A Unified Approach to Segmentation and Categorization of Dynamic Textures

Avinash Ravichandran[1], Paolo Favaro[2] and René Vidal[1]

[1]Center for Imaging Science, Johns Hopkins University, Baltimore, MD, USA
[2]Dept. of Electrical Engineering and Physics, Heriot-Watt University, Edinburgh, UK

Abstract. Dynamic textures (DT) are videos of non-rigid dynamical objects, such as fire and waves, which constantly change their shape and appearance over time. Most of the prior work on DT analysis dealt with the classification of videos of a single DT or the segmentation of videos containing multiple DTs. In this paper, we consider the problem of joint segmentation and categorization of videos of multiple DTs under varying viewpoint, scale, and illumination conditions. We formulate this problem of assigning a class label to each pixel in the video as the minimization of an energy functional composed of two terms. The first term measures the cost of assigning a DT category to each pixel. For this purpose, we introduce a *bag of dynamic appearance features* (BoDAF) approach, in which we fit each video with a linear dynamical system (LDS) and use features extracted from the parameters of the LDS for classification. This BoDAF approach can be applied to the whole video, thus providing a framework for classifying videos of a single DT, or to image patches (superpixels), thus providing the cost of assigning a DT category to each pixel. The second term is a spatial regularization cost that encourages nearby pixels to have the same label. The minimization of this energy functional is carried out using the random walker algorithm. Experiments on existing databases of a single DT demonstrate the superiority of our BoDAF approach with respect to state-of-the art methods. To the best of our knowledge, the problem of joint segmentation and categorization of videos of multiple DTs has not been addressed before, hence there is no standard database to test our method. We therefore introduce a new database of videos annotated at the pixel level and evaluate our approach on this database with promising results.

1 Introduction

Dynamic textures (DT) are video sequences of non-rigid dynamical objects that constantly change their shape and appearance over time. Some examples of dynamic textures are video sequences of fire, smoke, crowds, and traffic. This class of video sequences is especially interesting since they are ubiquitous in our natural environment. For most practical applications involving DTs, such as surveillance, video retrieval, and video editing, one is interested in finding the types of DTs in the video sequences (categorization) and the regions that these DTs occupy (segmentation). For example, in the case of surveillance one wants to know

whether a fire has started or not, and if so, its location and volume. Specifically, given a video sequence that contains multiple instances of multiple DTs, in this paper we are interested in obtaining, for each pixel in the image, a label for its category. This task is referred to as *joint segmentation and categorization*.

Most of the existing works on DT analysis have addressed either categorization alone [1–5] or segmentation alone [6–10]. Existing categorization algorithms proceed by first modeling the entire video sequence using a single linear dynamical system (LDS). The parameters of these LDSs are then compared using several metrics such as subspace angles [6], KL-Divergence [7], Binet-Cauchy kernels [8], etc. The classification is then done by combining these metrics with classification techniques such as k-nearest neighbors (k-NN) or support vector machines (SVMs). One of the biggest shortcoming of these approaches is that the metrics used for comparing DTS are not designed to be invariant to changes in viewpoint, scale, etc. As a consequence, these methods perform poorly when a single class contains DTs with such variabilities. In [10], we addressed this issue by using multiple local models instead of a single global model for the video sequence. This bag-of-systems (BoS) approach was shown to be better at handling variability in imaging conditions of the data. Nonetheless, although there exists a large body of work addressing the problem of categorizing DTs, these methods suffer from several drawbacks. Firstly, all the approaches mentioned above assume that the video sequence contains a single DT. Hence, these methods cannot directly be used for video sequences with multiple DTs. Secondly, the choice of the metric used in these approaches requires that the training and testing data have the same number of pixels. This poses a challenge when one wants to compare local regions of a video sequence and adds additional overhead for normalizing all video sequences to the same spatial size.

On the other hand, existing DT segmentation algorithms model the video sequence locally at a patch level [3] or at a pixel level [1, 2, 4, 5] as the output of an LDS. The parameters of these LDSs are used as cues for segmentation as opposed to using traditional features (e.g., intensity and texture). Such cues are then clustered with additional spatial regularization that ensures contiguous segmentations. The segmentation is then obtained by iterating between clustering the LDS parameters and extracting new LDS parameters. Such an iterative procedure is solved using an expectation maximization approach [3–5] or a level set based approach [1, 2]. As a consequence, these algorithms require an initialization and can be sensitive to it.

While none of the above methods addresses the joint segmentation and categorization of DTs, one could leverage such techniques by first segmenting the video into multiple regions and then categorizing each region. In practice, however, this incurs in several problems. First, a manual initialization for the segmentation algorithm may not be available (especially for large databases). Second, since the segmentation results have been obtained without any relation to semantic categories, a single segment may contain multiple categories of interest. Finally, even if we were given an oracle segmentation algorithm, current categorization methods for DTs are all designed for classifying whole videos only.

In order to jointly segment and categorize video sequences containing multiple DTs, one would like an algorithm that requires no initialization. Clearly, the brute force alternative of searching through all possible segmentations is also unfeasible, especially when considering several categories. Furthermore, the algorithm should be able to explicitly handle intra class variability, i.e., changes in viewpoint, scale, illumination, etc.

Paper Contributions. In this paper, we propose a scheme that possesses all the above characteristics. The following are our paper contributions.

1. We introduce a new classification framework that provides either *global classifiers* for categorizing an entire video sequence or *local classifiers* for categorizing a region in the video sequence. This framework explicitly accounts for the invariance to changes in imaging conditions such as viewpoint, scale, etc. We propose a bag of dynamic appearance features (BoDAF) representation that uses SIFT features extracted from the parameters of the LDS. Unlike the case of image categorization, where the bag-of-features (BoF) approach results in one histogram per image, in our case we have multiple histograms per video. As we will show later, this is due to the fact that we are comparing the parameters of the LDSs using the BoDAF approach. Since there are several parameters for a single LDS, this gives rise to multiple histograms. This poses challenges in adopting the BoF approach directly. Hence, we propose a method to that extends the BoF approach to multiple histograms.
2. We then propose a joint categorization and segmentation algorithm for DTs. To the best of our knowledge, there does not exists any other work that handles both the segmentation and categorization simultaneously for DTs. Our joint segmentation and categorization approach proceeds by dividing the video sequences into superpixels by adapting existing techniques. In doing so, we assume that the superpixels do not change as a function of time. Each superpixel is then classified using local classifiers, which we will introduce in this paper. Finally, since objects are spatially contiguous, we introduce a spatial regularization framework. The joint segmentation and categorization problem is then posed as a multi-label optimization problem and solved using the random walker framework. The weights of the random walker algorithm are defined based on two terms: the output of the classifier for each superpixel and the similarity between adjacent superpixels.
3. Our final contribution is to provide a large database of dynamic textures annotated with accurate spatial labeling. We hope that this database can be used in the future as a benchmark for other algorithms.

We wish to point out that the approach described above is completely different from [10] where we use a bag-of-systems approach to categorize video sequences of DTs. In our approach, we use a single global LDS and extract features from the parameters of this model as opposed to using multiple local LDSs to model a video sequence as in [10]. Additionally, we propose a new method to compare DTs, while [10] uses the traditional subspace angles as the metric of comparison.

2 Categorization of Videos of a Single Dynamic Texture using a Bag of Dynamic Appearance Features

In this section, we review the linear dynamical system (LDS) model that we use for video categorization. Then, we propose a classification scheme which is based on features from the LDS parameters. As we will see in §3, these classifiers will be the building blocks for our joint categorization and recognition framework.

2.1 Dynamic Texture Model

We denote a video sequence of a DT as $V = \{I(t)\}_{t=1}^{F}$, where F is the number of frames. This video sequence is modeled as the output of a LDS given by

$$z(t+1) = Az(t) + Bv(t) \quad \text{and} \quad I(t) = C^0 + Cz(t) + w(t), \tag{1}$$

where $z(t) \in \mathbb{R}^n$ is the hidden state at time t, $A \in \mathbb{R}^{n \times n}$ is the matrix representing the dynamics of the hidden state, $B \in \mathbb{R}^{n \times m}$ is the input-to-state matrix, $C \in \mathbb{R}^{p \times n}$ maps the hidden state to the output of the system, C^0 represents the mean image of the video sequence and $w(t)$ and $v(t)$ are the zero-mean measurement noise and process noise, respectively. The order of the system is given by n, and the number of pixels in each frame of the sequence is given by p . Notice that the i^{th} column of C, C^i is the vectorization of an image with p pixels. Hence, the matrix C can be also considered as a basis for the video frames consisting of n images. These images along with the mean image are called *dynamic appearance images* (DAI) i.e., $\mathcal{C} = \{C^0, C^1, \dots C^n\}$.

The parameters of the LDS that model a given video sequence can be identified using a suboptimal approach as outlined in [11]. However, the identified system parameters are unique only up to a change of basis. This basis ambiguity can be overcome by using a canonical form. In this paper, we use the Jordan Canonical form (JCF) outlined in [12]. In what follows, we show how we compare video sequences using the parameters of their corresponding LDS.

2.2 Representation and Comparison of LDSs

We assume that we are given N video sequences $\{V_j\}_{j=1}^{N}$, each of which belongs to one of L classes. Most categorization algorithms for DTs rely on analyzing the parameters of the LDS that model the video sequences. Traditional methods [6–8] compare the parameters (A_j, C_j) of the LDS by using several existing metrics as outlined in §1. However, one major drawback of such metrics is that they do not necessarily capture any semantic information and hence they might not be meaningful in the context of categorization. For instance, two videos from two different classes might be judged similar because their parameters are close in the chosen metric. However, two videos from the same class under different viewpoint, scale, and illumination conditions might be judged different. Previous methods [10] have addressed the latter problem by using multiple local models and by assuming that viewpoint changes are negligible at small scales. In this

paper we follow an alternative approach. We show that variability in viewpoint, scale, etc., can be accounted for with a mix of global and local representations.

In our approach we extract features at each pixel of each DAI. To achieve invariance to scale, orientation, and partially to illumination changes, we consider SIFT features [13]. When extracting such features we do not use any interest point detector to prune them, because we do not know in advance which regions are discriminative. While these features account for the local characteristics of a DT, we also need to capture the global statistics of the DT and provide a common representation for comparing different DTs. One such representation is the bag-of-features (BoF) approach [14, 15]. In a BoF framework, features extracted from an image are first quantized using a *visual codebook* and then used to compute a histogram of codeword occurrences. Since the histograms are assumed to follow a characteristic distribution for each class, they are used as features for classification. This approach is very popular for object recognition due to its simplicity and fairly good performance.

In our case, rather than comparing two images, we need to compare two C matrices, which are two sets of images. This introduces several challenges in adopting the BoF scheme, especially in the formation of the codebook and the quantization of features. The simplest way to form a codebook is to cluster all the features extracted from all the DAIs. This would result in a single codebook for all the DAIs. However, each of the DAIs captures a different piece of information. The mean C^0, for example, contains the low frequency texture information. In contrast, as the order n of the LDS increases, the corresponding DAI, C^n, describes higher frequency details of the textures. Hence, a single codebook would dilute the differences between the details captured by each DAI. To prevent this from happening, we build separate codebooks for each of the DAIs. We use K-means to cluster the features from a DAI to form the codebook for that DAI.

Let the codebooks have size K for each DAI. We represent each DAI using histograms. Such histograms can be calculated in several ways from the features in the DAI and the codebook. The most common choice is called the *term frequency* (TF) [14]. This approach essentially counts the number of codewords that occur in each DAI. Another approach is called the *term frequency - inverse document frequency* (TF-IDF) [15]. This approach discounts the codewords that occur in all the images and concentrates on the ones that occur less frequently. We refer the reader to [14, 15] for details about these approaches. Thus, for every video sequence V_j and its associated set of appearance images $\mathcal{C}_j = \{C_j^i\}_{i=0}^n$, we represent the appearance parameters using the set of histograms $\mathcal{H}_j = \{h_j^i \in \mathbb{R}_+^K\}_{i=0}^n$. Therefore, instead of $(A_j, \mathcal{C}_j)$, which was traditionally used for categorization, we now have $(A_j, \mathcal{H}_j)$.

2.3 Classifiers

The first step towards building a classifier for DTs is to define a distance to compare the representations $M_j = (A_j, \mathcal{H}_j)$ of each pair of video sequences. Among the several existing methods for comparing histograms, popular choices

include the χ^2 distance and the square root distance, which are given by

$$d_{\chi^2}(h_1, h_2) = \frac{1}{2}\sum_{k=1}^{K}\frac{|h_{1k} - h_{2k}|^2}{h_{1k} + h_{2k}} \quad \text{and} \quad d_{\surd}(h_1, h_2) = \cos^{-1}(\sum_{i=1}^{K}\sqrt{h_{1k}h_{2k}}), \quad (2)$$

respectively, where h_{jk} denotes the k^{th} element of the histogram vector $h_j \in \mathbb{R}^K$. As for comparing the A matrices, since they are already in their canonical form, one could use any preferred choice of the matrix norm. Also, one can form a kernel from these distances by using a radial basis function (RBF). Such a kernel is given by $\mathcal{K}(x, y) = e^{-\gamma d(x,y)}$, where γ is a free parameter.

In order to build a single classifier for multiple histograms, we first explored a multiple kernel learning (MKL) framework [16]. In this framework, one is given multiple kernels $\{\mathcal{K}_i\}_{i=1}^{n_b}$, which are referred to as basis kernels. These kernels can be, for example, the kernels between each DAI histogram or the product of the kernels between all the DAI histograms, or the kernel between the A parameters. Given these kernels, a single kernel is built as $\mathcal{K} = \sum_{k=1}^{n_b} w_k\mathcal{K}_k$, by choosing a set of weights such that $\sum w_k = 1$. The weights can be automatically learned from the data by training an SVM classifier. We refer the reader to [16] for more details on this method.

When we used the MKL algorithm to obtain the weights, we noticed that, in practice, the weight corresponding to the kernel based on the matrix A and the weight corresponding to the kernels based on individual DAI histograms were nearly zero, while the weight corresponding to the product kernel was nearly 1. For this reason we decided to disregard the kernel on the A matrix and the kernels on individual DAI histograms, and adopted the product kernel on the DAI histograms. One can show that for RBF kernels, using a product kernel is equivalent to concatenating the histograms into one single vector $H_j \in \mathbb{R}^{(n+1)K}$. Hence, we choose the concatenated version of the histogram as our representation for $\mathcal{H}_j$.

As for the choice of the classifiers for the histograms, we can use discriminative classifiers such as k-NNs or SVMs for the classification. The kernel of choice for SVMs is the RBF kernel, which we introduced earlier on. Alternatively, one could use a generative approach for the classifier. One such classification algorithm is the naïve Bayes classifier [14]. This algorithm works directly with the codeword occurrence as opposed to using the histogram representation. We refer the readers to [14] for details on implementing such a classifier.

3 Joint Segmentation and Categorization of Videos of Multiple Dynamic Textures

The classifiers described so far have been defined as *global classifiers* for the purpose of categorizing videos containing a single DT. However, in this paper we are interested in categorizing video sequences containing multiple DTs. In this section, we formulate a solution to this problem under the simplifying assumption

that the class of a pixel does not change as a function of time, so that the segmentation of the video into multiple DTs is common across all the frames.

When this segmentation is known, one can easily adapt the method described in §2 so that it can be applied to classifying each segmented region. More specifically, we can fit a single LDS to the video sequence as before, and then form multiple concatenated histograms of DAIs, one per region, by restricting the feature counts to the features within each region. Afterwards, each region can be classified independently based on its histograms. In general, however, the segmentation is not known and one needs to address both problems simultaneously.

In what follows, we describe the proposed framework for joint segmentation and categorization of DTs, which consists of three main ingredients: dividing the video sequence into superpixels, constructing a local classifier for each superpixels, and finding the segmentation and categorization of the video by minimizing an energy functional defined on the graph of superpixels.

3.1 Superpixels

The first step in our approach is to divide each video into superpixels. This step is required as the decision of whether a single pixel belongs to one class or another cannot be made based on the information from a single pixel. This calls for exploring supports over larger regions. However, searching through the space of all admissible supports drastically increases the complexity of the algorithm. One solution is to use a fixed window around each pixel. However, a fixed window could contain pixels from two different DTs. A better solution is to choose regions that are unlikely to contain two different DTs. This issue can be resolved by using superpixels, i.e., image patches, that are naturally defined by the data rather than by the user. Also, since superpixels contain pixels that are similar, we can categorize the whole superpixel as opposed to classifying each pixel contained in it. This reduces the computational complexity of the method.

Our algorithm for computing superpixels builds a feature vector at each pixel by concatenating the values of the normalized DAIs with the 2D pixel location. This results in an $n + 3$ dimensional vector, where $n + 1$ components are from the DAIs. We then use the quick shift algorithm [17] to extract the superpixels from such features.

3.2 Classifiers

Given a training video sequence, we first obtain its LDS parameters. We then cluster the dense SIFT features extracted from the DAIs of all the training videos in order to form the codebooks. We then divide the DAIs into superpixels as describe above. By restricting the feature counts to the features within each superpixel, we can build *local histograms*. All the local histograms from all the DAIs of all the training videos are then used to train *local classifiers*.

Given a test video sequence, we compute its LDS parameters and DAIs as before. We then quantize the features within each superpixel to form a local

histogram. We then use the local classifiers to assign, to each pixel inside a superpixel, the cost of assigning it to each of the DT categories. This procedure already provides us with a segmentation and a categorization of the video. However, this segmentation may not be smooth, since each superpixel is classified independently. Therefore, we find the joint segmentation and categorization of the video by minimizing an energy functional, which is the sum of the above classification cost plus a regularization cost, as described next.

3.3 Joint Segmentation and Categorization

We solve the joint segmentation and categorization problem using a graph theoretic framework. We define a weighted graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the set of nodes and $\mathcal{E}$ denotes the set of edges. The set of nodes $\mathcal{V} = \{s_i\}_{i=1}^{N_s} \cup \{g_l\}_{i=1}^{L}$ is given by the N_s superpixels obtained from the DAIs, $\{s_j\}_{j=1}^{N_s}$, and by L auxiliary nodes, $\{g_l\}_{l=1}^{L}$, one per each DT category. The set of edges $\mathcal{E} = \mathcal{E}_s \cup \mathcal{E}_c$ is given by the edges in $\mathcal{E}_s = \{(i,j) | s_i \sim s_j\}$, which connect neighboring superpixels $s_i \sim s_j$, and by the edges in $\mathcal{E}_c = \{(i,l) | s_i \sim g_l\}$, which connect each s_i to every g_l.

We also associate a weight to every edge, which we denote by w_{ij} if $(i,j) \in \mathcal{E}_s$ and by $\tilde{w}_{ij}$ if $(i,j) \in \mathcal{E}_c$. We set $\tilde{w}_{il}$ to be the probability that superpixel s_i belongs to class l, $P(c_l|s_j)$, as obtained from the trained classifiers. We restrict our attention to the SVM and the naïve Bayes classifiers for this case. For weights between s_i and s_j, we set $w_{ij} = \mathcal{K}_\chi(H_{s_i}, H_{s_j})$, where $\mathcal{K}_\chi$ is defined as

$$K_\chi(H_{s_i}, H_{s_j}) = 2 \sum_{m=1}^{K(n+1)} \frac{H_{s_i}^m \cdot H_{s_j}^m}{H_{s_i}^m + H_{s_j}^m}, \tag{3}$$

where H_{s_i} is the concatenated histogram from superpixel s_i. This is a kernel derived from the χ^2 distance. This weight measures the similarity of the concatenated histograms between the neighboring superpixels in order to enforce the spatial continuity. In order to have the flexibility to control the influence of the prior and the smoothing terms, we introduce two other parameters $\alpha_1 > 0$ and $\alpha_2 > 0$ and redefine the weights as $\tilde{w}_{ij} = \alpha_1 P(c_j|s_j)$ and $w_{ij} = \alpha_2 \mathcal{K}_\chi(H_{s_i}, H_{s_j})$. Varying these parameters will determine the contribution of these weights.

In order to solve the joint segmentation and categorization problem, we associate with each node s_i, a vector $\mathbf{x}_i = \left[x_i^1, \cdots, x_i^L\right]$, where x_i^l is the probability that s_i belongs to class l. For the auxiliary nodes g_l, this probability is denoted by $\tilde{x}_l^i = \delta(l-i)$, where $\delta(\cdot)$ is the discrete Dirac delta. A label y_i for each node is then obtained via $y_i = \arg\max_{y \in \{1,\ldots,L\}} x_i^y$. These labels can be estimated using several techniques such as graph cuts [18] and the random walker [19]. In this work, we propose to solve this problem using a random walker (RW) formulation as this formulation generalizes more easily to the multi-label case.

In the RW framework, the nodes g_l are considered as terminal nodes. A RW starting at the node s_i can move to one of its neighbors s_j with a probability $w_{ij} / \sum_{k \in \mathcal{N}_i} w_{ik}$, where $\mathcal{N}_i$ is the set of neighbors of s_i. We are interested in the probability that a RW starting from s_i reaches first the node g_l. This probability

is precisely the value x_i^l. It can be shown that the value x_i^l can be obtained by minimizing the following cost function

$$E(\mathbf{x}) = \sum_{l=1}^{L} \Bigg[\sum_{(i,j)\in\mathcal{E}_c} \tilde{w}_{il}(x_i^l - \tilde{x}_j^l)^2 + \sum_{(i,j)\in\mathcal{E}_s} w_{ij}(x_i^l - x_j^l)^2 \Bigg], \text{ s.t. } \forall i \sum_{l=1}^{L} x_i^l = 1. \quad (4)$$

As shown in [19], the minimizer of this cost function can be obtained by solving $L-1$ linear systems. Since by construction $w_{ij} \geq 0$ and $\tilde{w}_{il} \geq 0$, it can be shown that our cost function is convex and has a unique global minimum. We refer the reader to [19] for details on how this function can be minimized.

4 Experiments

This section presents two sets of experiments, which evaluate the proposed framework on the categorization of video sequences of a single DT, and on the joint segmentation and categorization of video sequences of multiple DTs. For the first task we use existing datasets and for the second one we propose a new dataset.

UCLA-8 [10]: This data set is a reorganized version of the UCLA-50 dataset [6] and consists of 88 video sequences spanning 8 classes. Here each class contains the same semantic texture (e.g., water), but under different conditions such as viewpoint and scale. Although this dataset consists of fewer classes than UCLA-50, this dataset is much more challenging than the UCLA-50.

Traffic Database [7]: This dataset consists of 254 videos spanning 3 classes (heavy, medium and light traffic) taken from a single fixed camera. The videos contain traffic conditions on a highway at different times of the day and under different weather conditions. However, the clips share the same viewpoint and area on the highway. We use this dataset to highlight the fact that our classification algorithm works even when the variations between the classes are subtle with respect to the appearance and large with respect to the temporal process.

Dyntex [20]: Existing datasets, i.e., UCLA-8 and Traffic, are not well-suited for testing our joint segmentation and categorization algorithm. This is because most of the video sequences in UCLA-8 contain only a single texture and seldom any background. As for the traffic dataset, since every video sequence shares the same view point, all video sequences have almost the same annotation. However, the Dyntex dataset is well-suited for our purposes. This dataset consists of 656 video sequences, without annotation either at the pixel level or at the video level. Even though there is a large amount of video sequences, there are several issues with adopting the dataset for testing. The first one is that this dataset contains video sequences that are taken by a moving camera. Such video sequences are beyond the scope of this paper. It also contains video sequences that are not DTs. Moreover, several of the classes in this dataset have very few samples, e.g., 2-3 video sequences, which is too little to be statistically significant. Hence, we used the 3 largest classes we could obtain from this dataset: waves, flags and fountains. This gave us 119 video sequences, which we manually annotated at the pixel level. Sample annotation from the dataset can be found in Figure 1.

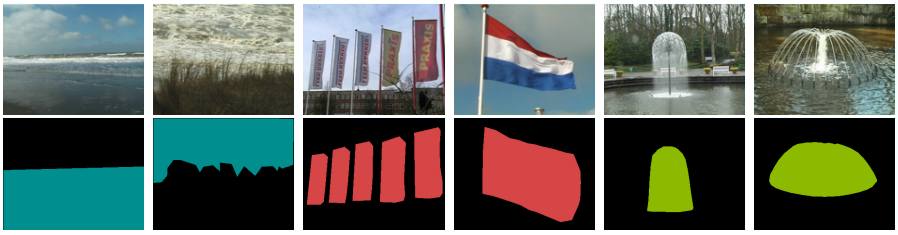

Fig. 1. Sample annotations for the subset of the Dyntex dataset we use for the joint segmentation and categorization. (Color convention: Black for background, red for flags, green for fountain and blue for waves.)

4.1 Whole Video Categorization

We now show experimental results on the UCLA-8 and traffic databases, which contain video sequences of a single DT. As mentioned earlier, we used SIFT features extracted from the DAIs. We extracted these features on a dense regular grid with a patch size of 12×12 as opposed to using interest points. We also tested our algorithm on descriptors extracted using interest points and we observed that the performance was significantly lower than with dense SIFT features.

We used both the TF and the TF-IDF representations to obtain the histogram of each DAI from its features and the codebook. However, our experimental results showed that the TF representation performs better than the TF-IDF representation. Hence, all results presented in this section are obtained using the TF approach. For the classifiers, we used k-NN with $k = 1$ and $k = 3$ using d_{χ^2} and $d_{\surd}$ as the distance metric. We denote these classifiers as 1NN- TF-χ^2, 1NN-TF- $\surd$, 3NN-TF-χ^2 and 3NN-TF-$\surd$. We also used the SVM and naïve Bayes (NB) classifier. The kernel used for the SVM was the RBF kernel based on the χ^2 distance. The free parameters for the SVM classifier were obtained using a 2-fold cross validation on the training dataset.

UCLA-8: We split the data into 50% training and 50% testing as in [10]. We modeled the video sequences using LDSs of order $n = 3$ and did not optimize this parameter. All the results reported here are averaged over 20 runs of our algorithm. This is because in every run of our algorithms the codewords can change. This is attributed to the convergence properties of the K-Means algorithms.

Table 1 shows the results of the different classifiers on UCLA-8. This table shows the maximum categorization performance over all choices of cluster sizes. From this table we observe that the NB algorithm outperforms all the other classifiers. We compared our method with the bag-of-systems (BoS) approach outlined in [10] and the method of using a single global model and the Martin distance as outlined in [6]. The performance of our method is at 100% as opposed to 80% using the BoS and 52% using a global LDS and the Martin distance.

Also shown in Table 1 is the performance of just using an individual vector of the DAI, i.e., we use only one h_j^i as opposed to using the concatenated H_j. Table 1 shows these results under the columns titled C^i, $i = 0 \ldots 3$. From this table we see that, although there exists a basis that performs well, its performance

Table 1. Categorization performance on UCLA-8 using different approaches

Classifier	C^0	C^1	C^2	C^3	Our Approach	Concat Frames
1NN-TF-$\sqrt{}$	70	82	90	85	**97**	80
1NN-TF-χ^2	70	81	90	83	**94**	79
3NN-TF-$\sqrt{}$	71	77	93	83	**97**	80
3NN-TF-χ^2	71	77	91	82	**93**	81
NB	72	80	91	88	**100**	82
SVM	64	76	90	78	**95**	80

does not match that of our approach. This shows that our approach is able to successfully combine the information from all the DAIs. One could argue that since these results are based on the DAI, the dataset could be classified only using the texture information, i.e., using the frames of the video sequence themselves. To address this issue, we substituted $n+1$ video frames in lieu of the $n+1$ DAIs in our classification framework. These frames were chosen at regular time intervals from the video sequence. The results of using such an approach are summarized in Table 1 under the column Concat Frames. We see that the performance of this approach is about 80% in all the cases. This accuracy is comparable to that of the BoS approach. This shows that comparing the parameters as opposed to comparing only the frames of the video sequences increases the categorization performance.

Traffic Dataset: For the Traffic dataset, we used the same experimental conditions as in [7], where the data was split into 4 different sets of 75% training and 25% testing. The maximum performance reported in [7] for this dataset is 94.5% for an order of $n = 15$. We report our results averaged over 10 runs of our algorithm using a codebook size of $K = 42$ for each of the DAI. Figure 2 shows the performance of our algorithm for different classifiers and different orders of the LDSs. It can be seen that for this dataset, we obtain a categorization performance of 95.6% for $n = 7$. Although the increase with respect to the state of the art is not as large as in the case of the UCLA-8 dataset, we notice that: (a) For any given order, our classifiers outperform [7]; (b) Our maximum classification performance is obtained at almost half the order as that of [7]. This clearly exhibits the fact that our classification framework is more powerful than traditional methods of comparison and works well across different datasets.

4.2 Joint Segmentation and Categorization

We first describe some of our implementation details. We use approximately 50% of the data for training and testing. We identify a LDS of order $n = 3$ for each video sequence. We use the quick-shift algorithm for extracting superpixels. This algorithm has three parameters: λ which controls the tradeoff between the spatial importance and the importance of the DAI, σ which is the scale of the kernel density estimation and τ which determines the maximum distance between the features of the members in the same region. We use the following parameters:

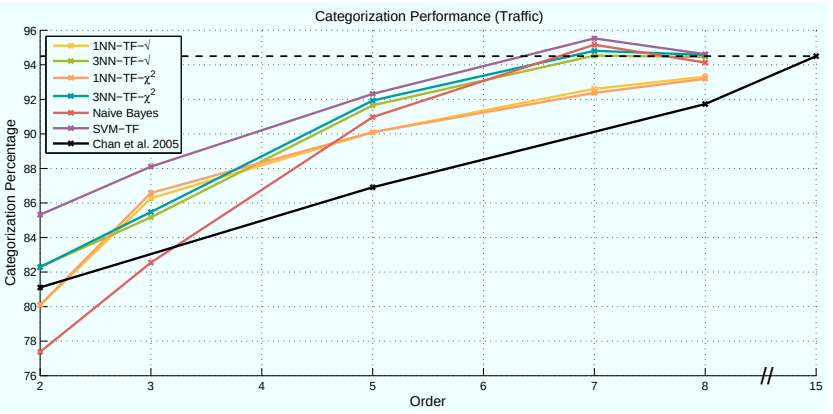

Fig. 2. Categorization performance of different classifiers on the Traffic Dataset of [7].

$\lambda = 0.25$, $\tau = 3$, $\sigma = 3$ for all our experiments. These values were chosen so that there were no extremely small superpixels. Using the manual annotations, we assign to each superpixel the label of the majority of its pixels. Concatenated histograms are then extracted from the DAIs by restricting the codeword bin counts to the support described by each superpixel. We then train both as SVM and a naïve Bayes classifier using these histograms. We used $K = 80$ as the size for the codebook for each DAI and $\alpha_1/\alpha_2 = 3/4$ for the RW) formulation. We report our results before and after applying the RW.

In order to compare our results with the ground truth segmentation, we use 3 metrics. For each class, we report the percentage of pixels across all the videos that are correctly assigned this class label. We also report our performance using the intersection/union metric. This metric is defined as the number of correctly labeled pixels of that class, divided by the number of pixels labeled with that class in either the ground truth labeling or the inferred labeling. The last metric is the average of the percentage of pixels that are correctly classified per video sequences (APV). The results of our algorithm are shown in Table 2. From this table we observe that the NB classifier does not perform well on the background class while the SVM classifier does not perform well on the fountain class. The SVM classifier overall is better over two metrics when compared to the NB classifier. Figure 3 shows a few sample segmentations from our dataset.

5 Conclusions

In this paper, we have proposed a method to address the problem of joint segmentation and categorization of DTs. To the best of our knowledge, this problem has not been addressed before. Instead of first segmenting the video sequence and then categorizing each segment, we provided a unified framework that can simultaneously perform both tasks. A key component in this framework is the new classification scheme that we have proposed. We have shown that this scheme

Table 2. Performance of Joint Segmentation and Categorization Algorithm

Method	Percentage Correctly Classified					Intersection/Union metric					APV
	Background	*Waves*	*Flags*	*Fountain*	*Average*	*Background*	*Waves*	*Flags*	*Fountain*	*Average*	
NB+RW	41.9	**99.6**	**77.1**	**71.3**	**72.5**	0.41	0.81	0.55	**0.30**	0.51	70.4
NB	47.5	97.2	70.8	59.5	68.8	0.46	0.79	0.48	0.25	0.50	69.6
SVM+RW	**87.5**	97.1	66.9	30.2	70.4	**0.69**	**0.87**	**0.60**	0.25	**0.60**	**81.7**
SVM	80.6	94.2	62.0	41.9	69.7	0.66	0.82	0.50	0.29	0.57	78.4

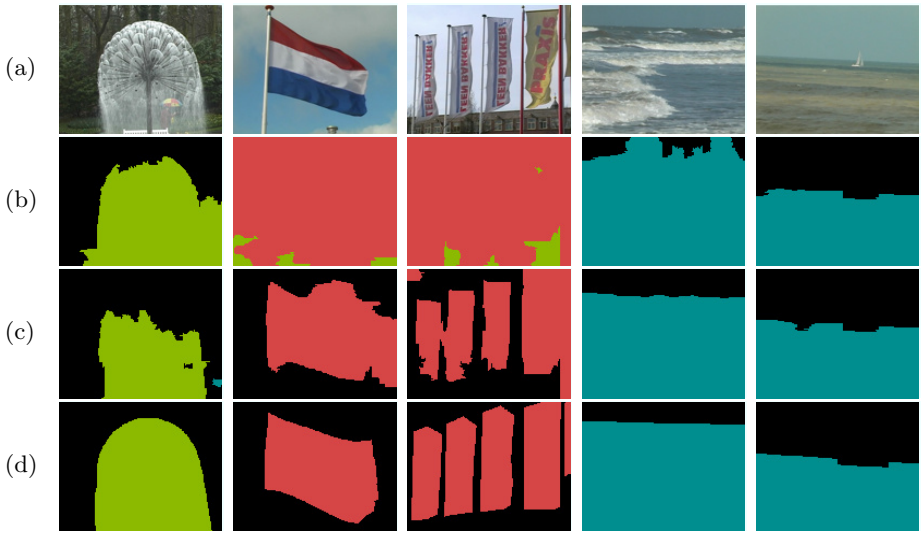

Fig. 3. Sample segmentations. (a) sample frame from the video sequence; (b) segmentation using Bayes + RW; (c) segmentation using SVM + RW; (d) ground truth segmentation (annotation).

performs very well for classifying whole videos and can also be used locally for our joint segmentation and categorization framework. Our results showed that by using this classifier, we outperformed existing methods for DT categorization. Since there are no available datasets to test our framework, we provided what is, to the best of our knowledge, the first dynamic texture database with annotations at the pixel level. We hope that this database can be used as a benchmark for future research.

Acknowledgements

The authors would like to thank Daniele Perone for his help with the annotations and Lucas Theis for his help with the experiments. This work was partially supported by grants ONR N00014-09-1-0084 and ONR N00014-09-1-1067.

References

1. Doretto, G., Cremers, D., Favaro, P., Soatto, S.: Dynamic texture segmentation. In: IEEE Int. Conf. on Computer Vision. (2003) 44–49
2. Ghoreyshi, A., Vidal, R.: Segmenting dynamic textures with Ising descriptors, ARX models and level sets. In: Intl. Workshop on Dynamic Vision. LNCS 4358 (2006) 127–141
3. Chan, A., Vasconcelos, N.: Modeling, clustering, and segmenting video with mixtures of dynamic textures. IEEE Trans. on Pattern Analysis and Machine Intelligence **30** (2008) 909–926
4. Chan, A., Vasconcelos, N.: Layered dynamic textures. IEEE Trans. on Pattern Analysis and Machine Intelligence (2009) 1862–1879
5. Chan, A., Vasconcelos, N.: Variational layered dynamic textures. IEEE Conf. on Computer Vision and Pattern Recognition (2009)
6. Saisan, P., Doretto, G., Wu, Y.N., Soatto, S.: Dynamic texture recognition. In: IEEE Conf. on Computer Vision and Pattern Recognition. Volume II. (2001) 58–63
7. Chan, A., Vasconcelos, N.: Probabilistic kernels for the classification of auto-regressive visual processes. In: IEEE Conf. on Computer Vision and Pattern Recognition. (2005) 846–851
8. Vishwanathan, S., Smola, A., Vidal, R.: Binet-Cauchy kernels on dynamical systems and its application to the analysis of dynamic scenes. Int. Journal of Computer Vision **73** (2007) 95–119
9. Vidal, R., Favaro, P.: Dynamicboost: Boosting time series generated by dynamical systems. In: IEEE Int. Conf. on Computer Vision. (2007)
10. Ravichandran, A., Chaudhry, R., Vidal, R.: View-invariant dynamic texture recognition using a bag of dynamical systems. In: IEEE Conf. on Computer Vision and Pattern Recognition. (2009)
11. Doretto, G., Chiuso, A., Wu, Y., Soatto, S.: Dynamic textures. Int. Journal of Computer Vision **51** (2003) 91–109
12. Ravichandran, A., Vidal, R.: Video registration using dynamic textures. In: European Conf. on Computer Vision. (2008)
13. Lowe, D.: Distinctive image features from scale-invariant keypoints. In: Int. Journal of Computer Vision. Volume 20. (2003) 91–110
14. Dance, C., Willamowski, J., Fan, L., Bray, C., Csurka, G.: Visual categorization with bags of keypoints. In: European Conf. on Computer Vision. (2004)
15. Sivic, J., Zisserman, A.: Video google: A text retrieval approach to object matching in videos. In: IEEE Int. Conf. on Computer Vision. (2003) 1470–1477
16. Rakotomamonjy, A., Bach, F., Canu, S., Grandvalet, Y.: Simplemkl. Journal of Machine Learning Research **9** (2008) 2491–2521
17. Vedaldi, A., Soatto, S.: Quick shift and kernel methods for mode seeking. In: European Conf. on Computer Vision. (2008) 705–718
18. Boykov, Y., Jolly, M.: Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. In: IEEE Int. Conf. on Computer Vision. (2001) 105–112
19. Grady, L.: Multilabel random walker image segmentation using prior models. In: IEEE Conf. on Computer Vision and Pattern Recognition. (2005) 763–770
20. Péteri, R., Huskies, M., Fazekas, S.: Dyntex: A comprehensive database of dynamic textures. (Online Dynamic Texture Database)