2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)
July 6-11, 2014, Beijing, China

Dynamic Texture Classification Using Local Fuzzy Coding

Liuyang Wang, Huaping Liu, and Fuchun Sun

***Abstract*—Recognition of complex dynamic texture is a challenging problem and captures the attention of the computer vision community for several decades. Essentially the dynamic texture recognition is a multi-class classification problem that has become a real challenge for computer vision and machine learning techniques. In this paper, we propose a new approach to tackle the dynamic texture recognition problem. First, we utilize the fuzzy clustering technology to design a fuzzy codebook, and then construct a soft assigned local fuzzy coding feature to represent the whole dynamic texture sequence. This new coding strategy preserves spatial and temporal characteristics of dynamic texture. Finally, by evaluating the proposed approach using with the DynTex dataset, we show the effectiveness of the proposed local fuzzy coding strategy.**

I. 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. Dynamic texture itself is very useful. It is widely used in texture synthesis [1], video registration [2], music modeling [3] and control education [4].

However, the classification of DT admits great challenging since DT include both spatial and temporal elements. To model DT, Ref. [1] developed a linear dynamic system (LDS) method. LDS can be used to model complex visual phenomena, such as smoke and waves in water, with a relatively low dimensional representation. The use of such a model allows for greater editing power, and the output sequences included images which are never a part of the original sequence. However, the outputs were blurry compared with those of non-procedural techniques. Moreover, the signal would rapidly decay and its intensity would become saturated. The work in [5] extended this work by introducing feedback control and modeled the system as a closed loop LDS. The feedback loop corrected the problem of signal decay. In [4], LDS was regarded as educational bridge to merge the gap between computer science and control engineering.

A difficult problem to use LDS for classification lies in the fact that LDS does not lie in an Euclidean space, and therefore many classification which is based on Euclidean distance and reconstruction cannot be utilized. In [1], the Martin distance between LDS is adopted to compare the similarity between different DTs. Martin distance, which was developed by control scientist, is effective to evaluate the distance between LDS, but cannot be effectively used for video with multiple dynamic textures. Therefore, many works which utilize Martin distance are limited to investigate simple video with single DT.

Very recently, Ref. [6] proposed to categorize DT using a novel Bag-Of-dynamic-Systems (BoS). It models each video sequence with a collection of LDSs, each one describing a small spatial-temporal patch extracted from the video. This BoS representation is analogous to the Bag-of-Words (BoW) representation for object recognition, except that it used LDSs as feature descriptors. This choice provides an effective strategy to deal with video sequences which are taken under different viewpoints or scales.

Because BoS model is similar to BoW, it naturally inherits the disadvantages of BoW. It is well known that BoW model assign only one codebook element to a descriptor, and therefore the quantization error is large. This usually degrades the classification performance. In [7] and [8], the sparse coding and local coding methods are proposed to address such problem. In such frameworks, more than one codebook element will be assigned to a descriptor and form coding vector. Such strategy can obviously improve the classification performance since the quantization error is attenuated.

Unfortunately, neither sparse coding nor local coding can be used for BoS. The intrinsic reason is that both methods depends on the linear subspace assumption and used the linear reconstruction error to design the object function. Since DT lies in non-Euclidean space, such reconstruction error cannot be adopted.

Very recently, the fuzzy logic was used for adjusting weights of dynamic texture features in background subtraction [9], but their work are restricted to feature selection only. Motivated by the fuzzy logic, which allows multiple memberships [10-13], we design a new coding method which only depends on the Martin distance instead of the reconstruction error. Such strategy preserves the advantage that more than one codebook element can be assigned to a descriptor, and the fuzzy membership function naturally serves to design the coding vector. Please note the fuzzy codebook has been developed in [14] but the proposed local fuzzy coding strategy in DT classification is new. In addition, we made extensive experiments on public dataset to show the advantages of the proposed method.

The rest of this paper is organized as follows: In Section 2 we give an overview of the proposed method. Section 3 gives a brief introduction about the DT and the Martin distance. Section 4 presents the details of the proposed method. Section

The authors are all with Department of Computer Science and Technology, Tsinghua University, State Key Laboratory of Intelligent Technology and Systems, TNLIST, China (e-mail: hpliu@tsinghua.edu.cn).

This work is jointly supported by the National Key Project for Basic Research of China (2013CB329403), the National Natural Science Foundation of China (Grants Nos: 91120011 and 61210013), Tsinghua Self-innovation Project (Grant No: 20111081111) and Tsinghua University Initiative Scientific Research Program (Grant No: 20131089295).

978-1-4799-2072-3/14/$31.00 ©2014 IEEE

5 shows the experimental results. In Section 6 we give some conclusions.

II. Overview

In this section, we present an overview of the proposed dynamic texture recognition method. The basic methodology flow is shown in Fig.1. First, we utilize the dense sampling method to extract local spatial-temporal patch from the training video sequences. Each spatial-temporal patch is modeled with a LDS which will be illustrated in Section III. After that, a fuzzy k-Medoids clustering method is used to obtain the fuzzy codebook. Each patch can be represented as feature vector which is constructed by the fuzzy membership function. The whole video sequence can be represented as a feature vector by pooling all of the feature vectors of the patches in this video.

After obtaining a feature vector, a classifier which is trained using the training data set can be used to obtain the label of the new video sequence. In this paper, a linear multi-class support vector machine is used to perform this task. The details can be found in Section IV.C.

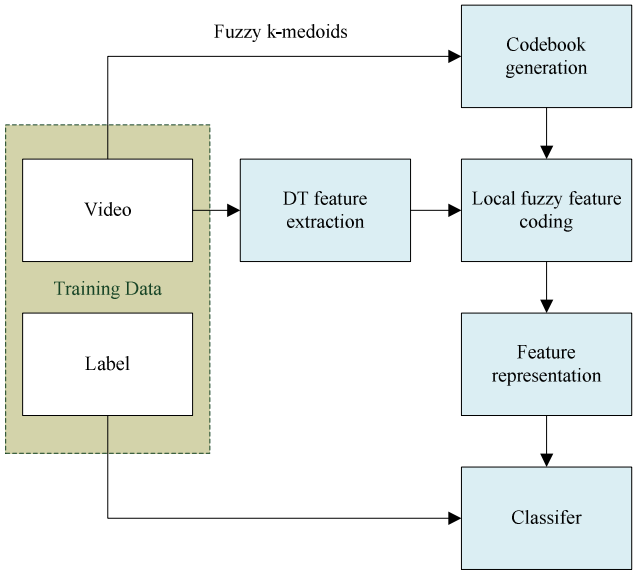

Fig.1: The overview of the proposed method.

III. Dynamic Texture Modelling

For self-containment, this section summarizes the key concepts in dynamic texture. According to [1], an LDS model can be used to fit a small spatial-temporal patch. Assume the short spatial-temporal patches includes τ frames with resolution $m \times n$. The image frames are vectorized as vector $\mathbf{y}(k) \in \mathbb{R}^{n_y}$ for $k = 1, 2, ..., \tau$, and $n_y = m \times n$.

The dynamic texture modeling can be seen as a system identification problem. The dynamic texture should obeys the standard state-space equation:

$$\begin{cases} \mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{u}(k), & \mathbf{u}(k) \sim N(0, \mathbf{Q}), \quad \mathbf{x}(0) = \mathbf{x}_0 \\ \mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{w}(k), & \mathbf{w}(k) \sim N(0, \mathbf{R}) \end{cases} \tag{1}$$

where $\mathbf{x}(k) \in \mathbb{R}^{n_x}$ is the hidden state variable vector, and $\mathbf{y}(k) \in \mathbb{R}^{n_y}$ is the data corresponding to the sequence of images, which are obtained by concatenating the image to column vectors. Usually n_x is much smaller than n_y. $\mathbf{A} \in \mathbb{R}^{n_x \times n_x}$, and $\mathbf{C} \in \mathbb{R}^{n_y \times n_x}$ are the parameter matrices. $\mathbf{u}(k) \in \mathbb{R}^{n_x}$ and $\mathbf{w}(k) \in \mathbb{R}^{n_y}$ are the zero-mean normally distributed random variables, which are used to compensate the modeling error. $\mathbf{Q} \in \mathbb{R}^{n_x \times n_x}$, and $\mathbf{R} \in \mathbb{R}^{n_y \times n_y}$ are the covariance matrices. This model is usually called the linear state-space model in the control community and LDS in the machine learning community.

Then the problem can be formulated as: Given the observed image sequence $\mathbf{y}(1)$, $\mathbf{y}(2), ..., \mathbf{y}(\tau)$ $(\tau > n_x)$, estimate the values of $\mathbf{A}$, $\mathbf{C}$, $\mathbf{Q}$, and $\mathbf{x}(k)$.

Denote $\mathbf{Y}_{1:\tau} = [\mathbf{y}(1), \mathbf{y}(2), ..., \mathbf{y}(\tau)] \in \mathbb{R}^{n_y \times \tau}$. It can be reformulated as

$$\mathbf{Y}_{1:\tau} = \mathbf{C}[\mathbf{x}(1), \mathbf{x}(2), ..., \mathbf{x}(\tau)] + [\mathbf{w}(1), \mathbf{w}(2), ..., \mathbf{w}(\tau)]. \tag{2}$$

Since $\mathbf{w}(1), ..., \mathbf{w}(\tau)$ are used to compensate for the modeling error, this is expected to be negligible. Therefore the following approximation is derived:

$$\mathbf{Y}_{1:\tau} \approx \mathbf{C}[\mathbf{x}(1), \mathbf{x}(2), ..., \mathbf{x}(\tau)]. \tag{3}$$

Then the problem changes to a determination of the values of $\mathbf{C}$ and $[\mathbf{x}(1), \mathbf{x}(2), ..., \mathbf{x}(\tau)]$ from the training data $\mathbf{Y}_{1:\tau}$. To this end, the well-known SVD is performed on the matrix $\mathbf{Y}_{1:\tau}$:

$$\mathbf{Y}_{1:\tau} = \overline{\mathbf{U}}\overline{\Sigma}\overline{\mathbf{V}}^T, \tag{4}$$

where $\overline{\mathbf{U}} \in \mathbb{R}^{n_y \times \tau}$, $\overline{\mathbf{V}} \in \mathbb{R}^{\tau \times \tau}$, $\overline{\mathbf{V}}^T\overline{\mathbf{V}} \in \mathbb{I}^{\tau \times \tau}$, and $\overline{\Sigma} = diag\{\sigma_1, \sigma_2, ..., \sigma_\tau\}$, where $\{\sigma_i\}_{i=1,2,...,\tau}$ are the singular values sorted in descending order.

For approximation and dimension reduction, the first n_x singular values are selected to form the following matrix:

$$\Sigma = diag\{\sigma_1, \sigma_2, \cdots, \sigma_{n_x}\}. \tag{5}$$

In addition, the first n_x columns of $\overline{\mathbf{U}}$ are extracted to form a new matrix $\mathbf{U} \in \mathbb{R}^{n_y \times n_x}$ and the first n_x columns of $\overline{\mathbf{V}}$ are extracted to form a new matrix $\mathbf{V} \in \mathbb{R}^{\tau \times n_x}$. Then the following approximation can be obtained:

$$\mathbf{Y}_{1:\tau} \approx \mathbf{U}\Sigma\mathbf{V}^T. \tag{6}$$

Note that $\mathbf{U}^T\mathbf{U} = \mathbb{I}$ and $\mathbf{V}^T\mathbf{V} = \mathbb{I}$ still hold true. Obviously, when n_x is equal to τ, then the perfect reconstruction can be achieved. In practice, however, the

value of n_x is expected to be as small as possible to reduce the complexity. Therefore the value of n_x should be selected to balance the approximation error and the model complexity.

A straightforward comparison between Eq. (3) and Eq. (6) leads to the estimated solutions of $\mathbf{C}$ as $\widehat{\mathbf{C}} = \mathbf{U}$, and the k-th column of $\Sigma\mathbf{V}^T$ is used to get the estimation of $\mathbf{x}(k)$, which is denoted as $\hat{\mathbf{x}}(k)$.

The above derivation neglects the dynamics relationship between $\mathbf{x}(k+1)$ and $\mathbf{x}(k)$, which can be captured as the following expression:

$$\hat{\mathbf{x}}(k+1) = \mathbf{A}\hat{\mathbf{x}}(k) + \hat{\mathbf{v}}(k), \tag{7}$$

for $k = 1, 2, ..., \tau - 1$. Similarly, $\hat{\mathbf{v}}(k)$ is used to compensate the modal error and is expected to be negligible. This yields

$$\hat{\mathbf{x}}(k+1) \approx \mathbf{A}\hat{\mathbf{x}}(k), \tag{8}$$

for $k = 1, 2, ..., \tau - 1$. Thus, the estimation of $\mathbf{A}$ can be uniquely determined (assuming distinct singular values) in the Frobenius sense, by solving the following least squares problem:

$$\widehat{\mathbf{A}} = \underset{\mathbf{A}}{\operatorname{argmin}} \sum_{k=1}^{\tau-1} \| \hat{\mathbf{x}}(k+1) - \mathbf{A}\hat{\mathbf{x}}(k) \|_2, \tag{9}$$

Remark *3.1*: It is obvious that Eq. (9) only provides a rough estimation of $\mathbf{A}$. Many other advanced identification technologies can be used to obtain a more precise estimation,.

Finally, the sample input noise covariance $\mathbf{Q}$ can be estimated from

$$\widehat{\mathbf{Q}} = \frac{1}{\tau - 1} \sum_{k=1}^{\tau-1} \hat{\mathbf{v}}(k) \hat{\mathbf{v}}^T(k), \tag{10}$$

where $\hat{\mathbf{v}}(k) = \hat{\mathbf{x}}(k+1) - \widehat{\mathbf{A}}\hat{\mathbf{x}}(k)$. Then the system's output can be estimated as

$$\hat{\mathbf{y}}(k) = \widehat{\mathbf{C}}\hat{\mathbf{x}}(k), \tag{11}$$

and the corresponding image can be reconstructed from the vector $\hat{\mathbf{y}}(k)$. Please note that the time index k can be taken as any desired positive integral number but not necessarily limited to the interval $[1, \tau]$. Once the approximation model is obtained, the dynamic texture can be continually synthesized by setting the values of k. From the abovementioned equations, it can be shown that the core of dynamic texture is the representation of an image $\mathbf{y}(k)$ as a linear combination of some template images that are stored in the matrix $\widehat{\mathbf{C}}$, while the linear combination coefficients $\hat{\mathbf{x}}(k)$ follow an intrinsic evolutionary relation which is characterized by the LDS.

By modeling dynamic textures, we can descript a dynamic texture patch using the tuple $\mathbf{M} = (\mathbf{A}, \mathbf{C})$. As a feature descriptor, the tuple need to be compared with each other. It means that the distance between different models has to be defined. One such family of distance between two models is based on principal angles between specific subspaces derived from the models, namely the observability subspaces[15]. The observability subspaces is the range of the extended observability denoted by

$$O_\infty(\mathbf{M}) = \left[\mathbf{C}^T, (\mathbf{CA})^T, (\mathbf{CA}^2)^T, ...\right]^T \in \mathbb{R}^{\infty \times n_x}.$$

Let θ_i be the i-th principal angles between the spaces, The Martin distance between $\mathbf{M}_1$ and $\mathbf{M}_2$ is defined as

$$d_M(\mathbf{M}_1, \mathbf{M}_2) = -\ln \prod_{i=1}^{n} \cos^2 \theta_i. \tag{12}$$

It can be calculated by first solving for $\mathbf{P}$ from the Lyapunov equation

$$\mathbf{A}^T\mathbf{P}\mathbf{A} - \mathbf{P} = -\mathbf{C}^T\mathbf{C}, \tag{13}$$

where $\mathbf{P} = \begin{bmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} \\ \mathbf{P}_{21} & \mathbf{P}_{22} \end{bmatrix} \in \mathbb{R}^{2n_x \times 2n_x}$,

$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 & 0 \\ 0 & \mathbf{A}_2 \end{bmatrix} \in \mathbb{R}^{2n_x \times 2n_x}$, $\mathbf{C} = \begin{bmatrix} \mathbf{C}_1 & \mathbf{C}_2 \end{bmatrix} \in \mathbb{R}^{n_y \times 2n_x}$.

Then the cosine of the subspace angles θ_i is calculated as $\cos^2 \theta_i = i\text{-th eigenvalue}(\mathbf{P}_{11}^{-1}\mathbf{P}_{12}\mathbf{P}_{22}^{-1}\mathbf{P}_{21})$.

IV. Classification using Fuzzy Codebook

A. Fuzzy Codebook Generation

The codebook is needed for coding the DT patches. With the codebook, the fuzzy coding feature can be generated in the next subsection.

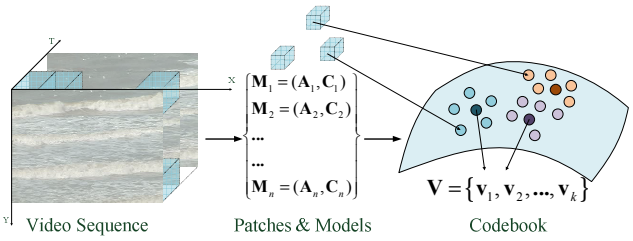

Fig.2: The process of the Fuzzy codebook generation.

The process of the Fuzzy codebook generation is shown in Fig.2. In the first stage, the videos used for training are divided into small patches. As the whole video is complicated and hard to be analyzed, we use the non-overlapped patches which are densely sampled from the videos to analyze and compute the DT models of all the patches[16]. After obtaining the spatial-temporal patches, the fuzzy codebook is generated from all the patch models. The method we use to generate the codebook is based on fuzzy k-Medoids clustering algorithm[17][18]. The details are illustrated as follows:

Let $\mathbf{P} = \{\mathbf{p}_1, ..., \mathbf{p}_M\}$ be the set of all the patches models,

and $\mathbf{D} \in R^{M \times M}$ be the dissimilarity matrix, of which element $\mathbf{D}(i,j)$ is the Martin distance between $\mathbf{p}_i$ and $\mathbf{p}_j$. The fuzzy k-Medoids clustering algorithm tries to find fuzzy prototypes $\mathbf{V} = \{\mathbf{v}_1, ..., \mathbf{v}_K\}$ and the corresponding membership degree $\mathbf{U} = \{\mu_{ik}\}$ which minimizes a criterion:

$$J(\mathbf{U}, \mathbf{V}) = \sum_{k=1}^{K} \sum_{i=1}^{M} (\mu_{ik})^m d_M(\mathbf{p}_i, \mathbf{v}_k), s.t. \sum_{k=1}^{K} \mu_{ik} = 1 \quad (14)$$

where $m > 1$ is a parameter to control the fuzziness. In the experiment we set $m = 2$.

The algorithm sets initial data of clusters and alternates two steps until convergence. One step is to find the best prototypes by fixing $\mathbf{U}$. The other step is to compute the new $\mathbf{U}$ matrix when the $\mathbf{V}$ is fixed. The details are described as follows:

(1) Initialization.

Set the number of clusters *K* which satisfies $2 \le K \ll N$; and the maximum iteration number T.

Set iteration number $t = 1$;

Randomly select *K* prototypes $\mathbf{v}_k^{(0)} \in \mathbf{P}$ ($k = 1, ..., K$);

Compute $\mu_{ik}^{(0)}$ for every $\mathbf{p}_i$ and $\mathbf{v}_k^{(0)}$.

$$\mu_{ik}^{(0)} = \left[\sum_{h=1}^{K} \left(\frac{d_M(\mathbf{p}_i, \mathbf{v}_k^{(0)})}{d_M(\mathbf{p}_i, \mathbf{v}_h^{(0)})} \right)^{1/(m-1)} \right]^{-1} \quad (15)$$

(2) Fix $\mathbf{U}^{(t-1)} = \left\{ \mu_{ik}^{(t-1)} \right\}$ and find the best prototypes $\mathbf{v}_k^{(t)}$ by finding $\mathbf{p}_l$ satisfy

$$l = \arg\min_{1 \le h \le n} \sum_{i=1}^{n} (\mu_{ik})^m d_M(\mathbf{p}_i, \mathbf{p}_h) \quad (16)$$

and set $\mathbf{v}_k^{(t)} = \mathbf{p}_l$.

(3) Fix $\mathbf{V}^{(t)} = \{\mathbf{v}_1^{(t)}, ..., \mathbf{v}_k^{(t)}\}$ and update the new $\mathbf{U}$ matrix according to

$$\mu_{ik}^{(t)} = \left[\sum_{h=1}^{K} \left(\frac{d_M(\mathbf{p}_i, \mathbf{v}_k^{(t)})}{d_M(\mathbf{p}_i, \mathbf{v}_h^{(t)})} \right)^{1/(m-1)} \right]^{-1} \quad (17)$$

(4) Compute criterion $J^{(t)}$

$$J = \sum_{k=1}^{K} \sum_{i=1}^{M} (\mu_{ik}^{(t)})^m d_M(\mathbf{p}_i, \mathbf{v}_k^{(t)}) \quad (18)$$

If $t > T$, then terminate the iteration; Otherwise we set *t* = *t* + 1 and go to Step (2).

Finally, the fuzzy codebook $\mathbf{V}^* = \{\mathbf{v}_1^*, ..., \mathbf{v}_K^*\}$ is generated. Each element of $\mathbf{V}^*$ is a DT model which is selected from the training video.

B. Fuzzy Coding Feature Design

What remains in choosing video representation is a method to effectively code the local descriptors. The coding stage partitions the local descriptor space into informative regions whose internal structure can be disregarded. These regions are also called visual words and a collection of visual words are called a visual vocabulary. A popular method for coding is vector quantization which searches the nearest neighbor to represent the descriptor. Such a representation is called BoW. BoW provides an effective way of treating a video as a collection of local DT descriptors, from which it quantizes into discrete "visual words", and then computes a compact histogram representation for video classification. One disadvantage of BoW is that it introduces significant quantization errors since only one element of the codebook is selected to represent the descriptor. To remedy this, a nonlinear SVM is normally designed as the classifier in an attempt to compensate for the quantization errors. However, since it uses nonlinear kernels, the SVM has a high training cost, including computation and storage. This means that it is difficult to scale up the algorithm.

To reduce the quantization error, Yang et al.[7] proposed to use the Sparse Coding (SC) to select several most significant elements of the codebook to represent the descriptor. Very recently, Wang et al. pointed out that the sparse coding approach proposed in [7] neglected the relationship among codebook elements. Since locality is more essential than sparsity[8], they proposed a Locality-constrained Linear Coding (LLC) method. LLC incorporates a locality constraint instead of the sparsity constraint, and produces several favorable properties. However, both SC and LLC method used linear subspace assumption and the linear reconstruction error is used to design coding vector. Since DT lies in non-Euclidean space, both SC and LLC cannot be effectively utilized. To tackle the above problems, we propose a new local fuzzy coding approach that preserves the locality of the selected codebook atoms.

Given a spatial-temporal patch which is denoted as $\mathbf{p}$, the corresponding feature vector is constructed as $\{\mu_k\}_{k=1}^{K}$. In this representation, μ_k is fuzzy membership value of sample $\mathbf{p}$ to the cluster identified by the center $\mathbf{v}_k^*$. It can be obtained by solving the following optimization problem:

$$\min \bar{J}_F(\mu_k) = \sum_{k=1}^{K} (\mu_k)^m d_M(\mathbf{p}, \mathbf{v}_k^*) \quad s.t. \sum_{k=1}^{K} \mu_k = 1 \quad (19)$$

In this case since $\mathbf{v}_k^*$ are known, the memberships can be easily obtained by

$$\mu_k = \left[\sum_{h=1}^{K} \left(\frac{d_M(\mathbf{p}, \mathbf{v}_k^*)}{d_M(\mathbf{p}, \mathbf{v}_h^*)} \right)^{1/(m-1)} \right]^{-1}. \quad (20)$$

An important difference between Eqs.(17) and (20) lies in the fact that Eq.(17) is used twistedly with Eq.(16) to simultaneously obtain the cluster prototypes and memberships, while Eq.(20) is solely used to get the

memberships for a new sample. That is to say, the sophisticated fuzzy clustering procedure is used only in the training stage and the obtained cluster prototypes $\mathbf{V}^*$ is fixed as a fuzzy codebook.

Since fuzzy technology assigns all codebook elements to the descriptor with the corresponding membership values, the obtained feature vector $\{\mu_k\}_{k=1}^K$ will be dense but not sparse. Motivated by the intuition that locality is more important [8], we construct a local fuzzy coding technology which modifies the above membership degree as

$$\bar{\mu}_k = \begin{cases} \mu_k & \text{if } \mathbf{v}_k^* \text{ is the k - nearest neighood of } \mathbf{p} \\ 0 & \text{Otherwise} \end{cases} \tag{21}$$

where the value of k reflects the locality range. When k is selected to be large enough, then $\bar{\mu}_k$ will be equal to μ_k for any k. Please note that $\{\bar{\mu}_k\}_{k=1}^K$ should be normalized to satisfy $\sum_{k=1}^{K} \bar{\mu}_k = 1$.

The coding vector for the patch $\mathbf{p}$ is then obtained as $\mathbf{c} = [\bar{\mu}_1, \bar{\mu}_2, ..., \bar{\mu}_K]^T \in R^K$.

Such a coding strategy preserves the intrinsic locality and attenuates the quantization error. In Fig.3 we give a illustrative comparison among the three coding methods.

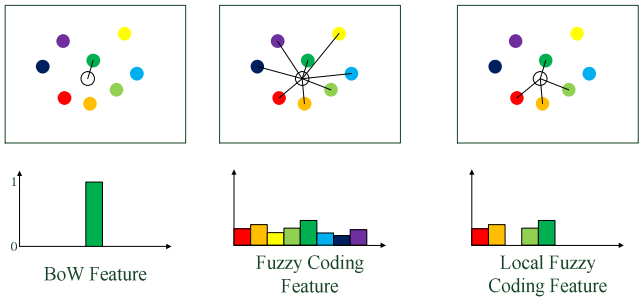

Fig.3: An illustrative comparison among the three coding methods. The empty circles are the descriptors and the colored circles are the codebook elements.

For a single video sequence, if we extract N local DT descriptors, then we can get the codes $\mathbf{C} = [\mathbf{c}_1, ..., \mathbf{c}_N] \in R^{K \times N}$ which are the results of applying the local fuzzy coding approach introduced in the above. Since the number of DT patches extracted from different video sequences may differ, we need an operator to pool all codes in a video into a single vector $\mathbf{u} \in R^K$. This operation is defined as

$$\mathbf{u} = \mathscr{P}(\mathbf{C}) \tag{22}$$

where the pooling function $\mathscr{P}$ is defined for each column of $\mathbf{C}$. Each column of $\mathbf{u}$ corresponds to the responses of all the local descriptors in the specific video. Therefore, different pooling functions construct different image statistics. In this study, we select the sum operator. Such a strategy results in

$$\mathbf{u}(i) = \sum_{j=1}^{N} |\mathbf{C}(i,j)|, \tag{23}$$

where $\mathbf{u}(i)$ is the i-th element of $\mathbf{u}$ and $\mathbf{C}(i,j)$ is the matrix element in the i-th row and j-th column of $\mathbf{C}$.

Remark: Although max-pooling is more preferred in Ref.[7-8] which used the reconstruction error to design the coding vector, it is not suitable in our case. The main reason is that we directly use the distance to design the coding vector, and therefore if max-pooling is used, the nearest atom will dominate other atoms. In our experiments, the max-pooling strategy always leads lower accuracy (about 30%) and we will not report the corresponding results.

C. Classification

The final stage is to design a classifier for recognition. Here the linear multi-class SVM [7] is used to predict the label of dynamic texture due to its advantages in speed and good performance for the fuzzy coding features. In the case of multiple classes, the training data are labeled $\{(\mathbf{u}_i, l_i)\}_{i=1}^{N_T}$, where $\mathbf{u}_i$ is the feature vector, l_i is the label and N_T is the number of the training samples. According to [7], the SVM aims to learn L linear functions $\{\omega_c^T \mathbf{u} \mid c \in \psi\}$, where ω_c is the weight vector. We use a one-versus-all strategy to train L binary linear SVMs, each solving the following unconstrained convex optimization problem

$$\min_{\omega_c} \left\{ J(\omega_c) = \|\omega_c\|_2^2 + C \sum_{i=1}^{N_T} \ell(\omega_c; l_i^c, \mathbf{u}_i) \right\} \tag{24}$$

where C is a parameter which is determined by the cross-validation. A larger C corresponds to the assignment of higher penalties to errors. The SVM constructs a hyperplane to classify the data and the weight vector ω_c serves as the normal vector to the hyperplane.

For multi-class classification, we set $l_i^c = 1$ if $l_i = c$, otherwise, $l_i^c = -1$,while $\ell(\omega_c; l_i^c, \mathbf{u}_i)$ is a hinge loss function which is defined as[7]:

$$\ell(\omega_c; l_i^c, \mathrm{f}_i) = \left[\max(0, \omega_c^T \mathrm{f} \cdot l_i^c - 1) \right]^2 \tag{25}$$

For a test sample feature $\mathbf{u}$, its class label is predicted by

$$l = \max_{c \in \psi} \omega_c^T \mathbf{u}, \tag{26}$$

Linear SVMs have become popular for solving classification tasks owing to their fast and simple online application to large scale data sets. However, many problems are not linearly separable. For these problems, kernel-based SVMs are often used, but unlike their linear variant they suffer from various drawbacks in terms of computational and memory efficiency. Thus, a more discriminant representation is required. Fortunately, the nonlinear SVM is not needed in this work.

V. Experimental Results

In this section, we present experimental results that validate the proposed algorithm in terms of DT recognition.

As indicated by [19], some existing DT data sets have a number of drawbacks such as the resolution is quite low (especially temporally); there is only a single occurrence per class, and not enough classes are available for practical classification purposes. To tackle this problem, Ref. [19] developed the DynTex dataset, which aims to serve as a reference database for dynamic texture research by providing a large and diverse database of high-quality dynamic textures that can be used for a wide variety of research and testing purposes. The DynTex database, which includes more than 650 sequences, is a diverse collection of high-quality dynamic texture videos. In [20], a subset of DynTex, which is named as DynTex++, was developed. DynTex++ eliminates sequences that contained more than one DT, or contained dynamic background. In addition, the samples in DynTex++ are cropped from the sample in DynTex.

Fig.4: Some representative dynamic texture samples from the DynTex dataset.

In this paper, we used the original data from DynTex for evaluation. Specially speaking, we use the subset of Beta and Gamma dataset which was provided by the DynTex website for classification evaluation. This dataset is composed of 290 dynamic textures divided into 12 classes. In Fig.4, we show some representative dynamic texture examples. For comparison, we list some samples of DynTex++ in Fig.5. It is obvious that they include simple texture only, while the original dataset is more complicated.

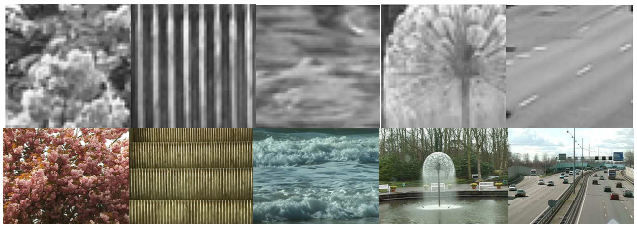

Fig.5: Samples of DynTex++ (the first row) and the videos (the second row) where they are cropped from.

We picked about half of the videos in every class as training data, and used the other videos for test. The numbers of videos for training or test in each class are listed in Table I. All video frames are transformed to be 320x240 gray scale images.

TABLE I
NUMBER OF EACH VIDEO GROUP

Class name	The number of the training videos	The number of the testing video
Calm water	15	15
Escalator	4	3
Flags	16	15
Flowers	15	14
Foliage	18	17
Fountains	19	18
Grass	12	11
Naked trees	13	12
Rotation	5	5
Sea	19	19
Smoke	8	8
Traffic	5	4
All	149	141

In the experiment, we generate the codebook using the fuzzy k-Medoids algorithm and the size of the fuzzy codebook is empirically set to $K = 64$. To investigate how the parameter k in local fuzzy coding influences the classification performance, we list the classification accuracy versus the value of k in Fig.6. From this figure we find that when k is larger than 10, the performance always superior to the case when k = 64, which corresponds to the conventional fuzzy coding. Especially, when k is about 50, the classification obtains the best results. In fact, when k=54, the accuracy reaches 81.56%.

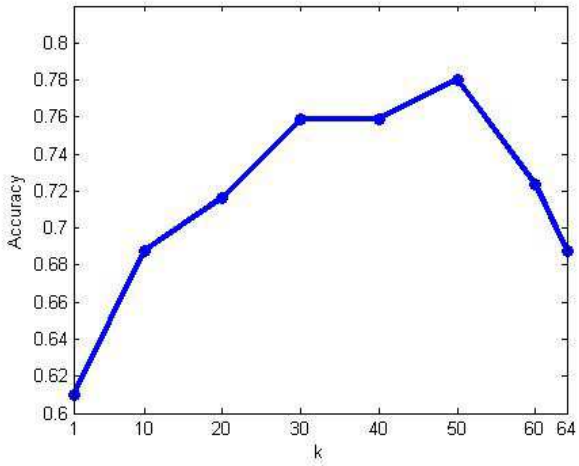

Fig.6: Classification performance depending on the k.

To show the advantages of the local fuzzy coding compared with BoS model, we design a baseline method which regards the whole video sequence as a single DT and a LDS model is used to fit it. With such a model the Martin distance is used to design the nearest neighbor (NN) classifier. In Table II we list the comparison results for those methods. In this table, BoS method corresponds to the method in [6]. We find that it is worse than the proposed method. In addition, the nearest neighbor (NN) classifier obtains the best results when the number of nearest neighbor is set to 1, i.e., the 1-NN classifier. Nevertheless, all of those NN classifiers perform

rather poor on this dataset.

TABLE II
ACCURACY OF DIFFERENT METHODS

Method	Accuracy
Our method(when k = 54)	81.56%
BoS	68.79%
1-NN	60.28%
3-NN	56.03%
5-NN	46.81%

Finally, we list the confusion matrix corresponding to our best results in Fig.7. The proposed method performs very well on the class *Flowers*, *Fountains* and *Sea*. However, for the classes of *Rotation* and *Smoke*, the performance is still weak. This remains our future work.

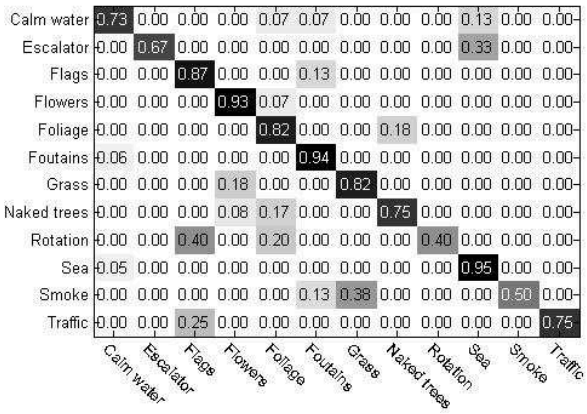

Fig.7: Confusion matrix of our method.

VI. CONCLUSIONS

In this paper, we propose a new approach to tackle the dynamic texture recognition problem. First, we utilize the fuzzy clustering technology to design a fuzzy codebook, and then construct a soft assigned fuzzy coding feature to represent the whole dynamic texture sequence. This new coding strategy preserves spatial and temporal characteristics of dynamic texture. Finally, by evaluating the proposed approach using the DynTex dataset, we show the effectiveness of the proposed local fuzzy coding strategy.

REFERENCES

[1] G. Doretto, A. Chiuso, Y. N. Wu and S. Soatto, "Dynamic textures," *International Journal of Computer Vision*, vol. 52, no. 2, pp. 91-109, 2003.

[2] A. Ravichandran and R. Vidal, "Video registration using dynamic textures," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, pp. 158-171, 2011.

[3] L. Barrington, A. B. Chan and G. Lanckriet, "Modeling music as a dynamic texture," *IEEE Trans. on Audio, Speech, and Language Processing,* vol. 18, no. 3, pp. 602-612, 2010.

[4] H. Liu, W. Xiao, H. Zhao, F. Sun, "Learning and understanding system stability using illustrative dynamic texture examples," *IEEE Trans. on Education*, vol.57, no.1, 2014, pp.4-11

[5] L. Yuan, F. Wen, C. Liu and H. Y. Shum, "Synthesizing dynamic texture with closed-loop linear dynamic system," in: *Proc. of European Conf. on Computer Vision (ECCV)*, 2004, pp. 1-14.

[6] A. Ravichandran, R. Chaudhry and R. Vidal, "Categorizing Dynamic Textures Using a Bag of Dynamical Systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 35, no. 2, pp. 342-353, 2013.

[7] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in: *Proc. of Computer Vision and Pattern Recognition(CVPR)*, 2009, pp. 1794-1801

[8] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang and Y. Gong, "Locality-constrained linear coding for image classification," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* , 2010, pp. 3360-3367.

[9] T. W. Chua, K. Leman and Y. Wang, "Fuzzy rule-based system for dynamic texture and color based background subtraction," in *Proc. of IEEE International Conference on Fuzzy Systems (FUZZ-IEEE),* 2012, pp. 1-7.

[10] J. K. Parker, L. O. Hall and J. C. Bezdek, "Comparison of scalable fuzzy clustering methods," in *Proc. of IEEE International Conference on Fuzzy Systems (FUZZ-IEEE),* 2012, pp. 1-9.

[11] F. de A T de Carvalho, and J. T. Pimentel, "A fuzzy clustering algorithm based on adaptive city-block distances," in *Proc. of IEEE International Conference on Fuzzy Systems (FUZZ-IEEE),* 2012, pp. 1-8.

[12] S. Miyamoto, S. Suzuki and S. Takumi, "Clustering in tweets using a fuzzy neighborhood model," in *Proc. of IEEE International Conference on Fuzzy Systems (FUZZ-IEEE),* 2012, pp. 1-6.

[13] N. Baili, and H. Frigui, "Fuzzy clustering with multiple kernels in feature space" in *Proc. of IEEE International Conference on Fuzzy Systems (FUZZ-IEEE),* 2012, pp. 1-8.

[14] V. Garg, S. Vempati and C. Jawahar, "Bag of visual words: A soft clustering based exposition," in *Proc. of Third National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG),* 2011, pp. 37-40.

[15] P. Saisan, G. Doretto, Y. N. Wu and S. Soatto, "Dynamic texture recognition," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001.* vol. 2, IEEE, 2001, pp. II-58.

[16] R. Avinash, C. Rizwan and V. Rene, "Dynamic Texture Toolbox," 2011. [Online]. Available: http://www.vision.jhu.edu.

[17] R. Krishnapuram, A. Joshi and L. Yi, "A fuzzy relative of the k-medoids algorithm with application to web document and snippet clustering," in *Fuzzy Systems Conference Proceedings,* vol. 3, 1999, pp. 1281-1286.

[18] F. de A T de Carvalho, Y. Lechevallier and . F. M. De Melo, "Relational partitioning fuzzy clustering algorithms based on multiple dissimilarity matrices," *Fuzzy Sets and Systems,* vol. 215, pp. 1--28, 2013.

[19] R. Peteri, S. Fazekas and M. J. Huiskes, "DynTex: A comprehensive database of dynamic textures," *Pattern Recognition Letters,* vol. 31, no. 12, pp. 1627--1632, 2010.

[20] B. Ghanem and N. Ahuja, "Maximum margin distance learning for dynamic texture recognition," in *Proc. of European Conf. on Computer Vision (ECCV)*, 2010, pp. 223--236.