DYNAMIC TEXTURE RECOGNITION USING 3D RANDOM FEATURES

Xiaochao Zhao[1], *Yaping Lin*[1], *Li Liu*[2,3]

[1]Hunan Provincial Key Lab of Trusted System and Network, Hunan University, China
[2]Center for Machine Vision and Signal Analysis, University of Oulu, Finland
[3]College of System Engineering, National University of Defense Technology, China
{s12103017, yplin}@hnu.edu.cn, li.liu@oulu.fi

ABSTRACT

In this paper, we present a novel, simple but effective approach for dynamic texture recognition using 3D random features. Compared with the existing dynamic texture recognition approaches using carefully designed features for high performance, our method use only a few 3D random filters to extract spatio-temporal features from local dynamic texture blocks, which are further encoded into a low-dimensional feature vector. To explore the representative power of the 3D random features, we use two different encoding schemes, the learning-based Fisher vector encoding and the learning-free binary encoding. The proposed method is tested on the UCLA and DynTex databases with various evaluation protocols. Experimental results demonstrate the high performance of our method for dynamic texture recognition.

***Index Terms*—** Dynamic texture recognition, random features, Fisher vector encoding, binary encoding

1. INTRODUCTION

Dynamic textures (DTs) are textures with motion [1]. Typical examples of DT are sea-waves, swaying trees, drifting smoke, *etc.* Studies related to dynamic textures include three main aspects, segmentation, synthesis and classification. DT analysis has various applications of visual processing, such as fire detection[2] and facial analysis[3]. In this paper, we focus on the recognition of DTs.

One prerequisite for effective DT recognition is that a compact DT representation should be constructed. According to the techniques applied, Feature extraction methods for DT recognition can be grouped into four classes: flow-based methods, model-based methods, transform-based methods, and discriminative methods. Many early works utilize optic flow to characterize DTs (*e.g.*, Peh and Cheong's work [4]).

Model-based methods try to learn a parametric model that generates the DTs. Ravichandran *et al.* [5] proposed to learn linear dynamical system (LDSs) from local DT blocks, from which a codebook is built. Transform-based methods extract geometric properties for DT descriptor. Xu *et al.* [6] utilized fractal dimension to describe DTs, resulting in a descriptor called dynamic fractal spectrum (DFS). One extension of DF-S is presented in [7].

Discriminative methods do not learn the underlying dynamic model and directly extract features from DTs. Many works [3, 8, 9, 10, 11, 1] in this class are based on local binary pattern (LBP) [12]. Besides LBP-based methods, dictionary-learning-based and graph-based methods (*e.g.*, [13, 14]) also fall in this group. Additionally, deep learning has also been utilized for DT recognition [15, 16].

Unlike the above mentioned methods that use either carefully designed features or those learned through complex steps, inspired by the impressive work in Ref. [17], we propose to make use of random features for DT recognition. Specifically, we use a few 3D random filters to extract spatio-temporal features from local DT blocks. The filter responses are encoded by Fisher vector (FV) encoding [18] and binary encoding. The binary representation shows comparable performance while the other one outperforms many state-of-the-art methods. As far as we know, this work is the first to use random feature for DT recognition.

The rest of this paper is organized as follows. Section 2 provides the background of this paper. The proposed method is detailed in Section 3. Experimental results are reported in Section 4. Section 5 concludes this paper.

2. BACKGROUND

Random projection [17] means the mapping of a set of high-dimensional data points to a randomly chosen low-dimensional subspace. In the field of compressed sensing, if a signal (audio, image or video) is known to be sparse or compressible, a small number of nonadaptive measurements in the form of random linear combinations of basis elements could provide near-optimal reconstruction of the original signal, without any further knowledge about the signal [19, 20]. This implies that a small number of random projections can well capture enough salient information in the signal. Random projection has achieved great success for static texture recognition [21, 22, 23, 17, 24, 25]. For example, Liu *et al.* [22, 17, 24, 25] used random projections as an encoder to extract local texture features. Their random features showed great superiority over many state-of-the-art methods.

In the field of information retrieval [26], random projection has been used for local sensitive hashing, which can alleviate the curse of dimensionality for approximate nearest neighbor searching in very high dimensional data.

In the field of deep learning, networks with random weights have also shown preferable properties [27, 28, 29]. Giryes *et al.* [27] found networks with random Gaussian filters can perform distance-preserving embedding of the original data while training is to find hyperplanes to separate different data points. Gilbert *et al.* [28] connected networks with random Gaussian filters for compressed sensing and analyzed the invertibility of networks, in which they observed that networks with random Gaussian filters could achieve good classification performance. Mongia *et al.* [29] applied random weights for static texture generation.

3. THE PROPOSED METHOD

As DTs are videos that contain spatially and temporally repetitive pattern, we argue that DTs are compressible, which is backed by the success of video compression algorithms. Therefore, we propose to extract spatio-temopral features with 3D random filters, followed by two encoding schemes.

3.1. Random Features

For a given DT video of size $X \times Y \times T$ ($X \times Y$ is the spatial size while T is the temporal size), we densely extract local blocks of size $s \times s \times s$, resulting in a set of vectorized spatio-temporal blocks $\{\boldsymbol{x}_i\}_{i=1}^N$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ ($d = s \times s \times s$) and $N = (X - \lfloor \frac{s}{2} \rfloor) \times (Y - \lfloor \frac{s}{2} \rfloor) \times (T - \lfloor \frac{s}{2} \rfloor)$. Then, zero-mean normalization is applied to $\{\boldsymbol{x}_i\}_{i=1}^N$, such that $\boldsymbol{x}_i$ is replaced by $\overline{\boldsymbol{x}}_i = \boldsymbol{x}_i - \overline{\boldsymbol{x}}$, where $\overline{\boldsymbol{x}} = \frac{1}{N}\sum_{i=1}^N \boldsymbol{x}_i$. Later, spatio-temporal random features can be extracted from $\{\overline{\boldsymbol{x}}_i\}_{i=1}^N$.

Similar to the work in Ref. [17], we draw L 3D filters $\{\boldsymbol{w}_j\}_{j=1}^L$ ($\boldsymbol{w}_j \in \mathbb{R}^d$ is vectorized) from the standard normal distribution. By filtering a local block $\overline{\boldsymbol{x}}_i$ with $\{\boldsymbol{w}_j\}_{j=1}^L$, we can get a L-dimensional random feature vector $\boldsymbol{f}_i = [f_{i1}, f_{i2}, \dots, f_{iL}]^T \in \mathbb{R}^L$ as

$$f_{ij} = \boldsymbol{w}_j^T \overline{\boldsymbol{x}}_i \, . \tag{1}$$

In the Sections 3.2-3.3, we will introduce how to encoding those random feature vectors with FV encoding and binary encoding, respectively.

3.2. FV encoding

FV encoding [18] is widely used to build a global descriptor out of local visual features for image or video representation. Specifically, it models the generative process of those local descriptors as a Gaussian Mixture Model (GMM). After fitting the GMM from training data, the global descriptor (*i.e.*, a FV) can be produced with the parameters of the GMM model.

For a training dataset of C DTs, we randomly sample S local blocks from each of the C DTs and extract random features from them, resulting in CS random feature vectors for training. Then we can fit a GMM of K components from those random features. The parameters of this model are denoted as $\Theta = (\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k, \pi_k : k = 1, \dots, K)$, in which $\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k$ and π_k are the respectively the mean, covariance and weight of the kth component.

After obtaining the GMM parameters Θ, we can use them to encode the densely extracted random features $\{\boldsymbol{f}_n\}_{n=1}^N$ of a given DT into a global descriptor. Firstly, we compute the soft assignment weight of each descriptor $\boldsymbol{f}_n$ to the kth component through

$$q_k^n = \frac{\pi_k P_k(\boldsymbol{f}_n)}{\sum_{t=1}^{t=K} \pi_t P_t(\boldsymbol{f}_n)} \tag{2}$$

where $P_k(\boldsymbol{f}_n)$ is the kth Gaussian probability density function. Secondly, the gradients of the kth component with respect to its mean and covariance are computed as

$$\boldsymbol{g}_k^u = \frac{1}{N\sqrt{\pi_k}} \sum_{n=1}^N q_k^n \left(\frac{\boldsymbol{f}_n - \boldsymbol{\mu}_k}{\boldsymbol{\sigma}_k} \right), \tag{3}$$

$$\boldsymbol{g}_k^v = \frac{1}{N\sqrt{2\pi_k}} \sum_{n=1}^N q_k^n \left[\left(\frac{\boldsymbol{f}_n - \boldsymbol{\mu}_k}{\boldsymbol{\sigma}_k} \right)^2 - 1 \right], \tag{4}$$

where $\boldsymbol{g}_k^u$ and $\boldsymbol{g}_k^v$ capture the first-order and second-order differences of the local descriptors to the kth Gaussian component, respectively. Thirdly, we concatenate these two types of differences as the global descriptor $\boldsymbol{F}_{fv} = [\boldsymbol{g}_1^{uT}, \boldsymbol{g}_2^{uT}, \dots, \boldsymbol{g}_K^{u\,T}, \boldsymbol{g}_1^{vT}, \boldsymbol{g}_2^{vT}, \dots, \boldsymbol{g}_K^{v\,T}]^T \in \mathbb{R}^{2KL}$. Finally, the signed square-rooting and the L2 normalization are sequentially applied to $\boldsymbol{F}_{fv}$ for performance enhancement [18].

3.3. Binary Encoding

The FV representation described in the previous section involves a learning process. It is unfair to compare it with existing learning-free descriptors, such as VLBP, LBP-TOP, *etc.* Therefore, we decide to encode the random features $\{\boldsymbol{f}_n\}_{n=1}^N$ with binary encoding, which is adopted by many LBP-based methods.

We first encode each random feature $\boldsymbol{f}_n$ into an integer code $BRF_n \in [0, 2^L - 1]$ by

$$BRF_n = \sum_{l=1}^L 2^{l-1} S(f_{nl}), \tag{5}$$

where $S(x)$ returns 1 if $x > 0$, otherwise 0. Then we build a histogram of the BRF codes to represent the DT by

$$\boldsymbol{F}_b(m) = \sum_n A(BRF_n == m), \tag{6}$$

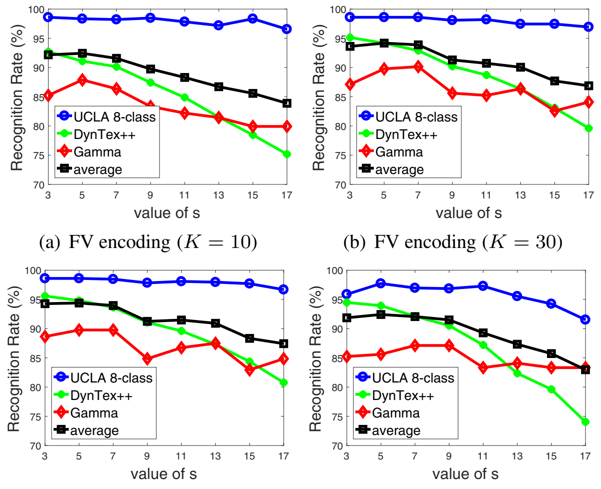
where $m \in [0, 2^L - 1]$ is an integer and $A(x)$ returns 1 if x is true, otherwise 0. Considering that different DTs may have different spatial or temporal sizes, L1 normalization is applied to $\boldsymbol{F}_b \in \mathbb{R}^{2^L}$ to build a coherent representation.

4. EXPERIMENTS

In this section, we experimentally evaluate the proposed two types of representations and compare then with the sate-of-the-art. When measuring dissimilarity, we use Euclidean distance for the FV feature, and Chi-square statistic for the binary feature. Additionally, the simple nearest neighbor classifier is adopted for DT recognition. Unless otherwise stated, the results of existing approaches are from the literature.

4.1. Dataset

We adopt two benchmark DT databases for experimental evaluation, *i.e.*, the UCLA [30] and DynTex [31] databases. Details about them are as follows.



(a) FV encoding ($K = 10$)

(b) FV encoding ($K = 30$)

(c) FV encoding ($K = 50$)

(d) Binary encoding

Fig. 1: Recognition rates as a function of parameter s with various encoding schemes.

The UCLA database contains 200 DTs of size $160 \times 110 \times 75$. Ghanem cropped these DTs to have frame size 48×48, capturing the key dynamical features[1] and we use this version. There are three commonly used evaluation protocols on this database, *i.e.*, 50-class, 9-class and 8-class breakdowns. As the results under 50-class breakdown are tending to be saturate, we only use the latter two protocols. In the 9-class breakdown, 200 DTs are grouped into 9 classes. The 8-class breakdown contains the remaining data of 9-class version after excluding the plant class. In evaluation, half of the DTs in each class are randomly chosen for training and the rest for test (the recognition rates are averaged over 20 random splits).

The DynTex database contains more than 650 DTs. There are five evaluation protocols (each with a recompiled dataset) on this database, which are denoted by DynTex-35[3], DynTex++[8], Alpha, Beta and Gamma, respectively. The DynTex-35 dataset is an old version of the database, containing 35 DTs of size $400 \times 300 \times 250$. According to Ref. [3], each of the 35 DTs are cropped into 10 sub-DTs of various sizes and a leave-one-group-out evaluation scheme is followed. The DynTex++ dataset is recompiled from 345 DTs, containing 100 DTs of size $50 \times 50 \times 50$ from each of the 36 classes. 50 DTs in each class are chosen for training and the rest for test, and the recognition rates are averaged over 10 trials. The Alpha, Beta and Gamma datasets are consist of 60 DTs from 3 classes, 162 DTs from 10 classes and 275 DTs from 10 classes, respectively. A leave-one-out evaluation scheme is applied on this three datasets.

4.2. Parameter Setting

There are four parameters, the local block size (s), the number of filters (L), the number of training blocks from each DT in the training dataset (S) and the number of GMM components (K). To make the GMM less dependent on the training data, we empirically choose $S = 500$, which is less than 0.4% of all the blocks in a DT of size $48 \times 48 \times 75$. As the dimension of $\boldsymbol{F}_b$ is a power of 2, we choose $L = 10$ to avoid producing high-dimensional feature vectors. To make our method efficient, the values of s and K should be small. Therefore, we test $s \in \{3, 5, 7, 9, 11, 13, 15, 17\}$ and $K \in \{10, 30, 50\}$ under three relatively challenging protocols (*i.e.*, UCLA 8-class, DynTex++ and Gamma) of the two databases, to choose appropriate values for s and K. The results are shown in Fig. 1.

From Fig. 1, it can be observed: 1) the Gama dataset is more challenging than others while the UCLA 8-class breakdown is less challenging; 2) a smaller s (less than 9) generally provides better performance; 3) The average recognitions (black curves) clearly indicate that $s = 5$ is a good choice and that further increasing s would degrade performance; 4) when using FV encoding, results on the UCLA 8-class breakdown is not sensitive to the value of K while a larger K could produce higher recognition rates on the DynTex++ and Gamma datasets. As a result, we choose $s = 5$ and present results with K being 10, 30 and 50 at the same time when comparing the proposed method with other methods. In brief, we use $S = 500$, $L = 10$, $s = 5$ and $K \in \{10, 30, 50\}$.

4.3. Comparison with Existing Methods

In this section, we compare our method with the state-of-the-art methods for DT recognition. As the proposed method

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

Table 1: Performance comparison of the proposed method with other methods

Method	**Recognition Rate(%)**							**Dim.**
	9-class	**8-class**	**DynTex-35**	**DynTex++**	**Alpha**	**Beta**	**Gamma**	
BoS [5]	78.00	84.00	-	-	-	-	-	96
DL-PEGASOS [8]	95.60	-	-	63.70	-	-	-	-
DFS [6]	97.50^S	99.00^S	95.92	89.90^S	-	-	-	316
WMFS [7]	96.95	97.18	-	88.80^S	-	-	-	702
DNG [14]	98.10	97.00	-	90.20	-	-	-	-
OTDL [13]	97.50	97.00	99.00	94.70^S	-	-	-	2700
High level feature [15]	92.67^S	85.65^S	-	69.00	-	-	-	-
VLBP [3]	96.30	91.96	-	87.35	-	-	-	16384
CVLBC [11]	99.20	**99.02**	98.86	91.31	-	-	-	11250
LBP-TOP [3]	96.00	94.34	91.43	89.50	86.67	80.86	81.44	768
MBSIF-TOP [9]	98.75	97.80	98.61	**97.17**	90.00	**90.70**	**91.30**	6144
ASF-TOP [32]	-	-	97.14	95.40	91.67	86.42	89.39	#class $\times$70
novel LBP [10]	98.35	97.50	98.57	96.28	-	-	-	1536
MPCAF-TOP [1]	99.15	98.26	-	96.52	-	-	-	3840
SOE-NET [16]	-	-	97.70	-	-	-	-	1600
$\boldsymbol{F}_{fv}$ (K=10)	**99.35**	98.34	99.14	91.09	**98.33**	84.57	87.88	200
$\boldsymbol{F}_{fv}$ (K=30)	99.18	98.59	**99.43**	94.16	**98.33**	87.65	89.77	600
$\boldsymbol{F}_{fv}$ (K=50)	99.24	98.59	**99.43**	94.80	**98.33**	89.51	89.77	1000
$\boldsymbol{F}_b$	98.40	97.72	98.57	93.93	93.33	84.57	85.61	1024

(Superscript "S" means the result is obtained using SVM classifier.)

belongs to the discriminative group. we mainly compare our method with other discriminative methods (VLBP/LBP-TOP [3], CVLBC [11], MBSIF-TOP [9], ASF-TOP [32], novel LBP [10], and MPCAF-TOP [1]). Performance of a few methods belonging to other groups are also compared. The results of various methods are presented in Table 1, in which we include the dimensionality of feature vectors (if available).

The FV representation outperforms the binary one on every dataset. Except for the DynTex++ and Beta datasets, the former with $K = 10$ shows better performance than the latter, which indicates that learning does make random features more discriminative. Increasing K from 10 to 30 would not make significant difference on the UCLA 8-class, UCLA 9-class and the DynTex-35 datasets, while improvements can be observed on other datasets. This is because the former three datasets are less challenging than other datasets and a GMM with more components is needed to model the variation in feature space.

On the UCLA 9-class dataset, the FV representation with $K = 10$ provides the highest recognition rate (99.35%). The binary achieves a high recognition rate of 98.40%, outperforming many other methods other than CVLBC (99.20%), MBSIF-TOP (98.75%) and MPCAF-TOP (99.15%). Similar situation can be observed on the UCLA 8-class dataset. CVLBC outperforms our FV and binary representations by 0.43% and 1.34%, respectively. Using SVM, DFS also outperforms the proposed method. Once again, MBSIF-TOP and MPCAF-TOP show marginal improvements over our binary representation.

On the DynTex-35 dataset, our 600-dimensional FV representation provides the highest recognition rate of 99.43%. OTDL utilizes very complex techniques and achieves 99.00%. Several other methods show marginal improvements over our binary representation. On the DynTex++ dataset, the proposed method only gives relatively good results, considering its low dimensionality. MBSIF-TOP, MPCAF-TOP and novel LBP show improvements higher than 1%. On the three datasets of Alpha, Beta and Gamma, only three existing methods (*i.e.*, LBP-TOP, MBSIF-TOP and ASF-TOP) have been tested with the same protocol we choose (results of LBP-TOP are quoted from Ref. [32]). On the Alpha dataset, our method is the best, achieving a recognition rate of 98.33%. On the Beta and Gamma datasets, MBSIF-TOP outperforms our methods by 1.19% and 1.53%, respectively.

All the methods that outperform our method, produce feature vectors with much higher dimensionality, while their improvements are marginal. Theirfore, the proposed method shows the advantages of high performance and low dimensionality, which makes it very practical for real-world video-based applications.

5. CONCLUSION

We propose to utilize random projection to extract spatio-temporal features for DT recognition. Fisher vector encoding and binary encoding are utilized to provide fair comparisons with existing learning-based and learning-free methods. Extensive experiments show the superiority of our method.

6. REFERENCES

[1] X. Zhao et al., "Dynamic texture recognition using multiscale pca-learned filters," in *ICIP*. IEEE, 2017, pp. 4152–4156.

[2] J. Huang et al., "Local binary pattern based texture analysis for visual fire recognition," in *CISP*. IEEE, 2010, vol. 4, pp. 1887–1891.

[3] G. Zhao and M. Pietikainen, "Dynamic texture recognition using local binary patterns with an application to facial expressions," *TPAMI*, vol. 29, no. 6, pp. 915–928, 2007.

[4] C. Peh and L. Cheong, "Synergizing spatial and temporal texture," *TIP*, vol. 11, no. 10, pp. 1179–1191, 2002.

[5] A. Ravichandran et al., "Categorizing dynamic textures using a bag of dynamical systems," *TPAMI*, vol. 35, no. 2, pp. 342–353, 2013.

[6] Y. Xu et al., "Dynamic texture classification using dynamic fractal analysis," in *ICCV*. IEEE, 2011, pp. 1219–1226.

[7] H. Ji et al., "Wavelet domain multifractal analysis for static and dynamic texture classification," *TIP*, vol. 22, no. 1, pp. 286–299, 2013.

[8] B. Ghanem and N. Ahuja, "Maximum margin distance learning for dynamic texture recognition," in *ECCV*. Springer, 2010, pp. 223–236.

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

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

[11] X. Zhao et al., "Dynamic texture recognition using volume local binary count patterns with an application to 2d face spoofing detection," *TMM*, vol. 20, no. 3, pp. 552–566, 2017.

[12] T. Ojala et al., "Multiresolution gray-scale and rotation invariant texture classification with local binary pattern-s," *TPAMI*, vol. 24, no. 7, pp. 971–987, 2002.

[13] Y. Quan et al., "Dynamic texture recognition via orthogonal tensor dictionary learning," in *ICCV*. IEEE, 2015, pp. 73–81.

[14] A. R. Rivera and O. Chae, "Spatiotemporal directional number transitional graph for dynamic texture recognition," *TPAMI*, vol. 37, no. 10, pp. 2146–2152, 2015.

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

[16] I. Hadji et al., "A spatiotemporal oriented energy network for dynamic texture recognition," in *CVPR*. IEEE, 2017, pp. 3066–3074.

[17] L. Liu and P. Fieguth, "Texture classification from random features," *TPAMI*, vol. 34, no. 3, pp. 574–586, 2012.

[18] F. Perronnin et al., "Improving the fisher kernel for large-scale image classification," in *ECCV*. Springer, 2010, pp. 143–156.

[19] David L Donoho, "Compressed sensing," *TIT*, vol. 52, no. 4, pp. 1289–1306, 2006.

[20] E. J Candes and T Tao, "Near-optimal signal recovery from random projections: Universal encoding strategies?," *TIT*, vol. 52, no. 12, pp. 5406–5425, 2006.

[21] K. Jarrett et al., "What is the best multi-stage architecture for object recognition?," in *ICCV*. IEEE, 2009, pp. 2146–2153.

[22] L. Liu et al., "Sorted random projections for robust texture classification," in *ICCV*. IEEE, 2011, pp. 391–398.

[23] A. M. Saxe et al., "On random weights and unsupervised feature learning," in *ICML*, 2011, pp. 1089–1096.

[24] L. Liu et al., "Sorted random projections for robust rotation-invariant texture classification," *PR*, vol. 45, no. 6, pp. 2405–2418, 2012.

[25] L. Liu et al., "Fusing sorted random projections for robust texture and material classification," *TCSVT*, vol. 25, no. 3, pp. 482–496, 2015.

[26] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," in *FOCS*. IEEE, 2006, pp. 459–468.

[27] R. Giryes et al., "Deep neural networks with random gaussian weights: A universal classification strategy?," *TSP*, vol. 64, no. 13, pp. 3444–3457, 2016.

[28] A. Gilbert et al., "Towards understanding the invertibility of convolutional neural networks," in *IJCAI*, pp. 1703–1710. 2017.

[29] M. Mongia et al., "On random weights for texture generation in one layer cnns," in *ICASSP*. IEEE, 2017, pp. 2207–2211.

[30] G. Doretto et al., "Dynamic textures," *IJCV*, vol. 51, no. 2, pp. 91–109, 2003.

[31] R. Péteri et al., "Dyntex: A comprehensive database of dynamic textures," *PRL*, vol. 31, no. 12, pp. 1627–1632, 2010.

[32] S. Hong et al., "Not all frames are equal: aggregating salient features for dynamic texture classification," *Multidim. Syst. Sign. Process.*, pp. 1–20, 2018.