View-Invariant Dynamic Texture Recognition using a Bag of Dynamical Systems

Avinash Ravichandran, Rizwan Chaudhry and René Vidal
Center for Imaging Science, Johns Hopkins University, Baltimore, MD 21218, USA

Abstract

In this paper, we consider the problem of categorizing videos of dynamic textures under varying view-point. We propose to model each video with a collection of Linear Dynamics Systems (LDSs) describing the dynamics of spatiotemporal video patches. This bag of systems (BoS) representation is analogous to the bag of features (BoF) representation, except that we use LDSs as feature descriptors. This poses several technical challenges to the BoF framework. Most notably, LDSs do not live in a Euclidean space, hence novel methods for clustering LDSs and computing codewords of LDSs need to be developed. Our framework makes use of nonlinear dimensionality reduction and clustering techniques combined with the Martin distance for LDSs for tackling these issues. Our experiments show that our BoS approach can be used for recognizing dynamic textures in challenging scenarios, which could not be handled by existing dynamic texture recognition methods.

1. Introduction

Dynamic textures are video sequences of complex scenes such as water on the surface of a lake, a flag fluttering in the wind, *etc*. The constant change in the geometry of the these objects poses a challenge for applying traditional vision algorithms to these video sequences. Over the past few years, several algorithms have been proposed for vision tasks such as registration, segmentation and categorization of such scenes. Among these tasks, the categorization of such video sequences is of specific interest to us, because it is critical to surveillance applications, such as detecting fires in tunnels, leaks in pipes on ships, *etc*.

Doretto et al. in [6], showed that dynamic textures can be modeled using a LDS. Using the LDS representation, several categorization algorithms have been proposed based on the LDS parameters, which live in a non-Euclidean space. One popular genre of methods first defines a distance or a kernel between the model parameters of two dynamical systems. Once such a distance or kernel has been defined, classifiers such as nearest neighbors or support vector machines (SVM) [7] can be used to categorize a query video sequence based on the training data. Among these methods, Saisan et al. [12] used distances based on the principal angles between the subspaces generated by the parameters of the LDSs. Vishwanathan et al. [15], used Binet-Cauchy kernels to compare the parameters of two LDSs. Further, Chan et al. used both the KL divergence [1] and the Martin distance [2] as a metric between dynamical system. Finally, Woolfe et al. [18] 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 et al. [8], divide the video sequences into blocks and compare the trajectories of the states in order to perform the inference. Alternatively, Vidal et al. [14] extended boosting to LDSs by using dynamical systems as weak classifiers.

Although there exists several methods for dynamic texture categorization, most of these methods have validated their performance on the database from [12], which consists of 50 classes with 4 video sequences per class. However, if one examines this database, several of the classes are from a single category, e.g., water, but from different viewpoints. Thus, in reality there are only 9 different semantic categories. In addition to the lack of intraclass variability, most existing methods work with a manually extracted subsequence at the center of the video, as opposed to using the whole video sequence. In this paper, we aim to address a more challenging problem, namely the categorization of dynamic textures with invariance to changes in view point. To the best of our knowledge, apart from [18], which addressed shift invariance, there is no other work addressing the dynamic texture categorization problem with invariance to viewpoint.

The bag of features (BoF) approach for object recognition has been shown to handle variability in viewpoints, illumination and scales. Recently, the BoF method was extended to categorizing video sequences [5, 17, 16, 9], wherein traditional image features were replaced by spatiotemporal features. Specifically, once feature points are extracted, a descriptor is formed based on gradients in a spatiotemporal neighborhood around the feature point. After this step, the pipeline is exactly the same as in BoF for images.

In principle, one could follow the same approach for categorizing dynamic textures. However, the fact that dynamic textures can be modeled with a LDS would not be exploited.

Moreover, videos of dynamic textures tend to be fairly homogeneous. Even though existing detectors can be used to find feature points in such video sequences, using gradients as descriptors may not be discriminative enough.

Paper Contributions. In this paper, we propose to extend the BoF approach to dynamic textures, by replacing traditional features by LDSs. By using LDSs as features, we obtain an alternate representation of these video sequences which preserves their dynamics, as opposed to image gradients. The fact that the parameters of a LDS live in a non-Euclidean space poses several challenges to the BoF framework. We propose a method to form a codebook from these non-Euclidean parameters by performing nonlinear dimensionality reduction and clustering in the space of LDSs. We evaluate various weighting schemes used to represent a video sequence using the codebook. We also show that by using the BoS approach, we are able to recognize dynamic textures under changes in viewpoint and outperform existing methods.

2. Preliminaries

In this section, we first introduce the dynamic texture framework and show how video sequences can be modeled using this framework. We then introduce the Martin distance between two LDSs. This distance will serve as our similarity metric in the Bag of Systems framework.

2.1. Dynamic Texture Model

Given a video sequence or a spatiotemporal patch, $\{I(t)\}_{t=1}^{F}$, we model it as the output of a LDS as

$$z(t+1) = Az(t) + Bv(t) \qquad (1)$$

$$I(t) = Cz(t) + w(t), \qquad (2)$$

where $z(t) \in \mathbb{R}^n$ is the hidden state at time t , $A \in \mathbb{R}^{n\times n}$ represents the dynamics of the system, $C \in \mathbb{R}^{p\times n}$ maps the hidden state to the output of the system, and $w(t) \sim \mathcal{N}(0, R)$ and $v(t) \sim \mathcal{N}(0, Q)$ are the measurement and process noise respectively. The order of the system is given by n and p is the number of pixels in one frame of the sequence or patch.

Given a video sequence, the model parameters can be identified using a Principal Component Analysis (PCA) based approach outlined in [6]. As a consequence of the identification algorithm, the parameter C belongs to the Stiefel Manfiold $\mathbb{ST}(p, n) = \{C|C^\top C = I \in \mathbb{R}^{n\times n}\}$. 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 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. Given two feature descriptors, $M_1 = (A_1, C_1)$ and $M_2 = (A_2, C_2)$, we first need to define a notion of similarity between these two descriptors. One such 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 $\mathcal{O}_\infty(M) = [C^\top, (CA)^\top, (CA^2)^\top, \ldots]^\top \in \mathbb{R}^{\infty\times n}$. 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

$$\mathcal{P} = \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} \in \mathbb{R}^{2n\times 2n}, \quad \mathcal{A} = \begin{bmatrix} A_1 & \mathbf{0} \\ \mathbf{0} & A_2 \end{bmatrix} \in \mathbb{R}^{2n\times 2n},$$
$$\mathcal{C} = \begin{bmatrix} C_1 & C_2 \end{bmatrix} \in \mathbb{R}^{p\times 2n}.$$

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

$$\cos^2\theta_i = i\text{-th eigenvalue}(P_{11}^{-1}P_{12}P_{22}^{-1}P_{21}). \qquad (3)$$

The Martin distance between M_1 and M_2 is now defined as

$$d_M(M_1, M_2)^2 = -\ln\prod_{i=1}^{n}\cos^2\theta_i. \qquad (4)$$

3. Bag of Systems

The BoF approach in computer vision draws its inspiration from the document retrieval community. Here, a document is hypothesized to be identifiable by the distribution of certain keywords. In the BoF approach, a similar philosophy is adopted, wherein feature descriptors are used in lieu of words. Images are now categorized by observing the distribution of a small collection of features called codewords.

The typical steps followed in the BoF framework are: (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 dataset is represented using the codebook, (4) a classifier is chosen to compare a new query image to the training set and thus infer its category.

In this section we describe the corresponding BoS analogue of the BoF steps. We wish to point out that our approach is not a mere change in the feature descriptor. As a consequence of using the parameters of a LDS as a feature descriptor, we now have features in a non-Euclidean space. Therefore, clustering these descriptors is highly challenging.

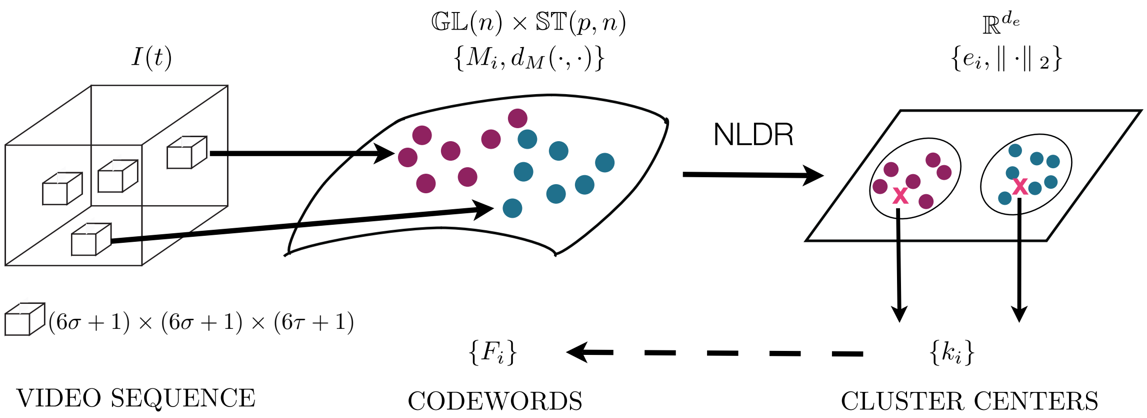

$I(t)$
$\mathbb{GL}(n) \times \mathbb{ST}(p,n)$
$\{M_i, d_M(\cdot,\cdot)\}$
$\mathbb{R}^{d_e}$
$\{e_i, \|\cdot\|_2\}$
NLDR
$(6\sigma+1)\times(6\sigma+1)\times(6\tau+1)$
$\{F_i\}$
$\{k_i\}$
VIDEO SEQUENCE
CODEWORDS
CLUSTER CENTERS

Figure 1. Overview of the bag of systems framework for categorizing dynamic textures with invariance to changes in view point.

In this paper we propose a framework for codebook formation by clustering in the space of LDSs. An overview of our framework is shown in Fig. 1.

3.1. Feature Extraction and Descriptors

The first step in our framework is to extract feature points from the video sequences. Over the past few years, several spatiotemporal feature point methods have been proposed. Laptev et al. [9] extend the Harris corner detector used for images to video sequences by considering a second moment matrix of the spatiotemporal gradients. Willems et al. [16] use the spatiotemporal Hessian. Dollar et al. [5] 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 et al. [17] model the video sequence with a LDS and extract features from the parameters of the LDS, as opposed to directly from the video sequence. A detailed comparison of these methods can be found in [16]. These methods however, have concentrated on classifying video sequences of human activity, gestures and mouse activity.

In this paper, we use existing feature point selection methods for detecting feature points. However, a key difference is that we use LDS parameters as a feature descriptor for a spatiotemporal volume. Once a feature point and its corresponding spatiotemporal volume is extracted, a LDS is identified from this volume as described in §2.1, and represented using $M = (A, C)$.

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 the codebook. However, in our case the descriptors are parameters of a LDS, i.e. (A, C), which lie on a non-Euclidean space. More specifically, $(A, C) \in \mathbb{GL}(n) \times \mathbb{ST}(p, n)$, where $\mathbb{GL}(n)$ is the group of all invertible matrices of size n and $\mathbb{ST}(p, n)$ is the Stiefel manifold. Hence, one cannot directly apply clustering algorithms that are used in the Euclidean case. However, one could find a low-dimensional Euclidean embedding of these points and apply the clustering algorithm in this space.

There exist several techniques for Non-Linear Dimensionality Reduction (NLDR). Some methods such as Locally Linear Embedding (LLE) [11], work directly with the points in the higher dimensional space, while other methods such as Multidimensional Scaling (MDS) [3], Isomap [13] work with the pairwise distances between the points in the higher dimensional space. In our case, we exploit the fact that the space of LDSs is endowed with several distances, e.g., the Martin distance (§2.2), and use this distance to perform dimensionality reduction and clustering to form the codebook.

Given the set of features $\{M_i\}_{i=1}^T$, where T represents the total number of features extracted from the 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 pairwise distances are available between the features, techniques such as Multidimensional Scaling (MDS), Isomap [13], *etc.* can be used to obtain a low-dimensional embedding of these points $\{e_i \in \mathbb{R}^{d_e}\}_{i=1}^T$, where d_e is the dimension of the embedding. This low-dimensional representation gives us a set of Euclidean points, which preserve the relationships in the high-dimensional nonlinear space. Clustering algorithms can be applied to $\{e_i\}_{i=1}^T$, as the the low-dimensional space is Euclidean.

After this clustering, we now have K cluster centers $\{k_i\}_{i=1}^K$. However, these cluster centers do not correspond to any of the original LDSs. Moreover, there is no explicit map to go from the lower dimensional embedding to the original space. Hence, in order to select our codewords $\{F_i\}_{i=1}^K$, we choose the corresponding systems in the high-dimensional space whose distance to the cluster center in the lower dimensional space is the least, i.e.

$$F_i = M_p, \quad p = \arg\min_j \|e_j - k_i\|^2. \qquad (5)$$

In this way, we obtain our codebook $\mathcal{C} = \{F_1, \ldots, F_K\}$, where $F_i = (A_i, C_i)$.

During the query phase, each detected feature is associated with model parameters $M = (A, C)$. The membership to the codeword is given by $m = \arg\min_i d_M(M, F_i)$.

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 using a weight vector $W = \{w_1, w_2, \ldots w_K\} \in \mathbb{R}^K$. There are several choices for such a representation.

Let us assume that codeword k occurs N_{ki} times in the i^{th} video sequence and there are total of N_i codewords in the i^{th} video sequence. Let V be the total number of video sequences and V_i be the number of video sequences in which codeword i occurs. The simplest representation is called the *term frequency (TF)* and is defined as

$$w_{ik} = \frac{N_{ki}}{N_i}. \tag{6}$$

A more popular approach is the *term frequency - inverse document frequency (TF-IDF)* defined as

$$w_{ik} = \frac{N_{ki}}{N_i} \ln\left(\frac{V}{V_i}\right). \tag{7}$$

Another recently proposed approach termed as the *soft-weighting (SW)* is defined as

$$w_{ik} = \sum_{i=1}^{k_0} \sum_{j=1}^{L_i} \frac{1}{2^{i-1}} d(F_k, M_j) \tag{8}$$

where k_0 represents the number of neighbors (typically $k_0 = 4$) and L_i represents the number of codewords that have F_k as their i^{th} neighbor. This method as opposed to the TF, exploits the entire structure of the codebook.

Once a weight vector W is computed, it is normalized by its L_1 norm to become a histogram. In the experimental section, we evaluate the various representation schemes outlined above and comment on their optimality for our approach.

3.4. Classification

The final step in our framework is to classify a new query video sequence using the training data. Given the training dataset $\{(W_i, l_i)\}_{i=1}^{V}$, where $l_i \in \{1, \ldots, m\}$ is the class label of the weight vector, our goal is to infer the class label of a new weight vector W_t. To compare the training weight vectors and the query weight vector we can use standard distances between histograms such as the χ^2 distance or the square root distance on the sphere [7], which are defined as

$$d_{\chi^2}(W_1, W_2) = \frac{1}{2} \sum_{i=1}^{K} \frac{|w_{1i} - w_{2i}|}{w_{1i} + w_{2i}}, \tag{9}$$

$$d_S(W_1, W_2) = \arccos\left(\sum_{i=1}^{K} \sqrt{w_{1i} w_{2i}}\right). \tag{10}$$

A simple classification approach is the k-nearest neighbors (k-NN) classifier [7], where the query video sequence is assigned the majority class label of its k closest weight vectors from the training dataset.

A generative classification approach is the naive Bayes (NB) classifier used in Csurka et al. [4] . The posterior i.e. $P(l = j|I_i(t))$ is proportional to the product of the prior $P(l = j)$ and the likelihood $P(I_i(t)|l = j)$. Given the fact that the video sequences $I_i(t)$ can be represented by the codewords and assuming independence of the codewords, the likelihood on the codewords are learnt using the equation

$$P(F_k|l = j) = \frac{1 + \sum_{\{i|l_i = j\}} N_{ki}}{K + \sum_{s=1}^{K} \sum_{\{i|l_i = j\}} N_{si}}. \tag{11}$$

Given a new video sequence $I_n(t)$, the inference is then computed as the maximum aposterori estimate of

$$P(l = j|I_n(t)) \propto P(l = j) P(I_n(t)|l = j) \tag{12}$$

$$= P(l = j) \prod_{s=1}^{K} P(F_s|l = j)^{N_{sn}} \tag{13}$$

A discriminative classification approach is SVM with kernels. The standard kernel used for the weight vector is the radial basis kernel, defined as $K(W_1, W_2) = e^{-\gamma d(W_1, W_2)}$, where γ is a free parameter usually learnt by cross validation and $d(W_1, W_2)$ is any distance on the space of histograms.

3.5. Implementation Details

In our framework, we use the approach outlined in [5] for extracting the features. We find this feature detector more appropriate for dynamic textures than those of [9, 16], which are based on image gradients. Once a feature point is detected at a spatial scale σ and temporal scale τ, a volume of size $(6\sigma + 1) \times (6\sigma + 1) \times (6\tau + 1)$ centered around the feature point is extracted. For each of these volumes, a dynamic texture model of order n is identified using PCA-based approach. Unlike [9, 16], we do not have a method to calculate scales automatically. Hence, we use multiple values of σ and τ, in order to detect feature points at different scales. Also, the Martin distance can be applied to systems of different orders, but requires the dimensions of the output spaces of the two systems, i.e. the number of pixels in a frame, to be the same. Hence, we normalize the spatial size

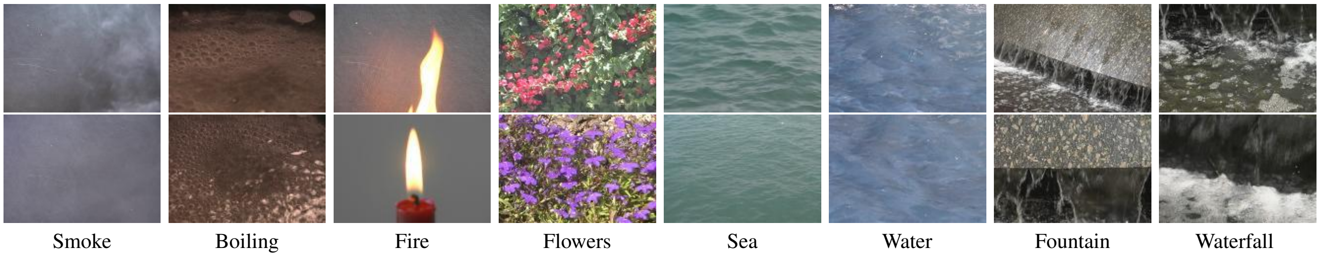


Figure 2. Sample snapshots from the reorganized dynamic texture database. This represents the 8 classes we consider in our framework, while in the original database these are considered as 22 classes.

of the spatiotemporal volumes to be uniform. We however do not require the same temporal size for these volumes.

In order to perform the clustering in the low-dimensional space, we use Hierarchical K-means (HKM)[10]. It is well known that K-means is sensitive to initialization. Hence, K-means is run multiple times and the cluster centers with the minimum intraclass variance are selected as the cluster centers. Since we want stable codewords, as opposed to minimum variance, we run HKM multiple times and select the most frequent cluster centers as our codewords. In our experiments, since the number of codewords used was small, we directly use the leaves of the HKM as the cluster centers.

4. Experiments

In this section we present a detailed experimental evaluation of various aspects of our algorithm. We use the dataset from [12]. This dataset consists of 50 classes of 4 video sequences each. The state of the art recognition results on this dataset reported by Chan et al. in [2], is 97.5% by using the kernel version of the dynamic texture model and 96.5% by using just the Martin distance on the LDS parameters.

We wish to point out that these numbers are based on the 50 class structure and the reported results are not on the entire video sequences, but on a manually extracted patch of size 48×48. However, if one combines the sequences that are taken from different viewpoints, the dataset can be reduced to a 9 class dataset with the classes being boiling water (8), fire (8), flowers (12), fountains (20), plants (108), sea (12), smoke (4), water (12) and waterfall (16). Here the numbers 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, we used the remaining 8 classes in our experiments. Sample frames from the video sequences in the database are show in Fig. 2.

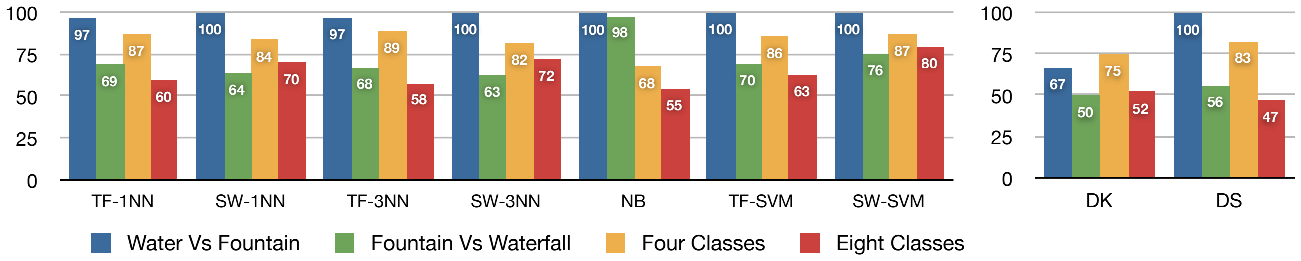
We show results for four different scenarios in this paper. We first consider an easy two class problem namely the case of water vs. fountain. Then a more challenging two class problem namely the fountain vs. waterfall. Our third set of experiments is on a four class problem and the last set is on the reorganized database with 8 classes. All the results presented in this section have been averaged over 20 trials.

Codebook formation. We experimented with the two NLDR methods, namely MDS and Isomap. However, we noticed that, with all other experimental conditions remaining the same, recognition rates by using MDS were higher than those of Isomap. We also found the optimal codebook size to be $K = 8$ and the number of restarts for the HKM to be 16. Also, the order of the LDSs was set to $n = 8$ and we did not optimize this parameter.

Representation and Classification. Once the codebook is formed, we have 3 choices for representing a video sequence, namely TF, TF-IDF and SW, and two choices of distances between histograms, namely χ^2 distance and the square root distance on the sphere. In addition, the classifier can k-NN, naive Bayes or SVM. Overall this gives us 14 ways to perform the inference, given the codebook. In our experiments, we noticed that the TF-IDF representation exhibited inconsistent performance. For some scenarios, this was the best representation and for others this was the worst. Since we would like a stable representation, which works across different scenarios, we omit this representation in our analyis. We also noticed that using the square root distance on the sphere always marginally lowered the recognition rate. Hence, we present results based only on the χ^2 distance.

As a baseline for comparison, we categorize the different scenarios using the existing single dynamic texture framework as outlined in [2, 12]. We model the entire video sequence using a single LDS. We calculate distances between LDSs using the Martin distance and categorize using both the nearest neighbor and SVM classifiers. We abbreviate these methods as DK and DS respectively. The methods based on the TF representation are abbreviated as TF-1NN, TF-3NN for the k-NN using $k = 1$ and $k = 3$ neighbors, and TF-SVM for the SVM classification. Similarly the methods using the SW representation are abbreviated as SW-1NN, SW-3NN and SW-SVM. Note that for the naive Bayes classifier there is no representation (TF or SW) that is required.

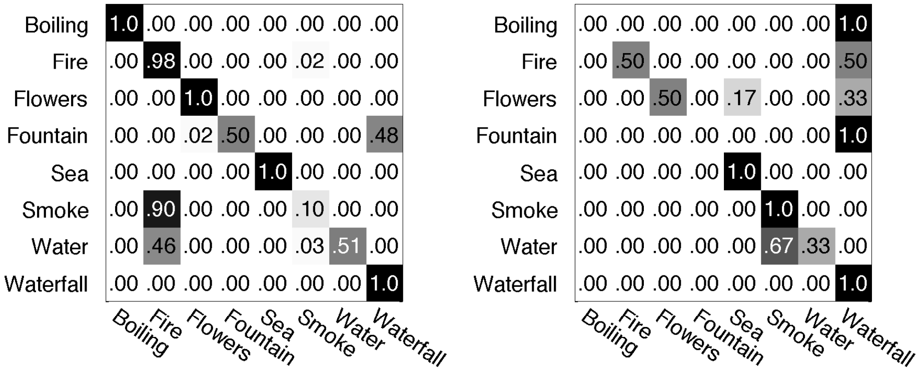
Water vs. Fountain. These classes are visually very different. Hence, we expect high classification rates for this problem. We consider 12 video sequences from the two classes and train using 50% of the data and test using the rest. The classification results for the various metrics are



(a) Results using our approach.

(b) Results using [12]

Figure 3. Classification results for the different scenarios.



	Boiling	Fire	Flowers	Fountain	Sea	Smoke	Water	Waterfall
Boiling	1.0	.00	.00	.00	.00	.00	.00	.00
Fire	.00	.98	.00	.00	.00	.02	.00	.00
Flowers	.00	.00	1.0	.00	.00	.00	.00	.00
Fountain	.00	.00	.02	.50	.00	.00	.00	.48
Sea	.00	.00	.00	.00	1.0	.00	.00	.00
Smoke	.00	.90	.00	.00	.00	.10	.00	.00
Water	.00	.46	.00	.00	.00	.03	.51	.00
Waterfall	.00	.00	.00	.00	.00	.00	.00	1.0

(a) Our approach

	Boiling	Fire	Flowers	Fountain	Sea	Smoke	Water	Waterfall
Boiling	.00	.00	.00	.00	.00	.00	.00	1.0
Fire	.00	.50	.00	.00	.00	.00	.00	.50
Flowers	.00	.00	.50	.00	.17	.00	.00	.33
Fountain	.00	.00	.00	.00	.00	.00	.00	1.0
Sea	.00	.00	.00	.00	1.0	.00	.00	.00
Smoke	.00	.00	.00	.00	.00	1.0	.00	.00
Water	.00	.00	.00	.00	.00	.67	.33	.00
Waterfall	.00	.00	.00	.00	.00	.00	.00	1.0

(b) Approach from [12]

Figure 4. Confusion matrix for the 8 class scenario.

shown in Fig. 3. Although DK and DS are not designed to be invariant to view points, we notice that their performance on this scenario are 67% and 100% respectively. Our proposed method is also able to achieve 100% classification rate.

Fountain vs. Waterfall. This is a more challenging example in which there are no clear appearance differences. We consider 16 video sequences for each class from the database. We train using 50% of the data and test using the rest. It must be noted that in this scenario, the testing involved video sequence from novel viewpoints, which were not used in the training. The classification results for the various metrics are shown in Fig. 3. The baseline results for this case are 50% using DK and 56.25% using DS. Our results for this scenario significantly outperform the baseline at 98%.

Notice from the two class scenarios that the best method to use is the naive Bayes approach. The SW-SVM approach is the second best method and outperforms the baseline. However, using the naive Bayes for more than two classes results in poor performance, while the SW-SVM still performs well, as we will results shown below.

Four Class. This scenario combines the categories from the above two scenarios along with the sea category. We use a total of 12 sequences per class and train using 50% of the data and test using the rest. The classification results for this scenario are shown in Fig. 3. We see that the proposed method is still able to outperform the baseline of 75% and 83% using DK and DS respectively. The best classification result we obtain using the BoS approach is 89%. In this scenario, one can notice that the increase in the classification rate is only 6%. This is primarily because there is only one novel viewpoint in the testing phase. Although the baseline methods are not invariant to changes in view point, in this database they are extremely effective in recognizing video sequences in the same viewpoint.

Eight Class. Our last and most challenging scenario is using all the classes from the database. We have a total of 88 video sequences with varying number of sequences per class. Similar to the other scenarios, we trained on 50% of the dataset and tested on the rest. The classification rates for this scenario are shown in Fig. 3. Our approach has a classification rate of 80%, while DK and DS are at 52.27% and 47.73% respectively. The confusion matrix for BoS and DK is shown in Fig. 4. In our methods, the smoke, fountain and water categories exhibit lower classification rates than the other categories. Except for the case of the water category, the confusion is between closely related categories. The DK method on the other hand, shows poor performance for all categories except sea, smoke and waterfall. In fact, most of the categories are wrongly recognized as waterfall.

5. Conclusions

In this paper, we have presented a method for the categorization of dynamic textures which is invariant to view point changes. Traditional methods cannot handle this variability and this fact is clearly reflected in the results we have presented. For the easy two class problem, our approach performs as well as the traditional methods. While for the hard two class problem, we show a gain of approximately 42%. In the case of the multiclass scenarios, the increase in performance is approximately 6% and 28% for four and eight classes, respectively. We achieve this by modeling a video sequence with a bag of of LDSs as opposed to using a single model for the video. To the best of our knowledge, no other existing work addresses this problem.

We have also evaluated the various weighting schemes. Based on our experiments, we observed that for a two class problem, using the naive Bayes classifier is the best approach. However, for higher number of classes using the SW with SVM is the best approach.

Acknowledgements

This work was partially supported by startup funds from JHU, by grants ONR N00014-05-10836, ONR N00014-09-1-0084, NSF CAREER 0447739 and ARL Robotics-CTA 80014MC.

References

[1] A. Chan and N. Vasconcelos. Probabilistic kernels for the classification of auto-regressive visual processes. In *CVPR*, volume 1, pages 846–851, 2005.

[2] A. Chan and N. Vasconcelos. Classifying video with kernel dynamic textures. In *CVPR*, pages 1–6, 2007.

[3] T. F. Cox and M. A. A. Cox. *Multidimensional Scaling*. Chapman and Hall, 1994.

[4] C. Dance, J. Willamowski, L. Fan, C. Bray, and G. Csurka. Visual categorization with bags of keypoints. In *ECCV*, 2004.

[5] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *VS-PETS*, October 2005.

[6] G. Doretto, A. Chiuso, Y. Wu, and S. Soatto. Dynamic textures. *IJCV*, 51(2):91–109, 2003.

[7] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Wiley-Interscience, October 2004.

[8] K. Fujita and S. Nayar. Recognition of Dynamic Textures using Impulse Responses of State Variables. In *Texture 2003*, Oct 2003.

[9] I. Laptev. On space-time interest points. *IJCV*, 64(2-3):107–123, 2005.

[10] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *CVPR*, pages 2161–2168, 2006.

[11] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[12] P. Saisan, G. Doretto, Y. N. Wu, and S. Soatto. Dynamic texture recognition. In *CVPR*, volume II, pages 58–63, 2001.

[13] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[14] R. Vidal and P. Favaro. Dynamicboost: Boosting time series generated by dynamical systems. In *ICCV*, 2007.

[15] S. Vishwanathan, A. Smola, and R. Vidal. Binet-Cauchy kernels on dynamical systems and its application to the analysis of dynamic scenes. *IJCV*, 73(1):95–119, 2007.

[16] G. Willems, T. Tuytelaars, and L. J. V. Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. In *ECCV*, volume 5303, pages 650–663, 2008.

[17] S.-F. Wong and R. Cipolla. Extracting spatiotemporal interest points using global information. In *ICCV*, pages 1–8, 2007.

[18] F. Woolfe and A. Fitzgibbon. Shift-invariant dynamic texture recognition. In *ECCV*, pages II: 549–562, 2006.