This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2018.2890362, IEEE Transactions on Multimedia

Dynamic Texture Classification Using Unsupervised 3D Filter Learning and Local Binary Encoding

Xiaochao Zhao, Yaping Lin*, *Member, IEEE*, Li Liu, *Member, IEEE*, Janne Heikkilä, *Senior Member, IEEE*, and Wenming Zheng, *Senior Member, IEEE*

***Abstract*—Local binary descriptors, such as local binary pattern (LBP) and its various variants, have been studied extensively in texture and dynamic texture analysis due to their outstanding characteristics, such as grayscale invariance, low computational complexity and good discriminability. Most existing local binary feature extraction methods extract spatio-temporal features from three orthogonal planes of a spatio-temporal volume by viewing a dynamic texture in 3D space. For a given pixel in a video, only a proportion of its surrounding pixels are incorporated in the local binary feature extraction process. We argue that the ignored pixels contain discriminative information that should be explored. To fully utilize the information conveyed by all the pixels in a local neighborhood, we propose to extract local binary features from the spatio-temporal domain with 3D filters that are learned in an unsupervised manner so that the discriminative features along both the spatial and temporal dimensions are captured simultaneously. The proposed approach consists of three components: 3D filtering, binary hashing, and joint histogramming. Densely sampled 3D blocks of a dynamic texture are first normalized to have zero mean and are then filtered by 3D filters that are learned in advance. To preserve more of the structure information, the filter response vectors are decomposed into two complementary components, namely, the signs and the magnitudes, which are further encoded separately into binary codes. The local mean pixels of the 3D blocks are also converted into binary codes. Finally, three types of binary codes are combined via joint or hybrid histograms for the final feature representation. Extensive experiments are conducted on three commonly used dynamic texture databases: UCLA, DynTex and YUVL. The proposed method provides comparable results to, and even outperforms, many state-of-the-art methods.**

***Index Terms*—Dynamic texture, motion, feature extraction, local binary pattern**

I. Introduction

DYNAMIC textures (DTs) are textures with motion [1] and they are usually viewed as videos of moving scenes that exhibit certain stationary properties in the time domain [2]. Typical forms of DTs include videos of flames, sea waves, running river, water, fountains and humans in crowds. The modeling and classification of DTs have received substantial attention over the past decade. DT classification has many applications, including video retrieval [3], activity recognition [4], traffic monitoring [5], fire detection [6], [7], facial analysis [8], crowd management [9], lip reading [10], micro-expression analysis [11], and tracking [12].

DT classification is more challenging than the static case because DTs vary not only in spatial appearance but also in their organization and dynamics over time. The extraction of powerful DT features is of great importance to the success of DT classification; consequently, most research on DT classification focuses on feature extraction [8], [13]–[18]. Compared to ordinary static textures, DTs extend the notion of self-similarity to the spatio-temporal domain. Hence, a thread of research focuses on extending the existing static texture descriptors to the spatio-temporal domain to capture temporal variations. One influential work was performed by Zhao and Pietikäinen [8], who extended LBP [19], which is widely used in static texture analysis [20], [21], to DT analysis and proposed volume LBP (VLBP) to combine motion and appearance. Due to the large number of VLBP patterns, the researchers further proposed to extract LBP features from three orthogonal planes (LBP-TOP) to make the feature extraction process computationally simple. The key idea of extracting features from three orthogonal planes in LBP-TOP has been widely followed by later researchers [16], [22]–[27] due to its simplicity and good performance. Typical examples are local phase quantization on three orthogonal planes (LPQ-TOP) [22], multi-scale binarized statistical image features on three orthogonal planes (MBSIF-TOP) [16], aggregated salient features (ASF-TOP) [24] and LBP-TOP with Michelson contrast [28] (named novel LBP) [25]. In summary, local binary features, such as LBP and its 3D extension LBP-TOP, stand out due to their outstanding characteristics, including good representation power, invariance to monotonic gray-level changes (*e.g.*, those caused by illumination variations), and computational simplicity. However, the popular LBP-TOP and its variants (*e.g.*, LPQ-TOP, MBSIF-TOP, novel LBP and ASF-TOP) have the following two shortcomings: (1) They are handcrafted and require strong prior knowledge to design, and they do not adapt well to new data [29]; (2) They extract local binary features from three orthogonal planes only, ignoring

X. Zhao is with the College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, P. R. China, and the Center for Machine Vision and Signal Analysis, P.O. Box 4500 FI-90014, University of Oulu, Finland. (e-mail: s12103017@hnu.edu.cn)

Y. Lin is with the College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, P. R. China. (e-mail: yplin@hnu.edu.cn)

L. Liu is with the College of System Engineering, National University of Defense Technology, Changsha 410073, P. R. China, and the Center for Machine Vision and Signal Analysis, P.O. Box 4500 FI-90014, University of Oulu, Finland. (e-mail: li.liu@oulu.fi)

J. Heikkilä is with the Center for Machine Vision and Signal Analysis, P.O. Box 4500 FI-90014, University of Oulu, Finland. (e-mail: Janne.Heikkila@oulu.fi)

W. Zheng is with the Key Laboratory of Child Development and Learning Science of Ministry of Education, School of Biological Sciences and Medical Engineering, Southeast University, Nanjing 210096, P. R. China. (e-mail: wenming_zheng@seu.edu.cn)

The asterisk indicates the corresponding author.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2018.2890362, IEEE Transactions on Multimedia

some pixels in the local neighborhood.

Features learned from data have recently shown superiority to handcrafted features [16], [17], [27], [29]–[34]. The popularity of handcrafted features appears to be overtaken by learning-based methods, especially by deep convolutional neural networks (CNNs). However, due to the disadvantages of CNNs, such as high computational complexity, the requirement for substantial training data and the lack of general invariance, traditional descriptors such as LBP and LBP-TOP are still irreplaceable. A recent thorough experimental study [20] demonstrated that advanced LBP and LBP-TOP variants still perform on par or better than recent deeply learned features in many practical scenarios, especially for problems with limited training data, such as the micro-expression recognition problem. To the best of our knowledge, no large-scale DT database exists for now, which limits the full utilization of deep CNN techniques. Fortunately, with the advent of modern deep learning techniques, research on local descriptors has seen a renaissance, with the development of learnable local feature descriptors [16], [17], [26], [27], [29]–[32], [35], [36] being an area of particular interest. Notable examples include binarized statistical image features (BSIF) [31], simultaneous local binary feature learning and encoding (SLBFLE) [36], MBSIF-TOP [16], and orthogonal tensor dictionary learning (OTDL) [17]. On the other hand, although recent learning-based local binary feature extraction methods, such as MBSIF-TOP [16] and MPCAF-TOP [27], have achieved encouraging DT classification performance and outperformed LBP-TOP, they also extract features from three orthogonal planes of DT videos. The multiscale 2D filters used in MBSIF-TOP [16] and MPCAF-TOP [27] are learned separately from images on three orthogonal planes. In these two methods, only a proportion of the surrounding pixels of a given pixel are utilized for feature extraction, which may result in information loss.

Therefore, in this paper, we question the dominant role that three-orthogonal-planes-based methods (*e.g.*, LBP-TOP, MBSIF-TOP, and novel LBP) play in the field of DT classification. Instead of extracting binary features for only three orthogonal planes, we develop an alternative feature representation based on the point distribution of all pixels in a local 3D neighborhood. To this end, we propose to use learned 3D filters to extract local DT features, which are further encoded via binary encoding. Specifically, we propose a binarized 3D feature (B3DF) extraction method that comprises three components: (1) 3D filtering, (2) binary hashing, and (3) joint histogramming. Densely sampled 3D blocks of a DT are first normalized to have zero mean and are then filtered by 3D filters that are learned in advance. To preserve more of the structure information, the filter response vectors are decomposed into two complementary components, namely, the signs and the magnitudes, which are further separately encoded into binary codes. The local mean pixels of the 3D blocks are also encoded into binary codes. Finally, three types of binary codes are combined via joint or hybrid histograms for the final feature representation. As for filter learning, four widely used unsupervised filter learning techniques (principal component analysis (PCA), independent component analysis (ICA), sparse filtering (SF) [37] and k-means clustering [38]) are considered.

To the best of our knowledge, whether using 3D filtering with local binary encoding is beneficial for DT classification has not been explored. Although the techniques used in the proposed method are simple, our work fills this gap and thus can be used as a baseline for similar future studies. Compared with the existing TOP-based local binary feature extraction methods, the proposed approach has three benefits: (1) As 3D filters operate in the spatial and temporal dimensions at the same time, the motion and spatial appearance are captured simultaneously; (2) Only one set of filters, instead of three sets, must be learned; (3) Considerable improvement is obtained due to the use of a simple filter learning technique.

The remainder of this paper is organized as follows. Related work is introduced in Section II. Details of the proposed B3DF are given in Section III. Experimental results and comparisons are presented in Section IV. Finally, the conclusions are presented in Section V.

II. Related Work

Since DT classification has been studied for many years, plenty of methods have been reported in the literature. According to whether the underlying dynamic system is modeled, those methods can be roughly categorized into generative methods and discriminative methods. In addition, we separately categorize deep-learning-based DT classification methods into a single class. The proposed approach belongs to the discriminative category; therefore, in the following part of this section,we focus on the recent studies in this category that are most relevant to our work.

A. Generative Methods

The methods in this category attempt to estimate the underlying system that generates DTs from a number of training DTs. Then, the estimated system parameters are used for DT classification and synthesis. Two representative models are the spatio-temporal autoregressive (STAR) model [1], [2], [39], [40] and the linear dynamical system (LDS) model [2], [13], [41]–[44]. Despite the success these methods have achieved, they are impractical for the following reasons [45]: (1) The model-estimation process is time consuming; (2) The assumption of well-segmented DTs does not always hold; and (3) The similarity measure for two models is not easy to define. Recently, Baktashmotlagh *et al.* [46] assumed that a video is generated by a linear combination of stationary sources and nonstationary sources. Then, they proposed discriminative nonlinear stationary subspace analysis to separate the stationary part of a DT from the nonstationary part and used the former for DT classification. Dubois *et al.* [47] adopted 2D+T curvelet transform to decompose a DT into local oscillating phenomena and nonlocal wavefronts for DT classification.

B. Discriminative Methods

Because there are many approaches in this category, for clarity, we further group the methods into four subcategories according to the techniques used: optical-flow-based methods, fractal-analysis-based methods, LBP-based methods, and 3D-filtering-based methods. As the proposed method utilizes 3D

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2018.2890362, IEEE Transactions on Multimedia

filtering and binary encoding, we emphasize the introduction of the last two subcategories.

Optical flow was used in early studies to model motion patterns in DTs [48], [49]. For example, Lu *et al.* [48] proposed to characterize DTs through spatio-temporal multiresolution histograms based on velocity and acceleration fields. One drawback of methods based on optical flow estimation is that the assumed properties of local smoothness and brightness consistency are generally difficult to justify [50]. On the basis of the observation that DTs have self-similarities across multiple scales (this property is referred to as fractal structure [51]), Xu *et al.* [14] proposed the dynamic fractal spectrum (DFS) for DT description. Later, they extended DFS and proposed two new methods: 3D oriented transform feature (3D-OTF) [52] and wavelet domain multiple fractal spectra (WMFS) [15]. One advantage of fractal-analysis-based methods is their robustness to environmental changes; however, when the scenes in DTs are complex, the local stochastic self-similarities are weakened, and the performance is poor.

LBP was applied to extract features from each frame of a DT video [53], [54]. Ghanem and Ahuja [53] proposed to learn feature-specific weights for LBP histograms and two other types of features through maximum margin distance learning (MMDL). The final similarity between two DTs was defined as the weighted sum of their three feature-specific distances. Ren *et al.* [54] argued that the performance of LBP for DT classification was limited by the reliability issues of LBP histograms and thus proposed to apply PCA on patchwise LBP histograms to learn more reliable features for DT classification.

Zhao and Pietikäinen [8] extended the original LBP to a spatio-temporal descriptor (*i.e.*, VLBP), in which several pixels in a local volume were sampled and thresholded by their center pixel and then encoded into a binary string. VLBP has a dimensionality issue, but it is effective for DT classification. Later, Ren *et al.* [55] proposed the maximal joint mutual information criterion to select discriminative VLBP patterns rather than directly using the predefined structures. Tiwari and Tyagi [56] proposed a completed version of VLBP (CVLBP) by incorporating VLBP with both the magnitude information and the center pixel information. Zhao *et al.* [18] extended the local binary count (LBC) pattern to the spatio-temporal domain for DT classification, resulting in a descriptor named volume LBC (VLBC). Given the same number of sample pixels, the feature dimensionality of VLBC is much smaller than that of VLBP. They also proposed a completed version of VLBC (CVLBC) to enhance performance.

Another important extension of LBP to the spatio-temporal domain, LBP-TOP, was proposed by Zhao and Pietikäinen [8] in 2007. Since then, many TOP-based methods [16], [22]–[25], [27] have been reported in the literature. Rahtu *et al.* [22] applied local phase quantization on three orthogonal planes for DT description. Chen *et al.* [23] applied the Weber local descriptor [57] in a similar way for DT segmentation. Tiwari and Tyagi [25] viewed a DT as three image sequences and combined LBP with Michelson contrast [28] and center pixel information in each image sequence. Then, the concatenated feature was used for DT classification. Harandi *et al.* [58] applied sparse coding on Grassmann manifolds to learn a dictionary (the kernelized version was named KGDL) from LBP-TOP features. Hong *et al.* [24] argued that LBP-TOP disregarded the distinct characteristics of each frame, so they applied k-means clustering to LBP-TOP features to aggregate the salient features (ASF-TOP) for DT classification. 2D filter learning has also been utilized for feature extraction from three orthogonal planes. Arashloo and Kittler [16] first learned three sets of multiscale 2D filters via ICA and then used those filters to extract multiscale features for DT representation (MBSIF-TOP). Zhao *et al.* [27] used PCA to learn filters in a manner similar to that of Arashloo and Kittler, resulting in a DT representation named MPCAF-TOP.

These LBP-based methods have three drawbacks: (1) Motion information is not modeled if the LBP features are extracted from each frame individually; (2) Methods similar to VLBP or LBP-TOP may not make full use of all the pixels in a local volume; and (3) Various learning techniques are applied either to optimize the LBP code structure or to learn 2D filters instead of learning features by viewing a local volume as a whole.

3D filtering has also been utilized for DT classification. For example, Rivera and Chae [59] proposed to use 3D Gaussian-like compass masks to build a directional transitional number graph (DNG) for DT representation. Similar works include [60] and [61]. Note that these methods used predefined filters. To the best of our knowledge, 3D filter learning has not been explored for DT classification. We also place the two sparse-coding-based dictionary learning methods proposed by Quan *et al.* [17], [30] in this subcategory. However, it is unclear why and how they manually chose 27 and 25 dictionary atoms from the two dictionaries, respectively. Hadji and Wildes [62] proposed the spatio-temporal oriented energy network (SOE-NET), which uses 3D Gaussian 3^{rd}-order derivative filters for convolution and is thus learning-free.

C. Deep Learning Methods

Deep learning has been used for various computer vision tasks, and good performance has been achieved. Here, we briefly introduce some works using deep learning for DT classification. Taylor *et al.* [63] proposed a convolutional gated restricted Boltzmann machine to learn spatio-temporal features. Yan *et al.* [64] proposed to use cascaded autoencoders to model videos. Wang and Hu [65] mapped low-level features to high-level features through a deep neural network. Qi *et al.* [66] adopted a well-trained 2D CNN to transfer image features for DT representation. Arashloo *et al.* [26] viewed a DT video as three image sequences and used three multiscale dual-layer CNNs (PCANet-TOP, using pre-learned 2D PCA filters) to extract features from each image sequence. Similarly, Andrearczyk and Whelan [67] trained three CNNs to process the three image sequences. The final output of these networks was a global score vector consisting of probabilities. In contrast to methods using 2D filters, Tran *et al.* [68] trained a 3D CNN (called C3D) for spatio-temporal feature learning. A 3D CNN [68], [69] generally requires a large dataset for training; existing DT databases contain a limited number of videos, which are insufficient to train such a network. Therefore,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2018.2890362, IEEE Transactions on Multimedia

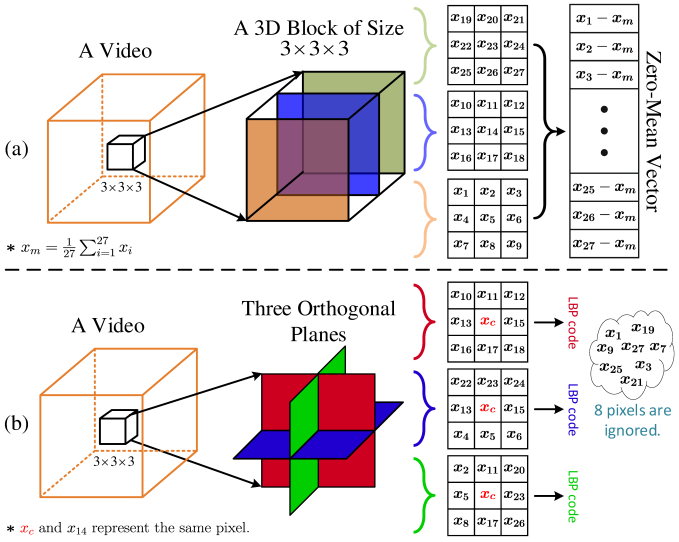


Fig. 1. Illustration of how a zero-mean vector (ZMV) is extracted from a 3D block of size $3 \times 3 \times 3$ (top). How LBP-TOP processes the same 3D block is also shown for comparison (bottom).

traditional descriptors such as LBP and LBP-TOP are still irreplaceable.

III. The Proposed Method

In this section, we present the derivation of the proposed B3DF descriptor, consisting of B3DF_S, B3DF_M and B3DF_C, given the learned 3D filters. Then, these codes are used to build hybrid histograms for DT representation. Finally, we briefly introduce how the 3D filters are learned.

A. B3DF

First, we demonstrate how the proposed method locally processes a video and compare it with that of LBP-TOP. Given a pixel in a video, its neighborhood is regarded as a whole, and all the pixels in the neighborhood (including itself) are converted into a zero-mean vector (ZMV) for further processing. Fig. 1 illustrates how B3DF and LBP-TOP process a $3 \times 3 \times 3$ block in a video, respectively. Clearly, all 27 pixels in the block are used by our method (Fig. 1(a)). However, only 19 pixels are used by LBP-TOP and many other TOP-based methods (Fig. 1(b)). We argue that the ignored 8 pixels contain discriminative information and should be exploited to build a powerful descriptor, which is the main motivation to design our B3DF.

1) B3DF_S: Assume $\mathbf{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_L]$ are L vectorized 3D filters (how they are learned will be introduced in Section III-C), where $\boldsymbol{w}_l \in \mathbb{R}^d$ $(1 \leq l \leq L, d = k^3)$ is the lth vectorized 3D filter. In this section, we describe how these L filters are used to extract the B3DF_S features from DTs.

Here, we view a given DT as a 3D volume of size $X \times Y \times T$ in 3D space, where $X \times Y$ is the spatial size and T is the temporal size. Then, we densely sample 3D blocks of size $k \times k \times k$ from the 3D volume (no padding is applied for border pixels). As shown in Fig. 1(a), where $k = 3$, the raw pixel intensities of a $k \times k \times k$ cubic neighborhood around a pixel are taken and reordered to produce a ZMV, namely, $\boldsymbol{x}_i$, in a d-dimensional feature space, *i.e.*, $\boldsymbol{x}_i \in \mathbb{R}^d$. Let $m_i = \frac{1}{d}\sum_{j=1}^{d} x_{ij}$. We have $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N]$ and $\boldsymbol{m} = [m_1, m_2, \cdots, m_N]^T$ (m_i is x_m in Fig. 1(a)), where $\boldsymbol{x}_n \in \mathbb{R}^d$ and $m_n \in \mathbb{R}$ $(1 \leq n \leq N)$ are, respectively, the nth ZMV and its corresponding mean pixel, and $N = (X - \lfloor \frac{k}{2} \rfloor) \times (Y - \lfloor \frac{k}{2} \rfloor) \times (T - \lfloor \frac{k}{2} \rfloor)$ is the total number of 3D blocks in the DT video. As the size of the zero-mean 3D block is the same as that of a 3D filter, the convolution of the zero-mean 3D block with the 3D filter is equivalent to the scalar product of their corresponding vectorized versions. Thus, $\boldsymbol{x}_n$ is filtered with the L learned 3D filters via Eqn. (1), resulting in L filter responses $\boldsymbol{r}_n = [r_{n1}, r_{n2}, \cdots, r_{nL}]^T \in \mathbb{R}^L$.

$$\boldsymbol{r}_n = \mathbf{W}^T \boldsymbol{x}_n \tag{1}$$

Now, we need to encode $\boldsymbol{r}_n$ in an appropriate way. In many existing works [8], [16], [19], [27], [56], [70], [71], the binary encoding scheme has shown to be an effective way to encode local features for both static and dynamic texture description. Additionally, many TOP-based methods [16], [22], [24], [25], [27] that we target also use binary encoding. Therefore, to make use of the effective binary encoding scheme and provide a fair comparison between our method and TOP-based methods, we adopt the binary encoding scheme to encode $\boldsymbol{r}_n$ into a binary string, generating a B3DF_S code by

$$B3DF\_S_{L,k} = \sum_{l=1}^{L} 2^{l-1} s(r_{nl}), \tag{2}$$

where the notation "_*S*" means the codes are built from signs and $s(x)$ returns 1 if $x \geq 0$, otherwise 0.

After the encoding process, we have N B3DF_S codes for the DT. To transform those codes into a compact DT representation, the statistical distribution of the codes (a 2^L-dimensional histogram, denoted as H_S) is generated and can be used for DT classification.

2) B3DF_M & B3DF_C: In conventional binary encoding based methods (*e.g.*, [8], [19]), only the sign component is used for feature encoding. However, Guo *et al.* [70] showed that using only the sign component is infeasible because it is sensitive to noise and different local structures may sometimes be characterized by the same binary code, which would reduce the discriminative capacity of the descriptor. Additionally, they also proved that the magnitude component and center pixels also contain discriminative information. As a result, this work has been followed in other papers [18], [56], [71] for both static and dynamic texture classification, resulting in significant performance improvement. Moreover, Zhao *et al.* [71] replaced the local center pixel with its local average gray level to make the feature more robust to noise. We have a component similar to the local average gray level, *i.e.*, the local mean pixel, which was previously used for data normalization. Motivated by these studies, we adopt the concept of completed modeling to build our completed B3DF, in which the magnitude component and the mean pixels are utilized for performance enhancement.

B3DF_M As shown by Eqn. (2), the B3DF_S feature encodes only the sign information. To make use of the magnitudes $\{|r_{nl}|\}_{l=1}^{L}$, we encode them into a binary code

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2018.2890362, IEEE Transactions on Multimedia

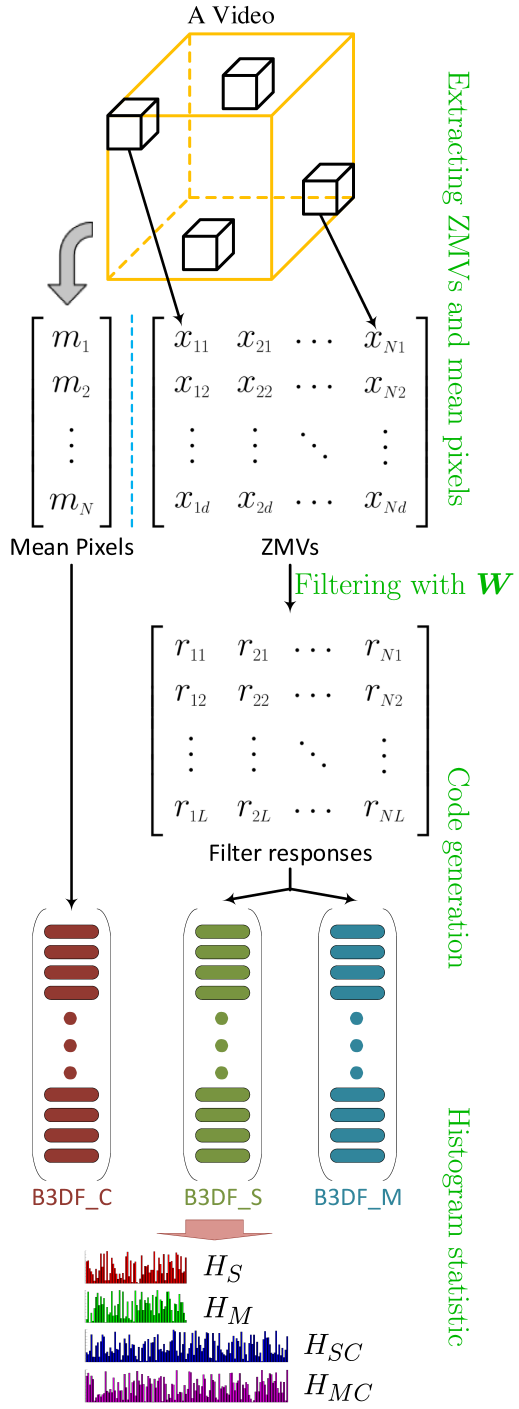


Fig. 2. Illustration of the feature extraction process of the proposed method.

named B3DF_M, where a threshold is used to binarize $|r_{nl}|$. For the binarization in a given DT, the threshold is set to the average value of all the magnitudes in the DT as

$$\mu_m = \frac{1}{NL}\sum_{n=1}^{N}\sum_{l=1}^{L}|r_{nl}|, \tag{3}$$

where $|x|$ is the absolute value of x. As μ_m is related only to the magnitudes in one DT (*i.e.*, one μ_m is computed for each DT), it is DT-specific.

After computing μ_m, the magnitudes $\{|r_{nl}|\}_{l=1}^{L}$ are binarized and then encoded by means of binary encoding through

$$B3DF\_M_{L,k} = \sum_{l=1}^{L} 2^{l-1} S(|r_{nl}| - \mu_m), \tag{4}$$

where the notation "$\_M$" means the codes are built from magnitudes and $s(x)$ is defined in Eqn. (2). Similar to the B3DF_S codes, we build a 2^L-dimensional histogram of these B3DF_M codes and denote it as H_M.

B3DF_C In [18], [56], the local center pixels are thresholded against the global mean pixel of a DT to obtain a binary code, which is beneficial for DT classification. In this paper, the local center pixel is not treated specially; thus, we threshold the local mean pixels to generate binary codes. As noted in [71], using the average local gray level is superior to using the center pixel directly; thus, we use the average value of all the mean pixels in a DT as the threshold

$$\mu_c = \frac{1}{N}\sum_{n=1}^{N} m_n. \tag{5}$$

As we need to compute one μ_c for each DT, it is also DT-specific. Then, we have

$$B3DF\_C_k = s(m_n - \mu_c), \tag{6}$$

where the notation "$\_C$" means that the codes are built from local mean pixels.

B. Hybrid Histograms for DT representation

The feature extraction process is illustrated in Fig. 2. As shown in the figure, we obtain three different binary codes ($B3DF\_S$, $B3DF\_M$ and $B3DF\_C$) after the code generation stage. We have already extracted two 1D histograms (H_S and H_M) from the former two types of codes. However, the three types of codes should be combined to make use of the different discriminative information they convey. A 3D joint histogram was used to integrate the three different types of information in [18]. If we used the same method here, we would obtain a 2^{2L+1}-dimensional feature vector, which is not practical for real-world applications. Rather than building a 3D joint histogram, we build two 2D joint histograms of 2^{L+1} bins,

$$\begin{aligned} H_{SC}(i,j) &= \sum_{x,y,t} I\{B3DF\_S = i \wedge B3DF\_C = j\}, \\ H_{MC}(i,j) &= \sum_{x,y,t} I\{B3DF\_M = i \wedge B3DF\_C = j\}, \end{aligned} \tag{7}$$

where i is an integer between 0 and $2^L - 1$; j is 0 or 1; the symbol $\wedge$ means "*and*"; and $B3DF\_S$, $B3DF\_M$ and $B3DF\_C$ are the codes at location (x, y, t).

The spatial and temporal dimensions of a DT can affect the computation of the similarities among different DTs. For example, suppose there are two DTs belonging to the same class, one of size $50 \times 50 \times 50$ and the other of size $150 \times 150 \times 150$. Although they are captured from the same scene, the Chi-square statistic of their corresponding histograms could be large, which may cause misclassification.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2018.2890362, IEEE Transactions on Multimedia

Therefore, we need to normalize the aforementioned 1D (H_S and H_M) and 2D (H_{SC} and H_{MC}) histograms to generate coherent representations. Specifically, the sum of all the bins in a histogram is first calculated; then, each bin is divided by the sum as

$$\begin{aligned} H_{1D}(j) &= \frac{H_{1D}(j)}{\sum_{j'} H_{1D}(j')}, \\ H_{2D}(i,j) &= \frac{H_{2D}(i,j)}{\sum_{i',j'} H_{2D}(i',j')}. \end{aligned} \tag{8}$$

For the ease of computing the dissimilarity of two DTs, we further vectorize H_{SC} and H_{MC}. Then, we concatenate H_{SC} and H_{MC} as another type of DT descriptor (denoted as H_{SMC}) to combine all three types of features. The dimension is 2^{L+2}. Moreover, we also consider the concatenation of H_S and H_M (denoted as H_{SM}) for DT classification.

C. 3D Filter Learning

The proposed method has one prerequisite: a set of 3D filters must be learned in advance from training data. When choosing a filter learning technique, the fact that the number of videos in most DT databases is limited must be considered. As a result, we find four representative unsupervised filter learning techniques (PCA, ICA, sparse filtering and k-means clustering) that work well with small datasets. To the best of our knowledge, these four filter learning methods were developed to learn 2D filters and have not been utilized to learn 3D filters. We believe that applying these filter learning methods to learn 3D filters for DT classification and comparing their performance is meaningful for future research in the field of DT analysis. We will briefly introduce the four filter learning processes in the following.

Training Data We follow the sampling scheme in [17] to obtain the data for filter learning. Quan *et al.* [17] randomly sampled 2000 3D patches from each DT class in the training dataset. Note that data in the test dataset are not used for dictionary learning. Therefore, suppose there are C classes in the training dataset. We randomly sample $M = 2000C$ 3D blocks of size $k \times k \times k$ from the training dataset for filter learning. As a result, M ZMVs are obtained. These ZMVs are formed into a matrix $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_M] \in \mathbb{R}^{d \times M}$ for 3D filter learning.

PCA The L vectorized filters are the first L eigenvectors (sorted in descending order with respect to their corresponding eigenvalues) of the covariance matrix $\mathbf{X}\mathbf{X}^T$. For more details, please refer to [26] or [27].

ICA we first need to compute a whitening matrix $\mathbf{Z}$ consisting of the first L rows of $\mathbf{D}^{-\frac{1}{2}}\mathbf{E}^{-1}$ for dimension reduction and data whitening, where $\mathbf{D}$ and $\mathbf{E}$ are obtained through eigendecomposition ($\mathbf{X}\mathbf{X}^T = \mathbf{E}\mathbf{D}\mathbf{E}^{-1}$). Then, an orthogonal matrix $\mathbf{U}$ can be estimated from the whitened data $\mathbf{Z}\mathbf{X}$ via ICA. Finally, the L vectorized filters are obtained by $\mathbf{W} = (\mathbf{U}\mathbf{Z})^T$. More details can be found in [31] and [16].

Sparse filtering The L vectorized filters ($\mathbf{W}$) are first randomly initialized. Suppose $\mathbf{F}_j^{(i)} = \boldsymbol{w}_j{}^T \boldsymbol{x}_i$ is the jth feature value of sample $\boldsymbol{x}_i$, where $1 \le j \le L$ and $1 \le i \le M$. Then, a two-step normalization is applied: 1) $\tilde{\mathbf{F}}_j = \mathbf{F}_j / ||\mathbf{F}_j||_2$ and 2) $\hat{\mathbf{F}}^{(i)} = \tilde{\mathbf{F}}^{(i)} / ||\tilde{\mathbf{F}}^{(i)}||_2$. Finally, $\mathbf{W}$ is obtained by optimizing the objective $\min_{\mathbf{W}} \sum_{i=1}^{N} ||\hat{\mathbf{F}}^{(\mathbf{i})}||_1$ in an iterative manner. The work in [37] explains the theory.

Fig. 3. DT examples from the UCLA (top), DynTex (middle) and YUVL (bottom) databases.

K-means clustering We follow the work in [38] with one difference: only zero-mean normalization is applied for data preprocessing. The L vectorized filters ($\mathbf{W}$) are first randomly initialized. We denote the feature matrix as $\mathbf{F}$, and $\mathbf{F}_j^{(i)}$ is the jth feature value of sample $\boldsymbol{x}_i$, where $1 \le j \le L$ and $1 \le i \le M$. Then, $\mathbf{W}$ is obtained by iteratively running the following three-step process: 1) $\mathbf{F}_j^{(i)} = \begin{cases} \boldsymbol{w}_j^T \boldsymbol{x}_i & \text{if } j = \arg\max_l |\boldsymbol{w}_l^T \boldsymbol{x}_i| \\ 0 & \text{otherwise} \end{cases}$, 2) $\mathbf{W} = \mathbf{X}\mathbf{F}^T + \mathbf{W}$, and 3) $\boldsymbol{w}_j = \boldsymbol{w}_j / ||\boldsymbol{w}_j||_2, \ \forall j$.

IV. Experiments

In this section, we present a detailed experimental evaluation of the proposed approach for DT classification. We use the nearest neighbor (NN) classifier or the nearest class center (NCC) classifier with Chi-square distance to evaluate the performance of the proposed method. We believe that a more advanced classifier such as SVM could further improve the classification performance. However, we deliberately use the NN or NCC classifier to emphasize the contribution of the proposed B3DF descriptors. The classification accuracies of our method are compared with those of the state-of-the-art approaches.

A. Experimental Setup and Implementation Details

1) DT Data Three DT databases are adopted, *i.e.*, the UCLA database [2], the DynTex database [72] and the YUVL database [50]. Frames from several typical DTs (flower, tree, flame, water, *etc.*) are shown in Fig. 3. Because few DT databases exist, these three databases have been refined and recompiled by many researchers to generate additional datasets for evaluation under various protocols. Details about these datasets and evaluation protocols will be described later. When evaluating our method, all the DT videos are converted to grayscale videos.

UCLA database: The UCLA database contains 200 DT videos from 50 classes, with four videos in each class. Each video consists of 75 frames of size 160×110. In this paper, we use a preprocessed version[1] of this database, in which all the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2018.2890362, IEEE Transactions on Multimedia

videos are cropped to frames of size 48×48 such that the key dynamic features are captured. Following [18], we use five popular evaluation protocols, in which the videos in UCLA are regrouped. The details of the five protocols are presented below.

Protocol 1 50-class leave-one-out classification: Each time one of the 200 DT videos is used for testing and the remaining 199 are used for training.

Protocol 2 50-class fourfold cross validation: For each class, three of the four videos are used for training, and the remaining one is used for testing. To ensure that each of the four videos is used as a test video one time, four split schemes are predefined[1] and this experiment is repeated four times. The average accuracy of the four experiments is the final performance indicator.

Protocol 3 9-class breakdown: In protocols 1 and 2, some videos from different classes can be semantically categorized into the same class. Thus, in this protocol, all 200 videos are regrouped into nine classes, which are smoke (4), fire (8), boiling water (8), water (12), flowers (12), sea (12), waterfall (16), fountains (20), and plant (108), where the number in parentheses denotes the number of videos in each class. For DT classification, 50% of the videos in each class are randomly selected for training, and the rest are used for testing. To assess the statistical significance, we repeated the experiment over 20 random partitions of the training and test sets.

Protocol 4 8-class breakdown: As more than one-half of the 200 videos belong to the plant class in protocol 3, the 8-class breakdown is obtained by removing the plant class. The experiment is performed in a similar way as the 9-class breakdown. We use the same data split scheme and repeat the experiment 20 times.

Protocol 5 shift-invariant evaluation: To test the translation invariance, each DT video is clipped into two non-overlapping left and right halves to remove the effect of identical viewpoints. There are two settings for this protocol: using 39 classes [73] and using 50 classes [61]. We choose the latter in this paper. Moreover, as the method used to crop the videos is unknown, we simply cut each video in the middle, resulting in a left half and a right half of equal size. In the evaluation, two experiments are conducted, one using the left-half data for training and the other using the right-half data for training. In each experiment, the remaining data are used for testing. Finally, the two classification rates are averaged as the final performance indicator.

Additionally, the NN classifier is used in all five evaluation protocols.

DynTex database: The DynTex database is a large dataset that contains more than 650 high-quality videos. Usually, only a portion of the dataset is chosen for DT classification. Five compilations of this dataset have been widely used in the literature.

DynTex-35 This dataset is an old version of the DynTex database, consisting of 35 videos. Each video belongs to a unique class and contains 250 frames of size 400×300. As only one video is included in each class, the work in [8] proposed to crop each video into 10 subvideos of different sizes. Specifically, each video is cropped at the point where $x = 170$ (horizontal direction), $y = 130$ (vertical direction) and $t = 100$ (time direction). Together with another two subvideos obtained by cutting at $t = 100$, we obtain 10 subvideos for each DT video. For classification, a leave-one-group-out[2] scheme is used with the NN classifier and the NCC classifier. When the NCC classifier is used, the features of the 9 training DT videos in each class are averaged as the class center, and a test video is classified according its similarity to the class center.

DynTex++ This dataset [53] is recompiled from 345 videos of the DynTex database. Those videos are preprocessed and cropped into videos of size $50 \times 50 \times 50$. The processed videos are grouped into 36 categories, each with 100 videos. For the evaluation, we follow the work in [53]: 50% of the videos in each class are randomly chosen for training, and the rest are used for testing. The test videos are classified with the NN classifier. This experiment is repeated 10 times, and the classification rates are averaged as the final result.

Alpha This dataset contains 60 videos from the DynTex database that are categorized into 3 classes: sea, grass, and trees.

Beta This dataset consists of 162 videos, which belong to 10 classes: sea, vegetation, trees, flags, calm water, fountains, smoke, escalator, traffic, and rotation.

Gamma This dataset is composed of 275 videos from the DynTex database. The videos belong to 10 classes: flowers, sea, naked trees, foliage, escalator, calm water, flags, grass, traffic, and fountains.

Details about the compilations of the Alpha, Beta and Gamma datasets are available on the homepage[3] of this database. All the videos in the three datasets contain 250 frames of size 352×288. For the evaluation, we use the leave-one-out classification scheme with the NCC classifier and the NN classifier.

YUVL database: **The YUVL database**[4] contains 610 videos of various resolution, temporal extents and frame rates. The authors [50] provided two ways to partition the videos: (1) basic-level (denoted as YUVL1) and (2) subordinate-level (denoted as YUVL2). Another compilation (named YUVL3) was introduced in [62].

YUVL1 According to the space time orientations present in a given pattern, all 610 videos are grouped into five classes: heterogeneous and isotropic, unconstrained, underconstrained, dominant, and multi-dominant.

YUVL2 Videos belonging to the two classes unconstrained and multi-dominant in YUVL1 are discarded. Each of the remaining three classes are further divided into two subclasses, resulting in six classes: flicker, aperture problem, single oriented, non-single oriented, wavy fluid, and stochastic.

YUVL3 Videos of the two omitted classes in YUVL2 are included, resulting in an 8-class dataset.

[1] http://www.bernardghanem.com/datasets

[2] Videos of the same size belong to a group.

[3] http://dyntex.univ-lr.fr

[4] http://vision.eecs.yorku.ca/research/spacetime-texture-data/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2018.2890362, IEEE Transactions on Multimedia

According to the existing work [62], which conducted experiments on the YUVL database, the leave-one-out classification scheme with the NN classifier is adopted.

2) Parameter Setting The proposed method has four key parameters, *i.e.*, the number of 3D blocks for filter learning (M), the 3D block size (k), the number of filters (L), and the filter learning method (PCA, ICA, sparse filtering or k-means). We use the same sampling scheme as that in [17]: 2000 3D blocks from each of the C DT classes in the training dataset are randomly sampled, resulting in $M = 2000C$ 3D blocks in total. For k and L, we empirically choose $k \in \{3, 5, 7, 9, 11\}$ and $L \in \{6, 8, 10, 12, 14\}$ to evaluate the proposed method. To identify good choices of k, L and filter learning method, we evaluate the proposed method with various parameter settings with one protocol from each of the UCLA and DynTex databases.

3) Efficient Implementation As shown in Fig. 2, we must traverse each pixel in a given DT to obtain the ZMVs and mean pixels. This traversing process can be very time consuming and may make the proposed method impractical for some real-world applications. However, Fig. 2 is drawn only for illustrative purposes; we implement the proposed method in a very efficient way. Specifically, all the mean pixels are obtained by filtering a DT with an average filter of size $k \times k \times k$, and Eqn. (1) is equivalent to filtering the DT with the mean-removed filters. Here, we briefly prove this result. Suppose $\overline{\boldsymbol{w}}_l$ is the mean-removed version of $\boldsymbol{w}_l$ and $m\_w_l = \frac{1}{d}\sum_{i=1}^{d} w_{li}$; then, we can filter the original 3D block $(\boldsymbol{x}_n + m_n)$ with $\overline{\boldsymbol{w}}_l$ as

$$\begin{aligned}\overline{\boldsymbol{w}}_l^T(\boldsymbol{x}_n + m_n) &= \textstyle\sum_{j=1}^{d}[(w_{lj} - m\_w_l)(x_{nj} + m_n)] \\ &= \textstyle\sum_{j=1}^{d} w_{lj}x_{nj} + m_n \sum_{j=1}^{d} w_{lj} \\ &\quad - m\_m_l \textstyle\sum_{j=1}^{d} x_{nj} - m_n \sum_{j=1}^{d} m\_m_l.\end{aligned} \tag{9}$$

Because $\boldsymbol{w}_l^T\boldsymbol{x}_n = \sum_{j=1}^{d} w_{lj}x_{nj}$, $\sum_{j=1}^{d} m\_m_l = \sum_{i=1}^{d} w_{li}$ and $\sum_{j=1}^{d} x_{nj} = 0$, Eqn. (9) is the same as $\overline{\boldsymbol{w}}_l^T(\boldsymbol{x}_n + m_n) = \boldsymbol{w}_l^T\boldsymbol{x}_n$. Thus, Eqn. (1) is equivalent to filtering the DT with the mean-removed filters. Additionally, Eqns. (2)–(6) contain mainly element-wise operations and summations of matrix elements, which can be conducted directly on 3D matrices. Therefore, we implement the proposed method efficiently through 3D filtering with mean-removed 3D filters, thereby avoiding the time-consuming ZMV extraction process. The computational complexities of the inefficient implementation (*i.e.*, traversing each pixel in a DT) and the efficient implementation are compared in Section IV-D.

B. Experimental Test

In this section, we study how the choices of k (the 3D block size), L (the number of 3D filters needed) and the filter learning method affect the DT classification performance. Specifically, we choose the UCLA 8-class breakdown dataset (protocol 4) and the DynTex Gamma dataset for the parameter evaluation. The reason for choosing these two datasets is that, according to previous works, each is relatively challenging in the corresponding database.

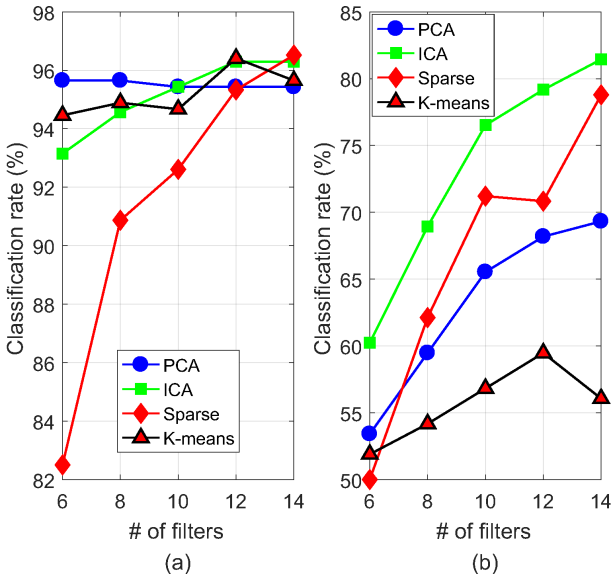


Fig. 4. Performance of B3DF_S with $k = 7$ and varying L on (a) the UCLA 8-class breakdown dataset and (b) the DynTex Gamma dataset.

First, we consider the choice of L, the number of 3D filters, while keeping the 3D block size constant at $7 \times 7 \times 7$ (*i.e.*, $k = 7$). Fig. 4 presents the classification accuracy over L for all four filter learning methods. From Fig. 4, we make the following observations. Generally, the classification performance increases with increasing L. The value of L has a greater impact on the DynTex Gamma dataset than on the UCLA 8-class breakdown dataset, except when the sparse filtering method is used. When using sparse filters, the classification rate on both datasets increases rapidly. Overall, PCA, ICA and sparse filtering achieve the best performance with $L = 14$, whereas k-means clustering performs best with $L = 12$. Note that we do not present results for $L > 14$ because a larger L produces longer feature vectors, which make the proposed method impractical for real-world applications.

Then, we fix $L = 10$ and perform experiments with varying k to study how the 3D block size affects the classification performance of B3DF_S. Fig. 5 presents the results. From

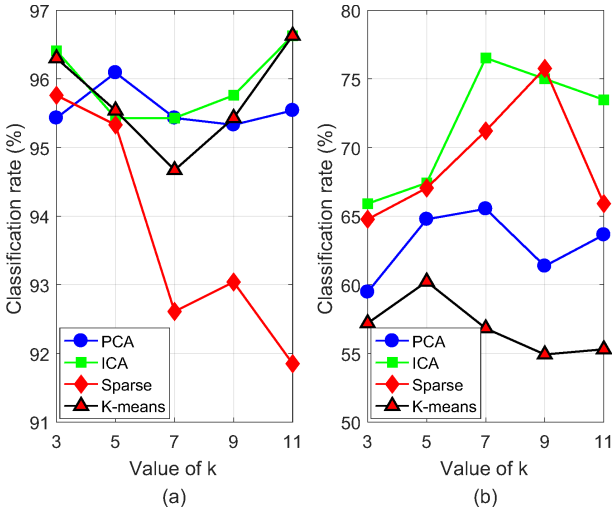


Fig. 5. Performance of B3DF_S with $L = 10$ and varying k on (a) the UCLA 8-class breakdown dataset and (b) the DynTex Gamma dataset.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2018.2890362, IEEE Transactions on Multimedia

TABLE I
PERFORMANCE COMPARISON OF THE PROPOSED FEATURES ON THE UCLA 8-CLASS BREAKDOWN AND DYNTEX GAMMA DATASETS

Filter Type	Feature Type	Classification Rate(%)	
		UCLA Prot. 4	DynTex Gamma
PCA	B3DF_S	95.43	69.32
	B3DF_SM	97.39	70.08
	B3DF_SMC	**97.50**	**71.21**
ICA	B3DF_S	96.30	**81.44**
	B3DF_SM	**98.15**	69.32
	B3DF_SMC	**98.15**	72.73
Sparse	B3DF_S	94.78	**76.89**
	B3DF_SM	97.93	65.15
	B3DF_SMC	**98.37**	70.83
K-means	B3DF_S	95.43	62.88
	B3DF_SM	96.96	61.36
	B3DF_SMC	**97.72**	**67.05**

Fig. 5, we can observe the following. Generally, k has a much stronger effect on the DynTex Gamma dataset than on the UCLA 8-class dataset. On the DynTex Gamma dataset, the classification performance first increases with increasing k and then decreases for all four filter learning methods. However, on the UCLA 8-class breakdown dataset, the effect of k is different. For the UCLA DT textures, in the case of sparse filtering, the classification performance decreases rapidly with increasing k. While for the other three filter learning methods, the classification performance fluctuates with increasing k, although the performance fluctuation is modest.

To choose an appropriate (L, k) pair, we should consider Fig. 4 and Fig. 5 as a whole. According to Fig. 4, we should learn 14 PCA filters, 14 ICA filters, 14 sparse filters, and 12 k-means filters. For value of k, a good trade-off can be achieved by setting the value of k according to Fig. 5(b) because the fluctuation range in Fig. 5(b) is larger than that in Fig. 5(a). As a result, the (L, k) pairs for PCA, ICA, sparse filtering and k-means clustering are $(14, 7)$, $(14, 7)$, $(14, 9)$ and $(12, 5)$, respectively.

After determining L and k for each filter type, we assess the performance of different filter learning methods and different feature combinations. The results are given in Table I, from which we can observe the following. First, on the UCLA 8-class breakdown dataset, B3DF_SMC shows better performance than B3DF_SM, which in turn outperforms B3DF_S, regardless of how the filters are learned. Four types of B3DF_SMC features provide similar results (the difference is less than 1%). However, the situation differs for the DynTex Gamma dataset. Except when using PCA filters, B3DF_SM gives worse results than B3DF_S and B3DF_SMC,

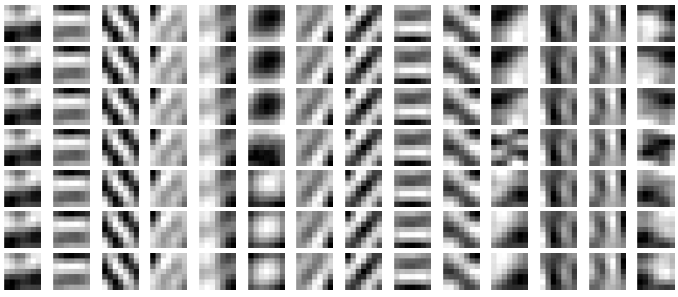


Fig. 6. ICA filters learned from the UCLA 50-class breakdown dataset. Each column shows 7 slices from a 3D filter of size $7 \times 7 \times 7$.

TABLE II
PERFORMANCE COMPARISON OF B3DF WITH OTHER METHODS ON THE UCLA DATABASE UNDER FOUR DIFFERENT EVALUATION PROTOCOLS (THE HIGHEST CLASSIFICATION RATES UNDER EACH PROTOCOL ARE HIGHLIGHTED IN BOLD)

Method	Classification Rate(%)				
	Prot. 1	Prot. 2	Prot. 3	Prot. 4	Prot. 5
AR-LDS [40]	89.50	-	-	-	-
KDT-MD [41]	-	89.50	-	-	-
BoS [13]	-	-	-	70.00	-
DL-PEGASOS [53]	-	99.00	95.60	-	-
HEM [43]	-	96.45	96.63	-	56.40
L2 Bhattacharyya [50]	81.00	-	-	-	42.30
DFS [14]	-	89.50	-	-	-
3D-OTF [52]	-	99.25	96.32	95.80	67.40
WMFS [15]	-	-	96.95	97.18	61.20
DNG [59]	-	-	98.10	97.00	-
OTDL [17]	-	98.50	97.50	97.00	**68.60**
EKDL [30]	-	-	98.60	-	-
High-level feature [65]	-	-	92.67	85.65	-
VLBP [8]	-	89.50	96.30	91.96	-
CVLBP [56]	-	93.00	96.90	95.65	-
CVLBC [18]	**99.50**	**99.50**	**99.20**	**99.02**	-
LBP-TOP [8]	-	94.50	96.00	94.34	-
MBSIF-TOP [16]	**99.50**	**99.50**	98.75	97.80	-
Novel LBP [25]	95.00	95.00	98.35	97.50	-
MPCAF-TOP [27]	**99.50**	**99.50**	99.15	98.26	-
PCANet-TOP [26]	-	**99.50**	-	-	-
DT-GoogleNet [67]	-	**99.50**	98.35	**99.02**	-
B3DF_S	99.00	99.00	97.95	96.30	46.00
B3DF_SM	98.50	98.50	*99.05*	*98.15*	*67.25*
B3DF_SMC	**99.50**	**99.50**	98.85	*98.15*	66.25

and B3DF_S using ICA filters is significantly better than all the other features. After a comprehensive consideration of the results in Table I, we believe that B3DF using ICA filters is the best choice. Therefore, we learn 14 ICA filters of size $7 \times 7 \times 7$ for feature extraction (*i.e.*, the parameters of the proposed method are fixed to using ICA with $L = 14$ and $k = 7$). As an example, Fig. 6 illustrates the 14 3D ICA filters learned from the UCLA 50-class breakdown dataset.

C. Comparative Evaluation

In this section, we compare the proposed method with various existing methods, especially those that extract features from three orthogonal planes. The parameter settings for our method were determined in the previous section. Unless otherwise stated, the classification accuracies of the compared methods are quoted directly from the original papers.

1) Results on the UCLA database The results of the proposed method and those of 22 published approaches (including recent state-of-the-art methods) are presented in Table II. Notably, most of the results in Table II were obtained using an NN classifier, with several exceptions: DL-PEGASOS [53] used MMDL+NN; high-level feature [65] and EKDL [30] used a kernel SVM; and DT-GoogleNet [67] used a softmax classifier. Additionally, the results of VLBP and LBP-TOP under protocols 2 and 4 are from [56] while those under protocol 3 are from [25].

From Table II, we make the following observations. First, in comparison with 22 DT classification methods in the literature, the proposed B3DF_SMC achieves state-of-the-art performance on protocols 1 and 2 and produces high classification rates very close to the best results on protocols 3, 4 and 5. The overall best results under protocols 1–4 are produced by

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2018.2890362, IEEE Transactions on Multimedia

TABLE III
PERFORMANCE COMPARISON OF B3DF WITH OTHER METHODS ON THE DYNTEX DATABASE (THE HIGHEST CLASSIFICATION RATES UNDER EACH PROTOCOL ARE HIGHLIGHTED IN BOLD)

Method	Classification Rate(%)								
	DynTex-35		DynTex++	Alpha		Beta		Gamma	
	NCC	NN	NN/SVM	NCC	NN	NCC	NN	NCC	NN
DL-PEGASOS [53]	-	-	63.70^{S}	-	-	-	-	-	-
HEM [43]	-	98.60	-	-	-	-	-	-	-
DFS [14]	97.63	-	89.90^{S}	83.60	-	65.20	-	60.80	-
3D-OTF [52]	96.70	-	89.17^{S}	-	-	-	-	-	-
WMFS [15]	96.50	-	88.80^{S}	-	-	-	-	-	-
DNG [59]	-	-	90.20^{N}	-	-	-	-	-	-
KGDL [58]	-	-	92.80^{S}	-	-	-	-	-	-
2D+T [47]	-	-	-	85.00	-	67.00	-	63.00	-
OTDL [17]	97.80	99.00	94.70^{S}	86.60	-	69.00	-	64.20	-
EKDL [30]	-	-	93.40^{S}	-	-	-	-	-	-
High-level feature [65]	-	-	69.00^{S}	-	-	-	-	-	-
PHA+LBP [54]	-	-	91.90^{N}	-	-	-	-	-	-
VLBP [8]	81.14	-	87.35^{N}	-	-	-	-	-	-
CVLBP [56]	85.14	-	-	-	-	-	-	-	-
CVLBC [18]	-	98.86	91.31^{N}	-	-	-	-	-	-
LBP-TOP [8]	97.14	-	89.50^{N}	-	86.67	-	80.86	-	81.44
LPQ-TOP [22]	-	-	95.00^{N}	-	-	-	-	-	-
MBSIF-TOP [16]	-	98.61	$\mathbf{97.17}^{N}$	-	90.00	-	90.70	-	91.30
ASF-TOP [24]	-	97.14	95.40^{N}	-	91.67	-	86.42	-	89.39
Novel LBP [25]	-	98.57	96.28^{N}	-	-	-	-	-	-
MPCAF-TOP [27]	96.73	99.59	96.52^{N}	86.67	96.67	71.60	91.36	67.42	89.02
B3DF_S	97.71	**100**	94.80^{N}	*90.00*	96.67	*74.07*	88.27	*81.44*	90.53
B3DF_SM	**98.00**	99.71	95.90^{N}	86.67	96.67	68.52	*90.12*	69.32	89.39
B3DF_SMC	96.57	99.71	95.58^{N}	86.67	95.00	72.22	*90.12*	72.73	*90.91*
C3D [68]	-	-	-	**100**	**100**	**95.68**	**99.38**	**96.21**	**96.97**
SOE-NET [62]	93.10	97.70	94.40^{S}	96.70	98.30	86.40	96.90	80.30	93.60

(Note: the superscript "S" stands for SVM, and "N" stands for NN.)

the CVLBC method, which is learning-free. However, the poor performance of CVLBC on the DynTex++ dataset, as shown in Table III, demonstrates its limited generalization capability since it does not involve a learning procedure. The OTDL method, which has a very complex learning process, performs best under the very challenging protocol 4, outperforming our method by a modest 1.35%.

Second, among the methods with a learning process, including the proposed method, MBSIF-TOP [16], PCANet-TOP [26], and MPCAF-TOP [27] generally outperform those methods without a learning process, such as LBP-TOP [8] and novel LBP [25]. This result again shows the benefit of learning.

Third, among the three learning-based methods, *i.e.*, the proposed method, MBSIF-TOP [16] and MPCAF-TOP [27], the proposed method performs second best, only slightly worse than MPCAF-TOP. However, our method outperforms MPCAF-TOP on datasets of large videos (*i.e.*, DynTex-35, DynTex-Alpha, DynTex-Beta, and DynTex-Gamma), as shown in Table III. Moreover, MPCAF-TOP learns 2D filters from densely sampled 2D patches, the number of which is much larger than that of the 3D blocks we require. Taking protocol 2 as an example, MPCAF-TOP uses approximately 7.5 million 2D patches at each scale (5 scales in total) for filter learning, whereas our method needs only 0.1 million 3D blocks. Despite the slight performance advantage (less than 0.3%) of MPCAF-TOP over our method, we believe the proposed method is more practical than MPCAF-TOP.

2) Results on the DynTex database The performance comparison of the proposed method and 23 published approaches is shown in Table III. Some methods were not originally evaluated on this database, and their results were reported by other researchers. For the DynTex-35 dataset, the results of VLBP and LBP-TOP are from [56] and [14], respectively. For the DynTex++ dataset, DL-PEGASOS used MMDL+NN. The results of LBP-TOP on DynTex-35 and DynTex++ are from [16], and those of WMFS are from [17]. The VLBP results are obtained by us using VLBP^{riu2}. For the Alpha, Beta and Gamma datasets, the results of DFS are from [17], and those of LBP-TOP are from [24]. Except for the DynTex++ dataset, the results of MPCAF-TOP are provided by us. As C3D [68] requires color videos as inputs, we evaluate it only on the Alpha, Beta and Gamma datasets. Specifically, we first resize the videos to frames of size 112×112 and then use the C3D network pretrained on the Sports-1M dataset [74] for feature extraction, resulting in a 4096-dimensional feature vector. Please refer to [68] for details. In contrast to the Chi-square distance we use, the similarity between two C3D [68] features is the Euclidean distance and that between two SOE-NET [62] features is the Bhattacharyya coefficient. However, the features are classified by the same classifiers, *i.e.*, the NCC classifier and NN classifier.

From Table III, we can make the following observations. In comparison with 14 existing methods on the DynTex-35 dataset, the proposed method achieves the state-of-the-art performance with either the NN or NCC classifier, outperforming even the learning-free network-based approach (*i.e.*, SOE-NET [62]) with notable improvement. The learning-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2018.2890362, IEEE Transactions on Multimedia

based methods, such as OTDL [17], MPCAF-TOP [27] and B3DF_SM, generally outperform the learning-free methods, including LBP-TOP [8] and novel LBP [25].

On the DynTex++ dataset, the proposed B3DF_SM outperforms all the methods (including SOE-NET [62] and OTDL [17]) that were evaluated with the SVM classifier and performs the fourth best when using the NN classifier. Specifically, the learning-based MBSIF-TOP [16] and MPCAF-TOP [27] outperform the proposed method by 1.27% and 0.62%, respectively, the novel LBP method outperforms the proposed method by 0.38%. We believe the reason for this performance difference may be that only 40 3D blocks, on average, are sampled from each DT video because there are 50 DT videos per class in the DynTex++ dataset used for filter learning. Overall, the learning-based methods provide better performance than the learning-free methods.

On the Alpha, Beta and Gamma datasets, only six existing approaches have been evaluated with the NCC classifier. With the exceptions of C3D [68] and SOE-NET [62], our proposed B3DF_S outperforms the learning-based methods (OTDL [17] and MPCAF-TOP [27]) and learning-free methods (2D+T [47] and DFS [14]), with substantial improvements, demonstrating the superiority of the proposed method. When using the NN classifier, C3D and SOE-NET again show better performance than the other methods. Our approach has similar performance to that of the best non-network-based methods. Moreover, B3DF_S performs better than the other features. We believe that including B3DF_M or B3DF_C would model more intraclass variation and thus degrade the performance because the dynamic scenes in the Alpha, Beta and Gamma datasets are very complex. Now, we compare the proposed method with the two network-based approaches (C3D requires training and SOE-NET is training-free). C3D performs the best on all three datasets, with a large improvement over our method and SOE-NET. On the Alpha and Beta datasets, SOE-NET performs much better than our method. However, our B3DF_S slightly outperforms SOE-NET by 1.14% when using the NCC classifier on the Gamma dataset. Therefore, in some situations, our simple learning-based method is comparable to the learning-free network-based SOE-NET. On the other hand, the proposed method still has some advantages in terms of the practicality of B3DF, SOE-NET and C3D, especially in resource-restricted scenarios, because B3DF requires much less training data and computational resources.

3) Results on the YUVL database As this database has not been widely adopted for performance evaluation, only C3D and SOE-NET have been evaluated on it (the results of C3D were reported in [62]). The performance comparison on the YUVL database is reported in Table IV. From Table IV, we can observe the following: 1) SOE-NET achieves the best performance on all three datasets; 2) The proposed B3DF_S outperforms C3D on the YUVL1 and YUVL3 datasets while C3D outperforms B3DF_S on the YUVL2 dataset; and 3) Including magnitude and center pixel information slightly degrades the performance of the proposed method. The reason C3D does not produce better results may be that it is trained with external data; thus, the method does not learn from the data in the YUVL database. Although SOE-NET also does not learn from these data, the method uses a complex handcrafted multiscale two-path network, which may be the reason for its high classification rates. On the other hand, the proposed method, a handcrafted method with a simple learning process, provides performance comparable to that of the two network-based methods. The comparison on this database shows that under some circumstance, a simple learning-based method can potentially outperform a well-trained network-based method.

TABLE IV
PERFORMANCE COMPARISON OF B3DF WITH OTHER METHODS ON THE YUVL DATABASE (THE HIGHEST CLASSIFICATION RATES UNDER EACH PROTOCOL ARE HIGHLIGHTED IN BOLD)

Method	Classification Rate(%)		
	YUVL1	YUVL2	YUVL3
C3D [68]	88.00	89.80	85.50
SOE-NET [62]	**95.60**	**91.70**	**91.00**
B3DF_S	*91.64*	*88.61*	*87.70*
B3DF_SM	91.48	87.82	86.89
B3DF_SMC	89.67	85.27	82.13

4) Results of training networks from scratch As shown above, network-based methods (*i.e.*, DT-GoogleNet, C3D and SOE-NET) are either trained on external data or carefully handcrafted. To study whether good DT classification results can be achieved by training networks from scratch on a DT database, we train a few networks on the DynTex++ dataset. We choose this dataset for two reasons: 1) It has 3600 DT videos, whereas the others have only a few hundred; and 2) The amounts of training and testing data are equal, which is different from dataset using the leave-one-out scheme. Regardless, the size of the dataset being used for training is an order of magnitude smaller than what is typically used for training CNNs on video recognition tasks (e.g., UCF101 [75] for action recognition) and such experiments often pretrain on even larger datasets (*e.g.*, Sports-1M [74]). Specifically, we train a 3D CNN and an optical-flow-based two-stream CNN, the architectures of which are presented in Fig. 7. As shown in Fig. 8, a convolutional autoencoder is also designed for comparison with other methods using unsupervised learning. Because the training dataset contains only 1800 DT videos, the three networks are designed to be shallow. During the

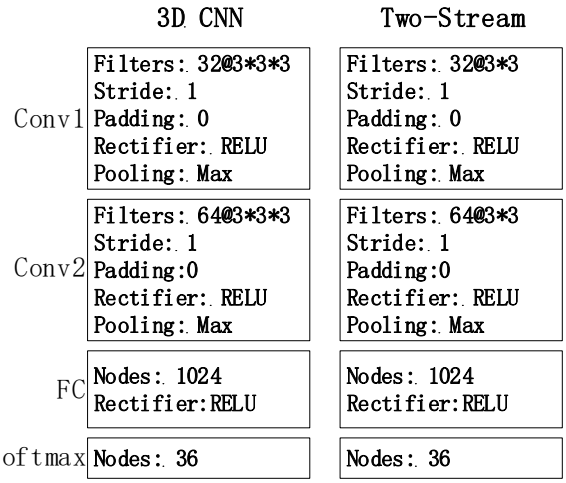


Fig. 7. Network Architectures for 3D CNN (left) and two-stream CNN (right, the image stream and the optical flow stream share the same architecture). Conv means convolution.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2018.2890362, IEEE Transactions on Multimedia

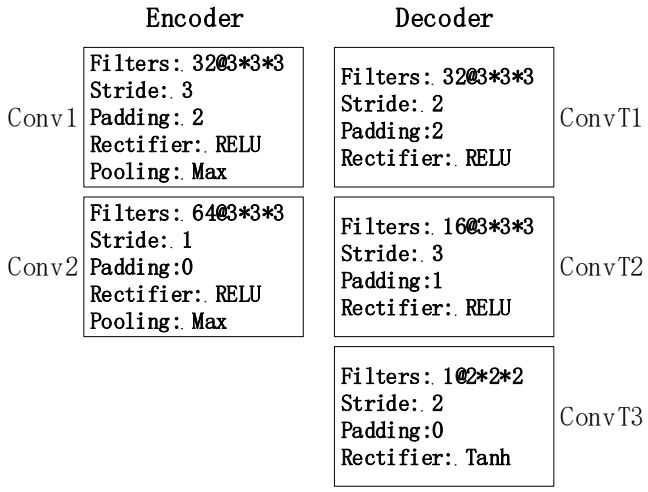


Fig. 8. Network Architectures for the convolutional auto-encoder. ConvT means transposed convolution.

training process, we adopt the stochastic gradient descent optimizer with a learning rate and momentum of 0.01 and 0.5, respectively. For the optical stream of the two-stream CNN, the vertical and horizontal optical flow maps are stacked as the input image. For the convolutional autoencoder, the outputs of Conv2 are vectorized and then classified based on their Euclidean distances.

Because 10 random splits are applied for the DynTex++ dataset, we train 10 models for each network and average their classification rates. Each model is trained for 100 epochs, and the corresponding average classification rates are presented in Fig. 9. We observe the following. For the 3D CNN, the performance first increases as the number of epochs increases and then becomes stable after the $87th$ epoch, providing a classification rate of approximately 65%. The results are similar for the two-stream CNN, except for the much higher classification rate of approximately 86%. The performance of the autoencoder first increases to its peak (65.52%) at the $51th$ epoch and then decreases. Overall, the two-stream CNN significantly outperforms the other two networks, likely because the two-stream CNN uses images as input and thus has more data for training. Regardless, the two-stream CNN is still considerably outperformed by most methods (including the proposed method), as shown in Table III. Therefore, to some extent, training a convolutional neural network from scratch is unsuitable for tasks with small datasets, such as DT classification.

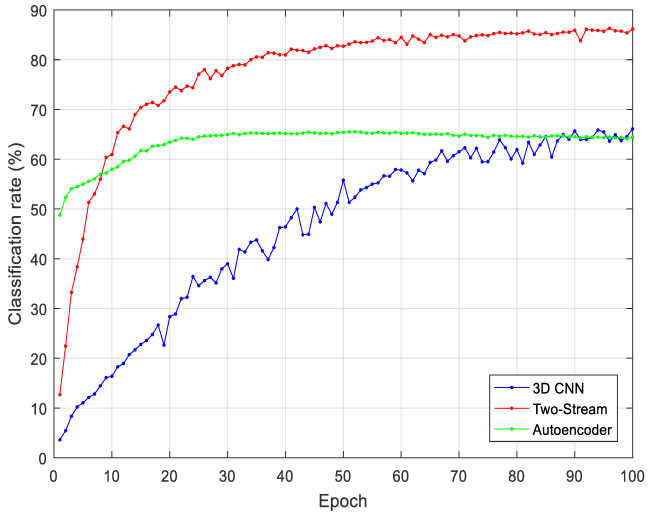


Fig. 9. Average classification rates of each network at each epoch.

TABLE V
COMPUTATION TIME (IN SECONDS) FOR 3D FILTER LEARNING UNDER VARIOUS PARAMETER SETTINGS

Learning Method	k=3	k=5	k=7	k=9	k=11
PCA	0.10	1.01	2.88	5.64	10.72
ICA	8.86	13.72	17.56	20.98	23.73
Sparse filtering	31.47	51.68	59.07	73.71	84.03
K-means clustering	21.00	55.25	124.16	293.92	496.17

D. Computational Efficiency

The computation time of the proposed method consists of two parts: the time for 3D filter learning and that for feature extraction. To measure the two components of the computation time, we run the proposed method in MATLAB on a server with four AMD Opteron 6128 CPUs and 128 GB RAM. Our program is the only workload on the system when measuring the efficiency.

To obtain the computation time for 3D filter learning, L is fixed to 14, and the other parameters are varied. The training data are 10^5 3D blocks sampled from the 50-class breakdown UCLA database. To obtain stable results, we repeat the filter learning process 50 times and use the average time as the final computation time. A comparison of the computation time of the four learning methods is presented in Table V. PCA is more efficient than the three other filter learning methods, which all contain an iterative process. ICA requires less time than sparse filtering and k-means clustering to learn the filters.

For the computation time of feature extraction, we apply the ICA filters ($L = 14$) to extract features from each of the 200 DT videos in the UCLA database. To obtain stable results, the time for processing each video is averaged as the final computation time. We compare the original implementation (involving 3D block extraction) with the efficient implementation (using 3D convolution) in Table VI. Additionally, the computation time of MBSIF-TOP and MPCAF-TOP is included for comparison. Clearly, the efficient implementation significantly accelerates the feature extraction process. On even an old server, our method requires only approximately 2.8 seconds to extract the features from a video in the UCLA database, demonstrating that it is practical for real-world

TABLE VI
COMPUTATION TIME (IN SECONDS) FOR FEATURE EXTRACTION USING ICA FILTERS

Feature Type		k=3	k=5	k=7	k=9	k=11
Original Impl.	H_S	6.76	7.59	9.54	12.35	16.44
	H_{SM}	6.84	7.61	9.62	12.53	16.85
	H_{SMC}	6.86	7.63	9.67	12.60	16.89
Efficient Impl.	H_S	0.59	1.28	2.75	6.89	11.46
	H_{SM}	0.66	1.34	2.81	6.95	11.52
	H_{SMC}	0.67	1.37	2.84	7.00	11.56
MBSIF-TOP [16]		24.63				
MPCAF-TOP [27]		10.44				

applications. On the other hand, MBSIF-TOP and MPCAF-TOP, respectively, require 24.63 seconds and 10.44 seconds, and are clearly more time consuming.

V. Conclusion

In this paper, we consider DTs in 3D space and propose to encode their 3D filter responses through binary encoding. In this way, only one set of 3D filters is needed, and motion features are simultaneously combined with appearance features. These 3D filters are efficiently learned from randomly sampled 3D blocks. After comparing four unsupervised filter learning methods, we find that ICA is most suitable for the task of DT classification. Additionally, our efficient implementation of the proposed method can substantially accelerate the feature extraction process. Compared with existing approaches, especially TOP-based ones, our method generally provides better performance on various databases, demonstrating its effectiveness for DT classification.

References

[1] M. Szummer and R. W. Picard, "Temporal texture modeling," in *Proceedings of 3rd IEEE International Conference on Image Processing*, vol. 3. IEEE, 1996, pp. 823–826.

[2] G. Doretto, A. Chiuso, Y. N. Wu, and S. Soatto, "Dynamic textures," *International Journal of Computer Vision*, vol. 51, no. 2, pp. 91–109, 2003.

[3] M. Haas, J. Rijsdam, B. Thomee, and M. S. Lew, "Relevance feedback: perceptual learning and retrieval in bio-computing, photos, and video," in *Proceedings of the 6th ACM SIGMM International Workshop on Multimedia Information Retrieval*. ACM, 2004, pp. 151–156.

[4] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior recognition via sparse spatio-temporal features," in *2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*. IEEE, 2005, pp. 65–72.

[5] A. B. Chan and N. Vasconcelos, "Classification and retrieval of traffic video using auto-regressive stochastic processes," in *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.* IEEE, 2005, pp. 771–776.

[6] B. U. Töreyin, Y. Dedeoğlu, U. Güdükbay, and A. E. Cetin, "Computer vision based method for real-time fire and flame detection," *Pattern recognition letters*, vol. 27, no. 1, pp. 49–58, 2006.

[7] J. Huang, J. Zhao, W. Gao, C. Long, L. Xiong, Z. Yuan, and S. Han, "Local binary pattern based texture analysis for visual fire recognition," in *2010 3rd International Congress on Image and Signal Processing*, vol. 4. IEEE, 2010, pp. 1887–1891.

[8] G. Zhao and M. Pietikainen, "Dynamic texture recognition using local binary patterns with an application to facial expressions," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 6, pp. 915–928, 2007.

[9] A. B. Chan, Z.-S. J. Liang, and N. Vasconcelos, "Privacy preserving crowd monitoring: Counting people without people models or tracking," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2008, pp. 1–7.

[10] G. Zhao, M. Barnard, and M. Pietikainen, "Lipreading with local spatiotemporal descriptors," *IEEE Transactions on Multimedia*, vol. 11, no. 7, pp. 1254–1265, 2009.

[11] X. Li, H. Xiaopeng, A. Moilanen, X. Huang, T. Pfister, G. Zhao, and M. Pietikainen, "Towards reading hidden emotions: A comparative study of spontaneous micro-expression spotting and recognition methods," *IEEE Transactions on Affective Computing*, 2017.

[12] K. J. Cannons, J. M. Gryn, and R. P. Wildes, "Visual tracking using a pixelwise spatiotemporal oriented energy representation," in *ECCV 2010: 11th European Conference on Computer Vision*. Springer, 2010, pp. 511–524.

[13] A. Ravichandran, R. Chaudhry, and R. Vidal, "View-invariant dynamic texture recognition using a bag of dynamical systems," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 1651–1657.

[14] Y. Xu, Y. Quan, H. Ling, and H. Ji, "Dynamic texture classification using dynamic fractal analysis," in *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 1219–1226.

[15] H. Ji, X. Yang, H. Ling, and Y. Xu, "Wavelet domain multifractal analysis for static and dynamic texture classification," *IEEE Transactions on Image Processing*, vol. 22, no. 1, pp. 286–299, 2013.

[16] S. R. Arashloo and J. Kittler, "Dynamic texture recognition using multiscale binarized statistical image features," *IEEE Transactions on Multimedia*, vol. 16, no. 8, pp. 2099–2109, 2014.

[17] Y. Quan, Y. Huang, and H. Ji, "Dynamic texture recognition via orthogonal tensor dictionary learning," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 73–81.

[18] X. Zhao, Y. Lin, and J. Heikkilä, "Dynamic texture recognition using volume local binary count patterns with an application to 2d face spoofing detection," *IEEE Transactions on Multimedia*, 2017.

[19] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[20] L. Liu, P. Fieguth, Y. Guo, X. Wang, and M. Pietikäinen, "Local binary features for texture classification: taxonomy and experimental study," *Pattern Recognition*, vol. 62, pp. 135–160, 2017.

[21] L. Liu, J. Chen, P. Fieguth, G. Zhao, R. Chellappa, and M. Pietikainen, "From BoW to CNN: Two decades of texture representation for texture classification," *International Journal of Computer Vision*, 2018.

[22] E. Rahtu, J. Heikkilä, V. Ojansivu, and T. Ahonen, "Local phase quantization for blur-insensitive image analysis," *Image and Vision Computing*, vol. 30, no. 8, pp. 501–512, 2012.

[23] J. Chen, G. Zhao, M. Salo, E. Rahtu, and M. Pietikainen, "Automatic dynamic texture segmentation using local descriptors and optical flow," *IEEE Transactions on Image Processing*, vol. 22, no. 1, pp. 326–339, 2013.

[24] S. Hong, J. Ryu, and H. S. Yang, "Not all frames are equal: aggregating salient features for dynamic texture classification," *Multidimensional Systems and Signal Processing*, pp. 1–20, 2016.

[25] D. Tiwari and V. Tyagi, "A novel scheme based on local binary pattern for dynamic texture recognition," *Computer Vision and Image Understanding*, vol. 150, pp. 58–65, 2016.

[26] S. R. Arashloo, M. C. Amirani, and A. Noroozi, "Dynamic texture representation using a deep multi-scale convolutional network," *Journal of Visual Communication and Image Representation*, vol. 43, pp. 89–97, 2017.

[27] X. Zhao, Y. Lin, and J. Heikkilä, "Dynamic texture recognition using multiscale pca-learned filters," in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 4152–4156.

[28] A. A. Michelson, *Studies in optics*. Courier Corporation, 1995.

[29] T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma, "Pcanet: A simple deep learning baseline for image classification?" *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 5017–5032, 2015.

[30] Y. Quan, C. Bao, and H. Ji, "Equiangular kernel dictionary learning with applications to dynamic texture analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 308–316.

[31] J. Kannala and E. Rahtu, "Bsif: Binarized statistical image features," in *Pattern Recognition (ICPR), 2012 21st International Conference on*. IEEE, 2012, pp. 1363–1366.

[32] Z. Lei, D. Yi, and S. Z. Li, "Discriminant image filter learning for face recognition with local binary pattern like representation," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 2512–2517.

[33] J. Lu, V. E. Liong, X. Zhou, and J. Zhou, "Learning compact binary face descriptor for face recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 10, pp. 2041–2056, 2015.

[34] Y. Duan, J. Lu, J. Feng, and J. Zhou, "Context-aware local binary feature learning for face recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 5, pp. 1139–1153, 2018.

[35] Z. Lei, M. Pietikäinen, and S. Z. Li, "Learning discriminant face descriptor," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 2, pp. 289–302, 2014.

[36] J. Lu, V. E. Liong, and J. Zhou, "Simultaneous local binary feature learning and encoding for homogeneous and heterogeneous face recognition," *IEEE transactions on pattern analysis and machine intelligence*, 2017.

[37] J. Ngiam, Z. Chen, S. A. Bhaskar, P. W. Koh, and A. Y. Ng, "Sparse filtering," in *Advances in neural information processing systems*, 2011, pp. 1125–1133.

[38] A. Coates and A. Y. Ng, "Learning feature representations with k-means," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 561–580.

[39] S. Soatto, G. Doretto, and Y. N. Wu, "Dynamic textures," in *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, vol. 2. IEEE, 2001, pp. 439–446.

[40] 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. CVPR 2001*, vol. 2. IEEE, 2001, pp. II–58.

[41] A. B. Chan and N. Vasconcelos, "Classifying video with kernel dynamic textures," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2007, pp. 1–6.

[42] 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.

[43] A. Mumtaz, E. Coviello, G. R. Lanckriet, and A. B. Chan, "Clustering dynamic textures with the hierarchical em algorithm for modeling video," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 7, pp. 1606–1621, 2013.

[44] A. Mumtaz, E. Coviello, G. R. Lanckriet, and A. B. Chan, "A scalable and accurate descriptor for dynamic textures using bag of system trees," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 4, pp. 697–712, 2015.

[45] D. Chetverikov and R. Péteri, "A brief survey of dynamic texture description and recognition," *Computer Recognition Systems*, pp. 17–26, 2005.

[46] M. Baktashmotlagh, M. Harandi, B. C. Lovell, and M. Salzmann, "Discriminative non-linear stationary subspace analysis for video classification," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 12, pp. 2353–2366, 2014.

[47] S. Dubois, R. Péteri, and M. Ménard, "Characterization and recognition of dynamic textures based on the 2d+t curvelet transform," *Signal, Image and Video Processing*, vol. 9, no. 4, pp. 819–830, 2015.

[48] Z. Lu, W. Xie, J. Pei, and J. Huang, "Dynamic texture recognition by spatio-temporal multiresolution histograms," in *Application of Computer Vision, 2005. WACV/MOTIONS'05 Volume 1. Seventh IEEE Workshops on*, vol. 2. IEEE, 2005, pp. 241–246.

[49] R. Péteri and D. Chetverikov, "Dynamic texture recognition using normal flow and texture regularity," in *Pattern Recognition and Image Analysis*. Springer, 2005, pp. 223–230.

[50] K. G. Derpanis and R. Wildes, "Spacetime texture representation and recognition based on a spatiotemporal orientation analysis," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 6, pp. 1193–1205, 2012.

[51] B. Mandelbrot, *The fractal geometry of nature*. WH Freeman, 1982.

[52] Y. Xu, S. Huang, H. Ji, and C. Fermüller, "Scale-space texture description on sift-like textons," *Computer Vision and Image Understanding*, vol. 116, no. 9, pp. 999–1013, 2012.

[53] B. Ghanem and N. Ahuja, "Maximum margin distance learning for dynamic texture recognition," in *Computer Vision – ECCV 2010: 11th European Conference on Computer Vision*. Springer, 2010, pp. 223–236.

[54] J. Ren, X. Jiang, and J. Yuan, "Dynamic texture recognition using enhanced lbp features," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 2400–2404.

[55] J. Ren, X. Jiang, J. Yuan, and G. Wang, "Optimizing lbp structure for visual recognition using binary quadratic programming," *IEEE Signal Processing Letters*, vol. 21, no. 11, pp. 1346–1350, 2014.

[56] D. Tiwari and V. Tyagi, "Dynamic texture recognition based on completed volume local binary pattern," *Multidimensional Systems and Signal Processing*, vol. 27, no. 2, pp. 563–575, 2016.

[57] J. Chen, S. Shan, C. He, G. Zhao, M. Pietikainen, X. Chen, and W. Gao, "Wld: A robust local image descriptor," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 9, pp. 1705–1720, 2010.

[58] M. Harandi, C. Sanderson, C. Shen, and B. C. Lovell, "Dictionary learning and sparse coding on grassmann manifolds: An extrinsic solution," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 3120–3127.

[59] A. R. Rivera and O. Chae, "Spatiotemporal directional number transitional graph for dynamic texture recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 10, pp. 2146–2152, 2015.

[60] W. N. Gonçalves, B. B. Machado, and O. M. Bruno, "Spatiotemporal gabor filters: a new method for dynamic texture recognition," *arXiv preprint arXiv:1201.3612*, 2012.

[61] K. G. Derpanis and R. P. Wildes, "Dynamic texture recognition based on distributions of spacetime oriented structure," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 191–198.

[62] I. Hadji and R. P. Wildes, "A spatiotemporal oriented energy network for dynamic texture recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3066–3074.

[63] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler, "Convolutional learning of spatio-temporal features," in *European conference on computer vision*. Springer, 2010, pp. 140–153.

[64] X. Yan, H. Chang, S. Shan, and X. Chen, "Modeling video dynamics with deep dynencoder," in *European Conference on Computer Vision*. Springer, 2014, pp. 215–230.

[65] Y. Wang and S. Hu, "Exploiting high level feature for dynamic textures recognition," *Neurocomputing*, vol. 154, pp. 217–224, 2015.

[66] X. Qi, C.-G. Li, G. Zhao, X. Hong, and M. Pietikäinen, "Dynamic texture and scene classification by transferring deep image features," *Neurocomputing*, vol. 171, pp. 1230–1241, 2016.

[67] V. Andrearczyk and P. F. Whelan, "Convolutional neural network on three orthogonal planes for dynamic texture classification," *Pattern Recognition*, vol. 76, pp. 36–49, 2017.

[68] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4489–4497.

[69] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 221–231, 2013.

[70] Z. Guo, L. Zhang, and D. Zhang, "A completed modeling of local binary pattern operator for texture classification," *IEEE Transactions on Image Processing*, vol. 19, no. 6, pp. 1657–1663, 2010.

[71] Y. Zhao, W. Jia, R.-X. Hu, and H. Min, "Completed robust local binary pattern for texture classification," *Neurocomputing*, vol. 106, pp. 68–76, 2013.

[72] R. Péteri, S. Fazekas, and M. J. Huiskes, "Dyntex: A comprehensive database of dynamic textures," *Pattern Recognition Letters*, vol. 31, no. 12, pp. 1627–1632, 2010.

[73] F. Woolfe and A. Fitzgibbon, "Shift-invariant dynamic texture recognition," in *European conference on computer vision*. Springer, 2006, pp. 549–562.

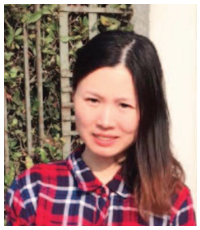
[74] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[75] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A dataset of 101 human actions classes from videos in the wild," *arXiv preprint arXiv:1212.0402*, 2012.

Xiaochao Zhao received his B.S. degree in software engineering from Hunan University, P. R. China, in 2012. Since 2014, he has been a Ph.D. candidate in the College of Computer Science and Electronic Engineering of Hunan University. His research interests include image processing and computer vision.

Yaping Lin (M'14) received his B.S. degree from Hunan University and his M.S. degree from National University of Defense Technology, in 1982 and 1985, respectively. He received his Ph.D. degree from Hunan University in 2000. He has been a professor and Ph.D. supervisor at Hunan University since 1996. During 2004-2005, he worked as a visiting researcher at University of Texas at Arlington. His research interests include cloud computing, network security, wireless sensor networks, and image processing.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2018.2890362, IEEE Transactions on Multimedia



Li Liu (M'12) received a B.S. degree in communication engineering, an M.S. degree in photogrammetry and remote sensing and a Ph.D. degree in information and communication engineering from the National University of Defense Technology, China, in 2003, 2005 and 2012, respectively. She joined the faculty at the National University of Defense Technology in 2012, where she is currently an Associate Professor with the College of System Engineering. During her Ph.D. study, she spent more than two years as a Visiting Student at the University of Waterloo, Canada, from 2008 to 2010. From 2015 to 2016, she spent ten months visiting the Multimedia Laboratory at the Chinese University of Hong Kong. From 2016 to 2018, she is visiting the Machine Vision Group at the University of Oulu, Finland. Dr. Liu was a co-chair of International Workshops at ACCV2014, CVPR2016, and ICCV2017. She served as a guest editor of the special issue Compact and Efficient Feature Representation and Learning in Computer Vision for IEEE TPAMI and of the special issue RoLoD: Robust local descriptors for computer vision, for the journal Neurocomputing. Her papers currently have over 1300 citations on Google Scholar. Her current research interests include texture analysis, image and video analysis, object detection and recognition. She has authored more than 30 papers in journals and conferences. She is a member of the IEEE.

Janne Heikkilä (SM'12) received his Doctor of Science in Technology degree in Information Engineering from the University of Oulu in 1998. Currently, he is a professor of computer vision and digital video processing on the Faculty of Information Technology and Electrical Engineering at the University of Oulu, and the head of the Degree Programme in Computer Science and Engineering. He has served as an area chair and a member of program and organizing committees of several international conferences. He is a senior editor of Journal of Electronic Imaging, an associate editor of IET Computer Vision and Electronic Letters on Computer Vision and Image Processing, a guest editor of a special issue in Multimedia Tools and Applications, a member of the governing board of the International Association for Pattern Recognition (IAPR), and a senior member of the IEEE. During 2006-2009, he was the president of the Pattern Recognition Society of Finland. He has been the principal investigator in numerous research projects funded by Academy of Finland, National Agency for Technology and Innovation (Tekes) and enterprises. His research interests include computer vision, machine learning, digital image and video processing, and biomedical image analysis. He has supervised 9 completed doctoral dissertations and published more than 160 peer reviewed scientific articles in international journals and conferences.

Wenming Zheng (SM'18) received a B.S. degree in computer science from Fuzhou University, China, in 1997, an M.S. degree in computer science from Huaqiao University, Quanzhou, China, in 2001, and a Ph.D. degree in signal processing from Southeast University, Nanjing, China, in 2004. Since 2004, he has been with the Research Center for Learning Science, Southeast University. He is currently a Professor with the School of Biological Sciences and Medical Engineering, Southeast University, and the Key Laboratory of Child Development and Learning Science of Ministry of Education, Southeast University. His research interests include neural computation, pattern recognition, machine learning, and computer vision. He is an associated editor of IEEE Transactions on Affective Computing, an associated editor of Neurocomputing and also an editorial board member of The Visual Computer.