All these constraints make it very difficult to define a Riemannian metric on the space of LDS that has a closed-form expression. In fact, finding a Riemannian metric for the space of LDS is an open problem. Moreover, quantities such as the mean on a manifold do not always exist or are not always uniquely defined, and are often calculated using an iterative process.

To get around the difficulty associated with finding a Riemannian metric for the space of LDS, there have been some attempts to develop metrics between the space of all the output sequences generated by two dynamical systems. The most commonly used distances between LDS consider the distances between the respective extended observability subspaces, i.e., the span of the columns of the infinite matrix $\mathcal{O}_\infty(M) = [C^\top, (CA)^\top, (CA^2)^\top, \ldots]^\top \in \mathbb{R}^{\infty \times n}$. One could perform clustering on the manifold of *finite* observability subspaces by taking a fixed number of rows of $\mathcal{O}$ as an approximation to clustering on the infinite dimensional manifold. However, such an approach is computationally very expensive.

In order to address the aforementioned issues, we propose two different methods in the following section.

3.2.1 Dimensionality Reduction + K-Means

In our first approach, we exploit the fact that the space of LDSs is endowed with several distances, e.g., the Martin distance described in Section 2.2. We use these distances to find a euclidean embedding of the LDSs into a low-dimensional linear subspace. Several nonlinear dimensionality reduction techniques, such as Locally Linear Embedding (LLE) [26], work directly with the points in the higher dimensional space, while other methods, such as Multi-dimensional Scaling (MDS) [7] and Isomap [32], work with the pairwise distances between the points in the higher dimensional space. Given a low-dimensional representation, we apply traditional euclidean clustering algorithms to the euclidean representation in order to form the codebook.

More specifically, given the set of LDS features $\{M_i\}_{i=1}^T$, where T represents the total number of features extracted from the N videos in the training set, we first form the matrix $D \in \mathbb{R}^{T \times T}$ such that $D_{ij} = d_M(M_i, M_j)$. Once all pairwise distances between the LDSs are available, we use MDS to obtain a low-dimensional embedding of these points $\{M_i^e \in \mathbb{R}^{d_e}\}_{i=1}^T$, where d_e is the dimension of the embedding. This low-dimensional representation gives us a set of euclidean points, which approximately preserve the distance relationships in the high-dimensional noneuclidean space. Then, clustering algorithms such as K-Means can be applied to $\{M_i^e\}_{i=1}^T$ because the low-dimensional space is euclidean.

After the clustering stage, we have K cluster centers $\{W_i^e\}_{i=1}^K$. However, these cluster centers do not correspond to any of the original LDSs. Moreover, as a result of MDS, there exists no explicit map from the lower dimensional embedding to the original space. Hence, in order to select LDS codewords, $\{W_i\}_{i=1}^K$, we choose the LDS in the training set whose corresponding low-dimensional representation is closest to the cluster center in the low dimensional space, i.e.,

$$W_i = M_p, \quad \text{where } p = \arg\min_j \|M_j^e - W_i^e\|^2. \tag{13}$$

3.2.2 K-Medoid

Our second approach to form the codebook is to use the distance matrix D directly instead of embedding the points into the euclidean space. One such algorithm that works with the distance between the points rather than the points themselves is the K-Medoid algorithm [17]. Similar to the K-Means algorithm, the K-Medoid algorithm starts by selecting a subset of data points as cluster centers. The remaining points are then grouped based on their distances to these chosen cluster centers. Given the clustering, the cluster centers are updated as the medoid of the data points within each group. A medoid of a set of data points is the data point that minimizes the sum of squared distances to all other data points. Therefore, the medoid is similar to the centroid of the data points, except that it is restricted to be one of the data points. Consequently, after running the K-Medoid algorithm on the distance matrix D, we can directly obtain a codebook whose elements are LDSs. This method is computationally more efficient than the K-Means approach we proposed.

Using either of the two methods, we can obtain the codebook $\mathcal{W} = \{W_1, \ldots, W_K\}$, where each $W_i = (A_i, C_i)$. Each feature is associated with a membership to the codewords which is given by

$$k = \arg\min_i d_M(M, W_i), \text{where } i \in \{1, \ldots, K\}. \tag{14}$$

In the experimental section, we will show the quality of the codebooks obtained using both these methods and, consequently, their impact on the categorization performance. In the next section, we show how the codebooks can be used to represent each of the training and testing video sequences.

3.3 Representing Videos Using the Codebook

Once the K codewords are available, each video sequence needs to be represented using this vocabulary. This is done by using a histogram $\mathbf{h} = [h_1, h_2, \ldots, h_K]^\top \in \mathbb{R}^K$. There are several choices for such a representation. In what follows, we will describe a few approaches that we will use to represent the video sequences. The experiments in Section 4 show which of these methods performs better.

We first introduce some notation. Let us assume that codeword k occurs c_{ik} times in the ith video sequence. Let N be the total number of video sequences and N_k be the number of video sequences in which codeword k occurs at least once. The simplest representation is called the *Term Frequency (TF)* and is defined as

Fig. 2. Sample snapshots from the UCLA8 database, which is a reorganized version of the UCLA50 dynamic texture database. Each image represents a sample frame from a different video sequence in the database.

$$h_{ik} = \frac{c_{ik}}{\sum_{k=1}^{K} c_{ik}}, k = 1, \ldots, K \text{ and } i = 1, \ldots, N. \quad (15)$$

Another popular approach is the *Term Frequency-Inverse Document Frequency (TF-IDF)* [30], defined as

$$h_{ik} = \left(\frac{c_{ik}}{\sum_{k=1}^{K} c_{ik}}\right) \ln\left(\frac{N}{N_k}\right). \quad (16)$$

Each of these two methods has its own advantages. The TF approach is the simplest of the approaches outlined above. Here, the focus is solely on the distribution of the codewords in a test video. The TF-IDF, on the other hand, discounts features that are common to all classes of video sequences and focuses on the ones that are unique to a particular class.

Once a histogram h is computed, we normalize it by its L_1 norm. As a consequence, h lies on the unit L_1 ball in $\mathbb{R}^K$. We exploit this fact in our classification framework, which we introduce in the next section.

3.4 Classification

The final step in our framework is to classify a new query video sequence using the training data. Given the training database $\{(\mathbf{h}_i, l_i)\}_{i=1}^{N}$, where $\mathbf{h}_i$ is a histogram extracted from the ith training video and $l_i \in \{1, \ldots, L\}$ is the class label of the ith training video, our goal is to infer the class label of a new histogram $\tilde{h}$ associated with a new video $\tilde{V}$.

A simple approach to obtain the label associated with $\tilde{h}$ is to use the k-nearest neighbors (k-NN) classifier [12], where the query video sequence is assigned the majority class label of its k closest histograms from the training database. In order to do this, we first need a notion of a distance between histograms $\mathbf{h}_i = [h_{i1}, \ldots, h_{iK}]$ of the training set and the query histogram $\tilde{\mathbf{h}}$. We can use standard distances between histograms such as the χ^2-distance or the square root distance on the sphere [12], which are defined as

$$d_{\chi^2}(\mathbf{h}_1, \mathbf{h}_2) = \frac{1}{2}\sum_{k=1}^{K} \frac{|h_{1k} - h_{2k}|^2}{h_{1k} + h_{2k}}, \quad (17)$$

$$d_{\surd}(\mathbf{h}_1, \mathbf{h}_2) = \arccos\left(\sum_{i=1}^{K} \sqrt{h_{1k} h_{2k}}\right), \quad (18)$$

where h_{ik} denotes the kth element of the histogram vector $\mathbf{h}_i$.

Another approach is to use a discriminative classification scheme such as a kernel SVM. The standard kernel used for histograms is the radial basis kernel, defined as $\mathcal{K}(\mathbf{h}_1, \mathbf{h}_2) = e^{-\gamma d(\mathbf{h}_1, \mathbf{h}_2)}$, where γ is a free parameter usually learned by cross validation and $d(\mathbf{h}_1, \mathbf{h}_2)$ is any distance on the space of histograms.

Alternatively, one could use a generative classifier such as the Naive Bayes (NB) classifier used in [8]. The naive Bayes classifier assumes that, given a category, each codeword is independently drawn from a multinomial distribution specific to that particular category. Moreover, each category is selected according to a prior probability. A training set of labeled sequences is used to learn the category prior and category-specific multinomial codeword distributions. Finally, given the codeword distribution in a test video, the maximum a posteriori estimate of the category is computed using Bayes' rule to determine the category of the test video sequence. We refer the readers to [8] for more details on the implementation of the classifier.

4 Experimental Results

In this section, we present a detailed experimental evaluation of various aspects of our algorithm. We use the dataset from [27]. This dataset consists of 50 classes of 4 video sequences each. The state-of-the-art recognition results on this dataset, reported by Chan and Vasconcelos [4], are 97.5 percent by using the kernel version of the dynamic texture model and 96.5 percent by using just the Martin distance on the LDS parameters.

However, as we argued in the introduction, existing experiments suffer from two drawbacks. First, the reported results are not obtained on the entire video sequences, but on a manually extracted patch of size 48×48. The selection of the patch is not consistent across the video; hence the results are not reproducible for novel videos without having to first select a patch of interest. Second, the 50 classes are obtained by artificially separating videos of the same semantic class that are taken from different viewpoints and scales and treating them as different classes. For example, water far and water close were considered as separate classes. If one combines the sequences of the same dynamic texture that were taken from different viewpoints/scales, the dataset can be reduced to a nine class dataset with the classes being boiling water (8), fire (8), flowers (12), fountains (20), sea (12), smoke (4), water (12), waterfall (16), and plants (108). Here the numbers in parenthesis represent the number of sequences in the dataset. Since the number of sequences of plants far outnumbered the number of sequences for the other classes, in this paper we consider two reorganized datasets: UCLA8, which contains 88 videos of 8 classes, and UCLA9, which contains 112 videos of 9 classes.[1] Fig. 2 shows some sample frames from this database

1. The UCLA9 database contains 32 randomly chosen plant video sequences as opposed to all 108 to prevent training bias.

Before presenting the results of our method on these two databases, we first explain implementation details of the various stages in our pipeline as well as the state-of-the art methods we compare against.

4.1 Implementation Details

Feature extraction. As stated earlier, we use a dense sampling approach for extracting features from the video sequences. Given a video sequence, we divide it into non-overlapping spatiotemporal volumes of size $\sigma \times \sigma \times \tau$, where σ represents the spatial size and τ represents the temporal size. We vary the spatiotemporal volumes such that $\sigma \in \{20, 30, 60\}$ and $\tau \in \{15, 25\}$. This gives rise to six different sets of spatiotemporal volumes. We model each spatiotemporal volume using an LDS of order n. In the experiments, we chose $n = 3$ and we do not optimize this parameter. Additionally, we wish to point out that we use each of the six sets separately and do not combine spatiotemporal patches of different sizes. In order to be able to extract such nonoverlapping volumes and make sure we do not have partial volumes, we resample the spatial size of the video sequences. In the original database, each frame of the video sequence is of size 110×160 and the video sequence contains 75 frames. We resample all the video sequences such that each frame is of size 120×180 to ensure that we utilized the entire video sequence while extracting nonoverlapping patches and not disregard any region. We then extract our features from the resampled videos as opposed to the original video sequences.

Codebook formation. In order to obtain the codebook for our categorization framework, we use the two approaches outlined earlier in Section 3.2. In the case of the K-Medoid approach, we run the algorithm directly on the Martin distances between the LDSs identified from the spatiotemporal volumes extracted from the training videos. For the K-Means approach, we first perform MDS on the distance matrix D as described in Section 3.2 and then use the K-Means algorithm to cluster the low-dimensional representation. We wish to point out that the dimension d_e of the euclidean points is directly obtained by the MDS algorithm. We do not specify a dimension a priori nor do we reduce the dimension of the points after the MDS step. For both approaches, we varied the number of clusters K from 8 to 96 in steps of 8.

Representation and classification. Once the codebook is formed, we have two choices by which we can represent a video sequence, namely TF and TF-IDF. Also, we have two choices of distances between histograms, namely, the χ^2-distance and the square root distance on the sphere. In addition, the classifier can be k-NN with $k = 1$ or 3, naive Bayes, or SVM. This results in eight k-NN classifiers, one naive Bayes classifier, and two SVM classifiers, which gives a total of 11 possible classifiers. We show results using all these methods and discuss which method is best. All results reported in this paper are averaged over 10 runs of our algorithm for each parameter choice. This is because in each run of the algorithm, the codebook can change as the K-Means and the K-Medoid algorithms are initialized randomly and can converge to different cluster centers.

From each class in the database, we choose 50 percent of the sequences for training codewords and the remaining 50 percent for testing purposes. For comparison, the same training/testing split is used for all the baseline methods which we will explain in the following.

4.2 Baseline Methods

We compare the performance of our BoS approach with two baselines. Our first baseline is the traditional single LDS approach to categorize dynamic textures [27] and our second baseline is the BoF approach [18], [37], [9] for categorizing videos. We briefly explain these methods and give some implementation details.

Single LDS approach. In our first baseline method, we model the entire video sequence using a single LDS. Given a test video sequence, we compute the Martin distance between the test LDS and each of the LDS models of the testing set. Based on this distance, we use three different classifiers, namely, the k-NN classifier with $k = 1$ and $k = 3$ and an SVM classifier and compare our results against the best out of these. As for the system order, we tested all system orders in the range $[2, \ldots, 10]$, and considered the best result out of these as the single LDS baseline. This approach is identical to the one originally proposed in [27].

Comparison of spatiotemporal features. Our second baseline method is the BoF approach for categorizing video sequences. We extract different features from each of the spatiotemporal volumes using the dense sampling approach and compare their performance. We use five different types of descriptors for each spatiotemporal volume, namely, the Dollar descriptor [9], the 3DHOG descriptor [18], and three different versions of the descriptors proposed in Willems et al. [37].

For the Dollar descriptor, we extract the gradients from each spatiotemporal volume and concatenate it into a single vector. Similarly to other works [9], [35] that use the Dollar descriptor, we reduce the dimensionality of the feature vector to a 100-dimensional vector using PCA [16]. We denote this feature as DOLLAR100. In order to obtain the descriptors we used the original code provided by the authors at http://vision.ucsd.edu/pdollar/toolbox/doc/index.html.

For the Willems detector, we used three types of features based on the original representation. These include sum of gradients, sum of gradients along with their absolute sum, and the sum of the gradients in each directions. We denote these features as ESURF, ESURF1, and ESURF2 since they are extended version of the SURF descriptors [2] that are used for images. We use the code provided by the authors at http://homes.psat.kuleuven.be/gwillems/research/Hes-STIP/ to extract these features.

Finally, for the 3DHOG descriptor, we use the code from http://lear.inrialpes.fr/software and use the dense sampling approach to extract the features.

For each of these descriptors, we use the K-Means algorithm to cluster them and then use them in a BoF approach. Similarly to our approach, we varied the codebook size from 8 to 96 in steps of 8 to be able to compare the results directly with our methods.

4.3 Results

As described above, there are a number of implementation details that need to be considered. Especially important are

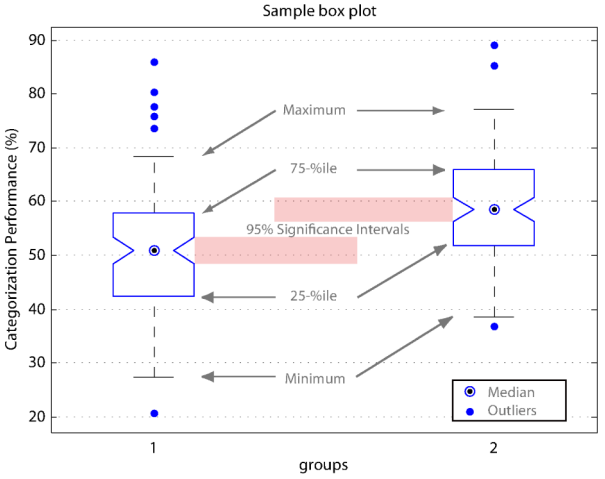

Fig. 3. Example of a box plot used to compare the statistics of two sets of categorization performances. The graphs display the nonoutlier minimum and maximum performance, the 25- and 75-percentiles, median and outliers. The notches around the median represent ranges for statistical significance tests at 95 percent for difference in median performance.

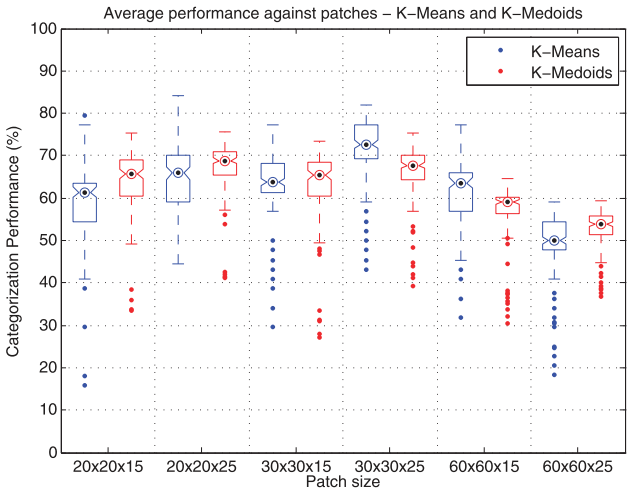

Fig. 4. Categorization performance of BoS on UCLA8 as a function of the patch size. Smaller patch sizes ($\sigma = 20, 30$) with larger temporal extents ($\tau = 25$) perform better than large patch sizes ($\sigma = 60$) or smaller temporal extents ($\tau = 15$).

1. the size of the spatiotemporal patch,
2. the clustering approach,
3. the number of clusters, and
4. the feature representation and classifier used.

All these factors directly affect the categorization performance and, while considering the parameter choices for one factor, we need to marginalize over the effect of all other factors. To analyze the variation in categorization performance with respect to each parameter, we will report various statistics of the categorization performance while fixing one particular parameter choice and varying across all other choices for the remaining parameters. For example, when testing the performance against patch sizes (six choices), there are two choices for the clustering method, 12 choices for the number of clusters, and 11 choices for the type of representation and classification method used. For each configuration of parameter choices, we ran 10 instances of the proposed categorization pipeline by randomly initializing the clustering method, and use the mean of these 10 runs as the categorization rate for that particular parameter choice configuration. This results in 132 different categorization results for each of K-Means and K-Medoid and each patch size. Therefore, to compare the categorization performance against patch sizes, we need to report statistics for the 132 runs for each patch size.

Box plots. We choose to display the results using box plots that will represent several statistics of categorization performance for each parameter choice over all variations of all other parameters. Fig. 3 shows a generic box-plot. Each plot displays a range of statistics for each group of data/results. The bottom and top whiskers represent nonoutliers (described later) minimum and maximum categorization percentages, the bottom and top of the box represent 25- and 75-percentiles and the central dot represents the median categorization rates. The dots beyond the whiskers represent automatically computed outliers that correspond to categorization performances greater than or smaller than 1.5 times the interquartile range beyond each of the respective quartiles. This corresponds to approximately $\pm 2.7\sigma$ or 99.3 coverage if the categorization percentages were normally distributed. The notches around the median represent ranges for confidence intervals to determine whether the difference in two median performances is statistically significant with 95 percent confidence under a Gaussian assumption. In particular, in Fig. 3, the difference between the two medians is statistically significant with 95 percent confidence as the notches do not overlap.

4.3.1 UCLA8 Database

Effect of patch size. Fig. 4 illustrates the categorization performance of the K-Means and K-Medoid-based clustering approaches against several patch sizes and the dense sampling approach. As we can see, the maximum categorization rate is achieved using K-Means-based clustering and a spatiotemporal patch size of $20 \times 20 \times 25$, whereas the median performance using a patch size of $30 \times 30 \times 25$ is statistically significantly better than the median performance across all other patch sizes. Although, the median performance of K-Means is not always significantly better than K-Medoid, the maximum performance of K-Means is always better than K-Medoid for all patch sizes.

We also experimented with a different sampling strategy, namely, the *grid sampling* approach. One of the disadvantages of using the dense sampling approach is that for the same video size, as the patch size decreases, the number of patches that can be extracted increases. This introduces a sampling bias as the total number of patches extracted for a small patch size is larger than the total number of patches extracted for a large patch size. To remove this sampling bias, we use grid sampling, where the number of extracted patches is the same for all patch sizes. We first divide the video into a grid of the largest patch size. Patches for all other patch sizes are extracted centered at the same pixels as the largest patch size. This guarantees that the number of spatiotemporal patch samples is the same across all patch sizes for a fair comparison between patch sizes. Fig. 5 shows the categorization rates. We see that for all patch sizes, the dense sampling approach gives better results than the grid-based approach. Moreover, the median performance for using

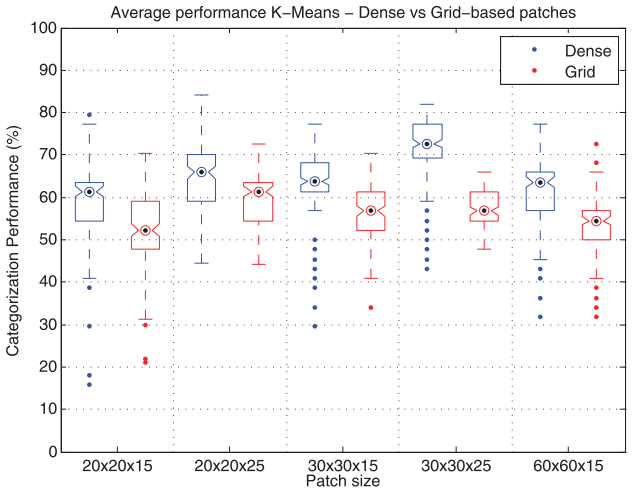

Fig. 5. Categorization performance of BoS on UCLA8 as a function of the patch sampling approach. Dense sampling performs better than grid-based sampling. Smaller patch sizes with larger temporal extent perform better for grid-based approaches as well.

dense patches is always better in a statistically significant sense. We observe that smaller patch sizes with larger temporal extents give the best overall performance ($20 \times 20 \times 25$) as well as the best statistically significant median performance ($30 \times 30 \times 25$). Larger patch sizes lead to degradation in performance.

Effect of representation and classifier. Fig. 6 displays the categorization performance against the choice of the classifier. We can see that overall the choice of the classifier does not affect the median performance of the algorithm by more than 10 percent. In fact, other than the naive Bayes classifier, the difference in median performance of any two classifiers is not necessarily statistically significant. Therefore, our framework is not particularly dependent on the choice of the representation, or the classifier used, but empirically, the median performance of TF-based representations is better than IDF-based representation. In this case, the naive Bayes

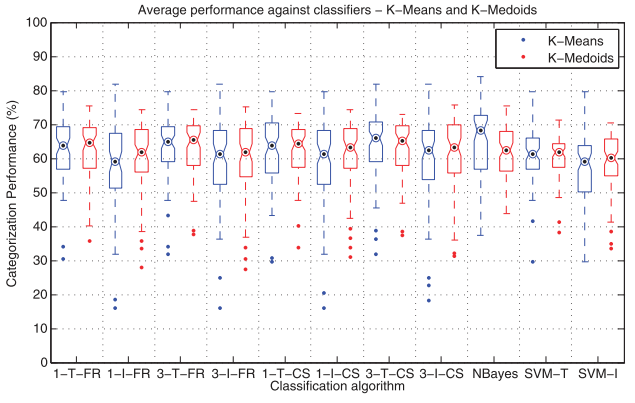

Fig. 6. Categorization performance of BoS on UCLA8 as a function of the classifier. The choice of representation and classifier does not significantly (statistically) affect the categorization performance. Here, the different classifiers are denoted by the tuple n-R-D, where n is the number of nearest neighbors (1 or 3), or SVM, R is the histogram representation (TF (T) or TF-IDF(I)), and D is the histogram distance (the chi-square distance (CS) or the distance on the sphere (FR)). For SVM, we only use the chi-square distance, and therefore we skip it in the axis labels.

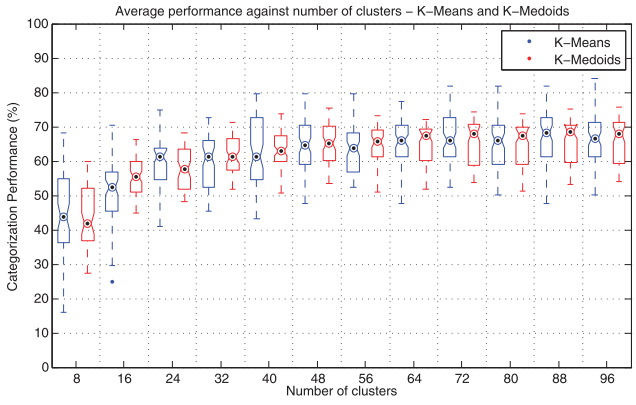

Fig. 7. Categorization performance of BoS on UCLA8 as a function of the codebook size. Larger codebooks give better performance for both K-Means and K-Medoid-based approaches. However, the performance saturates after a certain codebook size.

classifier is overall the better choice with a slightly better median and maximum categorization rate.

Number of codewords/clusters. Fig. 7 displays the performance of all the algorithms against the number of clusters used. As expected, using more clusters leads to increased performance, with the best performance achieved with 96 clusters. Again, we see that the performance of K-Means-based clustering is better than that of K-Medoid. Moreover, the difference in median performance when using more than 64 clusters is not statistically significant and hence, beyond a certain point, the choice of the number of clusters does not significantly change the categorization performance.

K-Means versus K-Medoid. From Figs. 4, 6, and 7, we see that although the difference in the median performance of K-Means versus K-Medoid-based clustering is not always statistically significant, except for the case of the naive Bayes classifier, the maximum performance of K-Means-based clustering approaches is always better than that of K-Medoid. This is because K-Means computes a geometrically accurate mean of each cluster. However, there is a tradeoff as K-Medoid gives better computational performance than K-Means at the cost of decreased accuracy. Overall, the choice of using K-Means versus K-Medoid does not affect the categorization performance in a significant way and either of the clustering methods can be chosen based on the tradeoffs mentioned.

Comparison to state of the art. Finally, to compare our approach against standard BoF-based approaches, we used the same patch sizes, number of clusters, and classifiers with K-Means-based clustering as for our BoS approach. Fig. 10 shows statistics of categorization performance of our BoS method against other BoF methods. Across all patch sizes, the maximum categorization rate of our proposed approach is at-par or better than other feature-based approaches. Moreover, for larger patch sizes, our approach performs significantly better than other BoF-based methods. The standard LDS-based approach performs at 52 percent, which is much worse than even standard BoF-based methods. Fig. 9 shows the confusion matrix for our approach on the UCLA8 database corresponding to the best categorization rate achieved, 84 percent. Comparing this to the confusion matrix in Fig. 8 using the single LDS-based approach, with a categorization

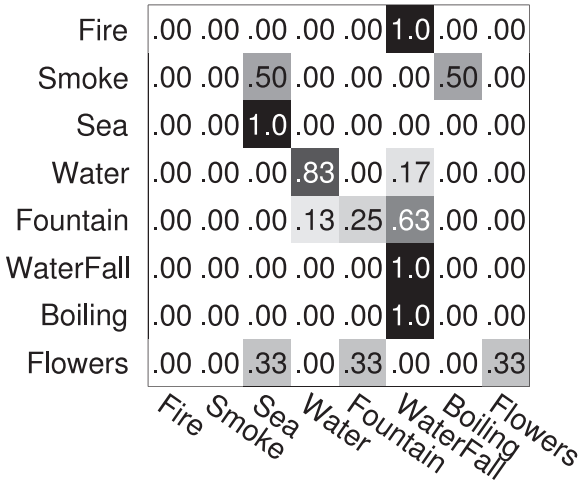

Fig. 8. Confusion matrix of the best performing single LDS baseline method on UCLA8 (Martin distance with SVM classifier). The overall categorization performance is 52 percent.

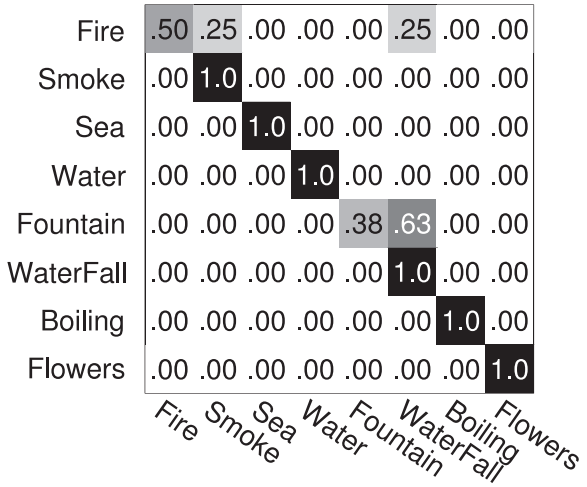

Fig. 9. Confusion matrix of the best performing BoS approach on UCLA8. The overall categorization rate is 84 percent.

rate of 52 percent, we can clearly see that our approach performs significantly better than the state of the art. In particular, our method only makes errors in two classes, fire and fountain, whereas the single LDS method altogether fails to recognize some classes, fire, smoke, sea, and boiling, and makes large errors in the remaining classes.

4.3.2 UCLA9 Database

We also tested our approach on the UCLA9 database that contains the additional plants class and all eight classes in UCLA8. Similar to our experiments for the UCLA8 dataset, Fig. 11 shows statistics for categorization performance of our approach against several state-of-the-art BoF-based approaches. We again see that overall our proposed approach performs at-par with or better than all state-of-the-art bag-of-features-based approaches and much better than the single LDS approach. The median performance of our approach is statistically better than all other approaches for all patch sizes except $20 \times 20 \times 15$. Fig. 13 displays the confusion matrix for the best categorization performance of our approach and Fig. 12 displays the confusion matrix achieved using the single LDS method. We achieve an overall categorization rate of 78 percent compared to 48 percent for the state-of-the-art single LDS approach. Here again we see that our method performs much better than the single LDS approach and only makes errors in four classes, whereas the single LDS method makes errors in all classes except one. We conclude that the increase in categorization comes from explicitly using local dynamical systems for modeling dynamic textures as opposed to using a single global model.

5 Discussion and Conclusions

In this paper, we proposed a Bag-of-Systems approach for categorizing dynamic textures. By modeling a video with the distributions of local dynamical models extracted from it, we showed that we are able to better handle the variations in view-point and scale in the training and test

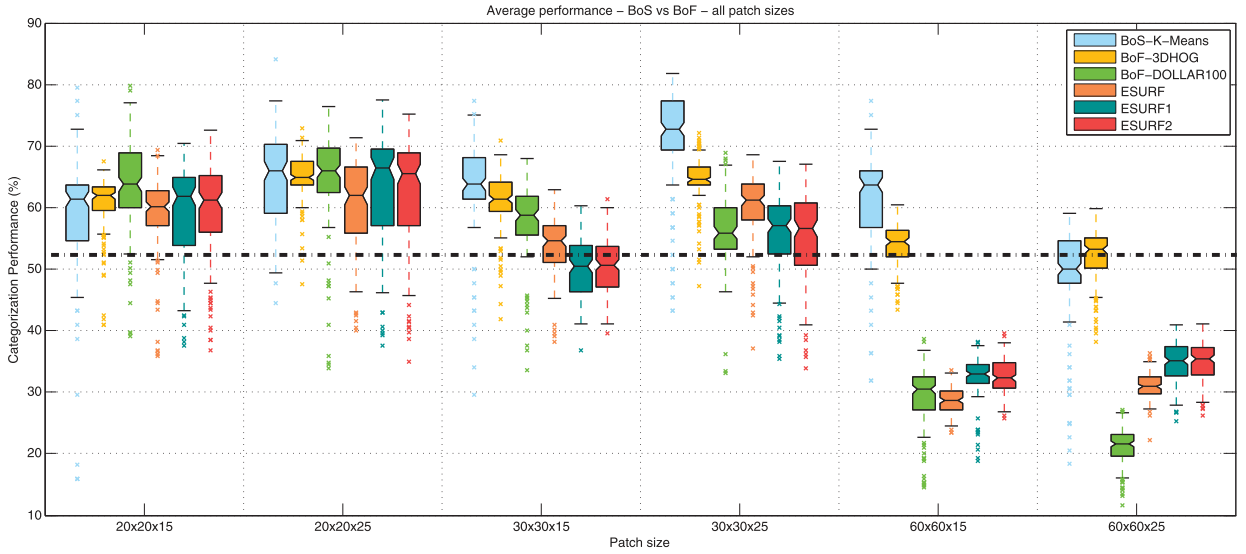

Fig. 10. Comparison of the categorization performance of the BoS method against several state-of-the-art BoF-based methods for different patch sizes on the UCLA8 database. A dashed line indicates the best performance of the single LDS approach from [10].

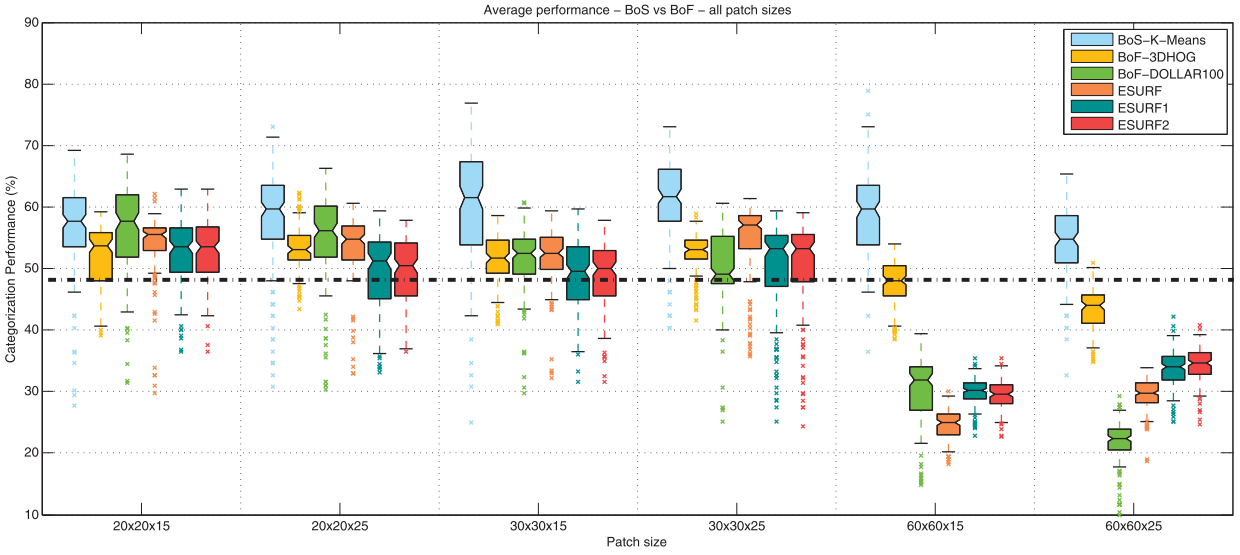

Fig. 11. Comparison of the categorization performance of the BoS method against several state-of-the-art BoF-based methods for different patch sizes on the UCLA9 database. A dashed line indicates the best performance of the single LDS approach from [10].

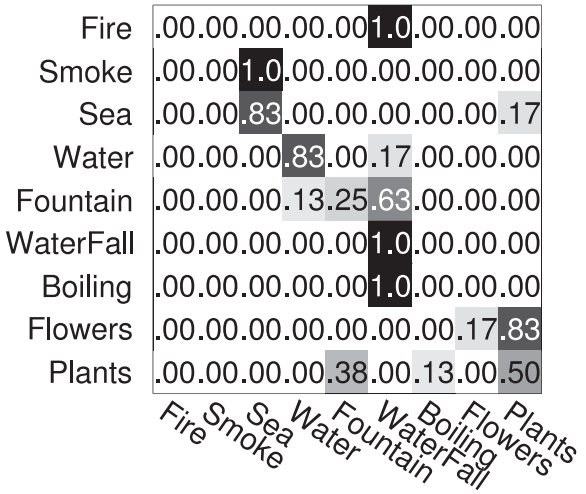

Fig. 12. Confusion matrix of the best performing single LDS baseline method on UCLA9. The overall categorization performance is 48 percent.

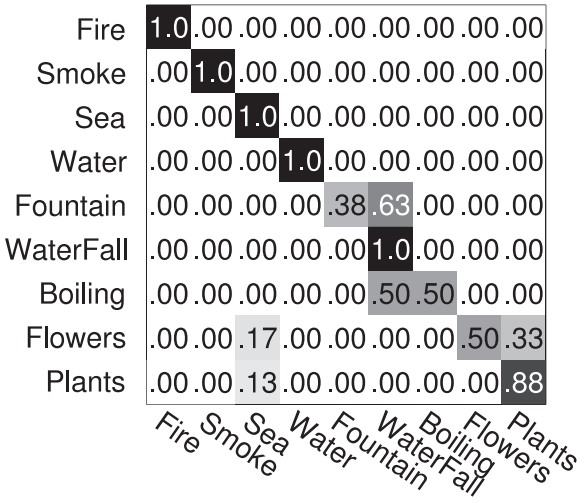

Fig. 13. Confusion matrix of the best performing BoS approach on UCLA9. The overall categorization performance is 78 percent.

data as compared to modeling the entire video sequence with a single global model. In order to construct the BoS model for a video sequence, we had to tackle several challenges, specifically in the codebook formation step of the pipeline. Toward this end, we proposed a method that uses distances on the space of dynamical systems, nonlinear dimensionality reduction techniques, and K-Means or K-Medoid to efficiently construct the dynamical systems codebook. We extensively compared our algorithm with standard Bag of Features approaches using a variety of different features for categorizing video sequences as well as the original single LDS approach. Our experimental results showed that our approach produces better results across different parameter choices and empirically established the superior performance of our proposed approach. By moving from the traditional single model approach to multiple models, we can see that the performance increases. One way to further improve this model is to combine both local and global methods. Our recent approach [25] showed that we achieve a better categorization performance by combining both local and global models.

References

[1] Z. Bar-Joseph, R. El-Yaniv, D. Lischinski, and M. Werman, "Texture Mixing and Texture Movie Synthesis Using Statistical Learning," *IEEE Trans. Visualization and Computer Graphics*, vol. 7, no. 2, pp. 120-135, Apr.-June 2001.

[2] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded Up Robust Features," *Proc. European Conf. Computer Vision*, May 2006.

[3] D. Blei, A. Ng, and M. Jordan, "Latent Dirichlet Allocation," *J. Machine Learning Research*, vol. 3, pp. 993-1022, 2003.

[4] A. Chan and N. Vasconcelos, "Classifying Video with Kernel Dynamic Textures," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 1-6, 2007.

[5] A. 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.

[6] K.D. Cock and B.D. Moor, "Subspace Angles and Distances between ARMA Models," *System and Control Letters*, vol. 46, no. 4, pp. 265-270, 2002.

[7] T.F. Cox and M.A.A. Cox, *Multidimensional Scaling.* Chapman and Hall, 1994.
[8] C. Dance, J. Willamowski, L. Fan, C. Bray, and G. Csurka, "Visual Categorization with Bags of Keypoints," *Proc. European Conf. Computer Vision,* 2004.
[9] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior Recognition via Sparse Spatio-Temporal Features," *Proc. IEEE Int'l Workshop Visual Surveillance and Performance Evaluation of Tracking and Surveillance,* Oct. 2005.
[10] G. Doretto, A. Chiuso, Y. Wu, and S. Soatto, "Dynamic Textures," *Int'l J. Computer Vision,* vol. 51, no. 2, pp. 91-109, 2003.
[11] G. Doretto, D. Cremers, P. Favaro, and S. Soatto, "Dynamic Texture Segmentation," *Proc. IEEE Conf. Computer Vision,* pp. 44-49, 2003.
[12] R. Duda, P. Hart, and D. Stork, *Pattern Classification.* Wiley-Interscience, Oct. 2004.
[13] K. Fujita and S. Nayar, "Recognition of Dynamic Textures Using Impulse Responses of State Variables," *Proc. Third Int'l Workshop Texture Analysis and Synthesis,* Oct. 2003.
[14] A. Ghoreyshi and R. Vidal, "Segmenting Dynamic Textures with Ising Descriptors, ARX Models and Level Sets," *Proc. Int'l Workshop Dynamic Vision,* pp. 127-141, 2006.
[15] T. Hofmann, "Probabilistic Latent Semantic Analysis," *Proc. Uncertainty in Artificial Intelligence,* 1999.
[16] I. Jolliffe, *Principal Component Analysis,* second ed. Springer-Verlag, 2002.
[17] L. Kaufman and P. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis.* Wiley, 1990.
[18] A. Kläser, M. Marszałek, and C. Schmid, "A Spatio-Temporal Descriptor Based on 3D-Gradients," *Proc. British Machine Vision Conf.,* pp. 995-1004, 2008.
[19] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick, "Graphcut Textures: Image and Video Synthesis Using Graph Cuts," *ACM Trans. Graphics,* vol. 22, pp. 277-286, 2003.
[20] I. Laptev, "On Space-Time Interest Points," *Int'l J. Computer Vision,* vol. 64, nos. 2/3, pp. 107-123, 2005.
[21] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 2, pp. 2169-2178, 2006.
[22] D. Nister and H. Stewenius, "Scalable Recognition with a Vocabulary Tree," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 2161-2168, 2006.
[23] 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.
[24] A. Ravichandran, R. Chaudhry, and R. Vidal, "View-Invariant Dynamic Texture Recognition Using a Bag of Dynamical Systems," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2009.
[25] A. Ravichandran, P. Favaro, and R. Vidal, "A Unified Approach to Segmentation and Categorization of Dynamic Textures," *Proc. Asian Conf. Computer Vision,* pp. 425-438, 2010.
[26] S. Roweis and L. Saul, "Nonlinear Dimensionality Reduction by Locally Linear Embedding," *Science,* vol. 290, no. 5500, pp. 2323-2326, 2000.
[27] P. Saisan, G. Doretto, Y.N. Wu, and S. Soatto, "Dynamic Texture Recognition," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 2, pp. 58-63, 2001.
[28] A. Schödl, R. Szeliski, D.H. Salesin, and I. Essa, "Video Textures," *Proc. ACM Siggraph,* pp. 489-498, 2000.
[29] R. Shumway and D. 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.
[30] J. Sivic and A. Zisserman, "Video Google: A Text Retrieval Approach to Object Matching in Videos," *Proc. IEEE Int'l Conf. Computer Vision,* pp. 1470-1477, 2003.
[31] M. Szummer and R.W. Picard, "Temporal Texture Modeling," *Proc. IEEE Int'l Conf. Image Processing,* vol. 3, pp. 823-826, 1996.
[32] J.B. Tenenbaum, V. de Silva, and J.C. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction," *Science,* vol. 290, no. 5500, pp. 2319-2323, 2000.
[33] R. Vidal and P. Favaro, "Dynamicboost: Boosting Time Series Generated by Dynamical Systems," *Proc. IEEE Int'l Conf. Computer Vision,* 2007.
[34] S. Vishwanathan, A. 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.
[35] H. Wang, M.M. Ullah, A. Kläser, I. Laptev, and C. Schmid, "Evaluation of Local Spatio-Temporal Features for Action Recognition," *Proc. British Machine Vision Conf.,* p. 127, Sept. 2009.
[36] L. Wei and M. Levoy, "Fast Texture Synthesis Using Tree-Structured Vector Quantization," *Proc. ACM Siggraph,* pp. 479-488, 2000.
[37] G. Willems, T. Tuytelaars, and L.J.V. Gool, "An Efficient Dense and Scale-Invariant Spatio-Temporal Interest Point Detector," *Proc. European Conf. Computer Vision,* 2008.
[38] S.-F. Wong and R. Cipolla, "Extracting Spatiotemporal Interest Points Using Global Information," *Proc. IEEE Int'l Conf. Computer Vision,* pp. 1-8, 2007.
[39] F. Woolfe and A. Fitzgibbon, "Shift-Invariant Dynamic Texture Recognition," *Proc. European Conf. Computer Vision,* pp. 549-562, 2006.

Avinash Ravichandran received the BE degree in electronics and communication engineering from the University of Madras in 2003, the master's degree in electrical and computer engineering in 2007, the master's degree in applied mathematics and statistics in 2009, and the PhD degree in electrical and computer engineering in 2010 from The Johns Hopkins University. Currently, he is working as a post-doctoral fellow with the Vision Lab at the University of California, Los Angeles. His research interests include analysis of temporal events, activities, and dynamic textures. He is a member of the IEEE.

Rizwan Chaudhry received the BSc (Honors) degree with a double major in computer science and mathematics from the Lahore University of Management Sciences in Lahore, Pakistan, in 2005. He received the PhD degree in computer science from The Johns Hopkins University in 2012 where he was associated with the Vision, Dynamics and Learning lab. He is currently part of the Video Cognition group at Microsoft. His areas of research include modeling dynamic visual phenomena, human activity recognition, and tracking. He is a member of the IEEE.

René Vidal received the BS degree in electrical engineering (highest honors) from the Pontificia Universidad Catolica de Chile in 1997 and the MS and PhD degrees in electrical engineering and computer sciences from the University of California, Berkeley, in 2000 and 2003, respectively. He was a research fellow at the National ICT Australia in the Fall of 2003 and currently is an associate professor in the Department of Biomedical Engineering at The Johns Hopkins University. He has coauthored more than 150 articles in biomedical image analysis, computer vision, machine learning, hybrid systems, and robotics. He is a recipient of the 2012 J.K. Aggarwal Prize "for outstanding contributions to generalized principal component analysis (GPCA) and subspace clustering in computer vision and pattern recognition," the 2012 Best Paper Award in Medical Robotics and Computer Assisted Interventions, 2011 Best Paper Award Finalist at the IEEE Conference on Decision and Control, the 2009 ONR Young Investigator Award, the 2009 Sloan Research Fellowship, the 2005 NFS CAREER Award, and the 2004 Best Paper Award Honorable Mention at the European Conference on Computer Vision. He also received the 2004 Sakrison Memorial Prize for completing an exceptionally documented piece of research, the 2003 Eli Jury award for outstanding achievement in the area of systems, communications, control, or signal processing, the 2002 Student Continuation Award from NASA Ames, the 1998 Marcos Orrego Puelma Award from the Institute of Engineers of Chile, and the 1997 Award of the School of Engineering of the Pontificia Universidad Catolica de Chile to the best graduating student of the school. He is a senior member of the IEEE and the ACM.