



HAL
open science

Learning Graph Representation with Randomized Neural Network for Dynamic Texture Classification

Lucas C Ribas, Jarba Joaci de Mesquita Sá Junior, Antoine Manzanera, Odemir M Bruno

► **To cite this version:**

Lucas C Ribas, Jarba Joaci de Mesquita Sá Junior, Antoine Manzanera, Odemir M Bruno. Learning Graph Representation with Randomized Neural Network for Dynamic Texture Classification. Applied Soft Computing, 2021. ⟨hal-03431533⟩

HAL Id: hal-03431533

<https://hal.science/hal-03431533v1>

Submitted on 16 Nov 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



HAL Authorization

Learning Graph Representation with Randomized Neural Network for Dynamic Texture Classification

Lucas C. Ribas^{a,b,c}, Jarbas Joaci de Mesquita Sá Junior^d, Antoine Manzanera^c, Odemir M. Bruno^{b,a}

^a*Institute of Mathematics and Computer Science
University of São Paulo*

*Avenida Trabalhador São-Carlense, 400, Centro
13566-590 São Carlos, SP, Brazil*

^b*São Carlos Institute of Physics
University of São Paulo*

PO Box 369, 13560-970, São Carlos, SP, Brazil

^c*U2IS, ENSTA Paris, Institut Polytechnique de Paris,
828 Boulevard des Maréchaux, 91120 Palaiseau, France*

^d*Curso de Engenharia da Computação*

*Programa de Pós-Graduação em Engenharia Elétrica e de Computação
Universidade Federal do Ceará (Federal University of Ceara), Campus de Sobral
Rua Coronel Estanislau Frota, 563, Centro
Sobral, Ceará, CEP: 62010-560, Brasil*

Abstract

Dynamic textures (DTs) are pseudo periodic data on a space \times time support, that can represent many natural phenomena captured from video footages. Their modeling and recognition are useful in many applications of computer vision. This paper presents an approach for DT analysis combining a graph-based description from the Complex Network framework, and a learned representation from the Randomized Neural Network (RNN) model. First, a directed space \times time graph modeling with only one parameter (radius) is used to represent both the motion and the appearance of the DT. Then, instead of using classical graph measures as features, the DT descriptor is learned using a RNN, that is trained to predict the gray level of pixels from local topological measures of the graph. The weight vector of the output layer of the RNN forms the descriptor.

Several structures are experimented for the RNNs, resulting in networks with final characteristics of a single hidden layer of 4, 24, or 29 neurons, and input layers 4 or 10 neurons, meaning 6 different RNNs. Experimental results on DT recognition conducted on Dyntex++ and UCLA datasets show a

high discriminatory power of our descriptor, providing an accuracy of 99.92%, 98.19%, 98.94% and 95.03% on the UCLA-50, UCLA-9, UCLA-8 and Dyntex++ databases, respectively. These results outperform various literature approaches, particularly for UCLA-50. More significantly, our method is competitive in terms of computational efficiency and descriptor size. It is therefore a good option for real-time dynamic texture segmentation, as illustrated by experiments conducted on videos acquired from a moving boat.

Keywords: Dynamic Texture, Complex Networks, Learned Features, Randomized Neural Networks

1 **1. Introduction**

2 Dynamic textures (DT) are visual patterns that vary both spatially and
3 temporally. Therefore, they can represent a wide range of time-varying natu-
4 ral phenomena, such as fire, sea waves, fountains, smoke, among others. The
5 dynamics of the textures are related to the objects or processes present in the
6 videos, that exhibit a certain stationarity with respect to both space and time
7 [1]. In general, they can be characterized by simple and repetitive patterns and
8 non-rigid motion guided by non linear and stochastic dynamics such as sway-
9 ing branches or bubbling water [2, 3]. Over the last years, DTs have received
10 significant attention from the computer vision community, due to their several
11 applications, such as face spoofing detection [4], traffic monitoring [5], crowd
12 behavior analysis [6], fire detection [7], etc.

13 The basic task of DT recognition consists in classifying a piece of video
14 (i.e. a 3D space \times time volume of data), with respect to meaningful classes
15 such as those mentioned earlier. This task can then be extended to further
16 goals like semantic segmentation of dynamic surfaces, or discriminative detection
17 of objects over non stationary backgrounds. The notorious difficulty of DT
18 recognition comes from the large variability and the diffuse character of natural
19 textures that, unlike objects, cannot - in general - be precisely located in space
20 and time. To these challenges are added those specific to the dynamic nature

21 of the problem: the model should represent the different types of movement
22 that can affect the texture: flow, swell, growth, waving, etc. In addition, as a
23 basic step, DT recognition is expected to work in a fast and reactive manner. It
24 must then be able to run in real-time and to adapt itself on line to a changing
25 environment. Motivated by these challenges, a range of approaches has been
26 proposed, that can be divided into the following five categories: (i) filter-based,
27 (ii) motion-based, (iii) discrimination-based, (iv) learning-based and (v) model-
28 based methods.

29 The filter-based methods extend still texture analysis techniques to charac-
30 terize the dynamic textures at different scales in the spatio-temporal domain.
31 In [8, 9], the authors used oriented energy filters to extract the spatio-temporal
32 characteristics of the dynamic textures. Then, Arashloo and Kittler [10] devel-
33 oped a multiresolution binarized statistical image features approach that gener-
34 ates binary code by filtering operations on different regions of the space \times time
35 and in three orthogonal planes. Independent component analysis learns the fil-
36 ters for each orthogonal plane, which generally represents a significant compu-
37 tational cost. Feichtenhofer et al. [11] presented a bag of space \times time energies
38 framework that extracts primitive features from a bank of spatio-temporally
39 steerable filters. In [12] is proposed the spatio-temporal Directional Number
40 transitional Graph (DNG) descriptor. The feature extraction is based on the
41 direction of the temporal flow to compute the structure of the local neigh-
42 borhood and the transition of the principal directions between frames. **Other**
43 **approaches explore high-order features from Gaussian gradients [13] or use un-**
44 **supervised 3D filter learning and local binary encoding [14]. In the same way, in**
45 **[15], the authors introduced a novel filtering kernel, named difference of deriva-**
46 **tive Gaussians, which is based on high-order derivative of a Gaussian kernel.**
47 The methods of this category have achieved good results in several databases,
48 however, they are generally limited both in terms of computational complexity
49 and performance.

50 The motion-based techniques essentially use the kinematic information es-
51 timated from frames to describe the dynamic textures. Many techniques use

52 the optical flow due to its efficiency in the description of the motion, like [16]
53 that used the global magnitude and direction of the vector field of normal flow.
54 Other approaches are based on the combination of normal flow features and pe-
55 riodicity [17], multi-resolution histogram of the velocity and acceleration fields
56 [18] or rotation and scale invariant features of the image distortions computed
57 using optical flow [19]. More recently, Nguyen et al. [20] presented an operator
58 based on local vector patterns that encode the local motion features from beams
59 of dense trajectories with good results. In [21] the authors developed a method
60 that is based on the time-varying vector field and used singular patterns pooled
61 from the bag-of-keypoint-based coefficients dictionary. Although the methods
62 from this category are efficient for motion description, they often neglect most of
63 the appearance information, which is essential in many problems. Furthermore,
64 the optical flow techniques suppose brightness constancy and local smoothness
65 in the dynamic textures, which can be a very strong constraint [12].

66 Discrimination-based methods generally use local features such as the Local
67 Binary Patterns (LBP), widely used in image analysis. In fact, most of the
68 methods based on discrimination for dynamic textures are extensions of LBP
69 methods to the space \times time domain. In this sense, Zhao and Pietikäinen
70 [22, 23] proposed the two most popular methods, which have the advantage
71 of simplicity. The Volumetric LBP (VLBP) [23] encodes the local feature by
72 means of 3D neighborhood and uses as feature vector a very large histogram
73 (i.e., 16384 descriptors). Next, the authors proposed the LBP-TOP [22], which
74 applies the LBP operator on three orthogonal planes (two temporal planes and
75 one spatial plane) and combines the three histograms, reducing the size of the
76 feature vector. Recently, other works also have extended the LBP operator such
77 as multiresolution edge-weighted local structure pattern [24], helix local binary
78 pattern [25] and rotation-invariant version [26]. On the other hand, in [27] the
79 authors introduced the LTGH descriptor, which combines LBPs and gray-level
80 co-occurrence matrix (GLCM) on orthogonal planes (TOP). Nguyen et al. [28]
81 proposed the momental directional patterns framework that extends the Local
82 Derivative Pattern operator to improve the capture of directional features. In

83 [29], it is proposed the local tetra pattern operator on three orthogonal planes,
84 which computes feature codes based on the central pixel and directions of the
85 neighbors. In general, these methods provide promising performances, however,
86 they have some limitations such as sensitivity to noise [25] and large feature
87 vectors [23, 30].

88 Following the success of the deep convolutional neural networks (CNN) on
89 image classification, there has recently been a growing interest in learning-
90 based methods for dynamic texture analysis. Qi et al. [31] applied pre-trained
91 CNN to extract mid-level features from the frames. The first and second or-
92 der statistics from the mid-level features are used to create the feature vector.
93 Later, Arashloo et al. [32] presented a deep multi-scale convolutional network
94 (PCANet-TOP) architecture that learns filters employing the principal compo-
95 nent analysis (PCA) on each orthogonal plane. Andrearczyk and Whelan [33]
96 proposed a framework based on applying CNNs on three orthogonal planes.
97 This framework used the AlexNet and GoogleNet models that were trained
98 on spatial frames and temporal slices from the dynamic texture videos. The
99 output of the three CNNs are combined to obtain a feature vector. The ap-
100 proaches of this category are powerful and usually obtain outstanding results in
101 DT classification. However, they have known limitations that can make them
102 unfeasible for many real-world problems, such as their difficulty to be imple-
103 mented on embedded platforms, and the need for a considerable number of
104 training samples, that can be impossible to get, particularly when online adap-
105 tation/learning is needed. Zhao et al. [34] explore two different approaches to
106 learn 3D random features: learning-based Fisher vector and the learning-free
107 binary encoding. In [35] is proposed a DT descriptor, which employs Random-
108 ized Neural Networks (RNNs) to learn the local features from three orthogonal
109 planes. The determining interest is that the RNN has a single feed-forward
110 hidden layer and a fast learning algorithm, making the feature extraction ex-
111 tremely efficient. In the model-based category, the methods analyze the DTs
112 through mathematical or physical models. Popular methods of this category
113 are based on linear dynamical systems (LDS). In [1], the estimated parameters

114 of the LDS model are used for characterizing the DT. However, the method has
115 limitations such as a poor invariance to rotation, scale and illumination [33].
116 To overcome the view-invariance limitation, Ravichandran et al. [36] proposed
117 the Bag-of-dynamical Systems (BoS), which uses the LDSs features with non-
118 Euclidean parameters computed from non linear dimensionality reduction and
119 clustering. Later, in [37] the authors extended the method to extract interest
120 points with a dense sampling and used two alternative approaches for forming
121 the code books. Wang and Hu [38] also proposed a bag-of-words approach to
122 encode chaotic features. More recently, methods that use deterministic walkers
123 [39] and Complex Network (CN) theory [40, 3] have been used with success for
124 DT analysis. In particular, the CN-based methods obtained great results due
125 to their flexibility and ability to represent the motion and appearance in the
126 DTs. These methods model the DT as graphs, and extract statistical measures
127 from them. Despite the promising results, we believe that a more robust char-
128 acterization of the graph can improve the performance compared to the classic
129 statistical measures. Furthermore, for graph modeling, these methods have as
130 a drawback the need to adjust four parameters.

131 In summary, the main drawbacks present in the existing methods from the
132 literature are: (i) strong emphasis on either appearance or motion, and poor
133 combination of the two aspects; (ii) large feature vectors; (iii) complex and
134 computationally costly algorithms; (iv) many parameters to adjust; and (v)
135 very simple graph measures.

136 To address these problems, we present in this paper a DT method that
137 extends the approach proposed in [41] for static textures, that is, it combines
138 a graph based description from the Complex Network (CN) framework, and
139 a learned representation from the Randomized Neural Network (RNN) model.
140 Using a small piece of DT data, the RNN can be trained to predict the gray
141 level value of a pixel, from its local topology features, provided by measures on
142 the space \times time graph of the video. The learned weight vector of the output
143 layer of the RNN is then used as descriptor of the video. Thus, we refer to
144 our method as **Complex Patterns learned using Randomized Neural Networks**

145 (CPNN). Our contributions are:

- 146 • A more simple directed graph modeling than [3, 40, 42] for dynamic tex-
147 tures based on only one parameter (radius).
- 148 • A robust learned representation for graphs using the RNN, instead of
149 using classic statistical measures. The RNN is an extremely compact
150 neural network that has a fast and few-shots learning algorithm, which is
151 a determining advantage with respect to deep neural network approaches.
- 152 • A DT descriptor that provides competitive accuracy and processing time
153 compared to many literature methods.

154 We evaluate this descriptor in recognition task using different classifiers on
155 two popular DT datasets. Then, we illustrate the potential reactivity of the
156 model on a DT segmentation example. This paper is organized as follows:
157 Section 2 presents the related background on CN and RNN. Section 3 explains
158 our approach in detail. Section 4 explains our experiments and shows some
159 results. Section 5 presents and discusses quantitative results on DT recognition,
160 and qualitative results on reactive DT segmentation. Finally, Section 6 presents
161 the conclusions and future works.

162 **2. Methodologies**

163 *2.1. Complex Networks*

164 The complex network research field emerges from the intersection of the areas
165 of graph theory, physics, statistics and computer science, in order to understand
166 and analyze complex systems. Indeed, many systems and data are formed by
167 a set of elements that interact with each other and that can be represented as
168 networks by defining the entities (vertices) and the relationships among them
169 (edges). Some examples of these systems are the internet, the WWW and social
170 networks. In particular, researches in scale-free networks [43], identification
171 of community structure in many networks [44] and small-world [45] networks

172 have drawn attention from the scientific community on the study of complex
 173 networks, which is a multidisciplinary research field. In computer vision, the
 174 flexibility and expressiveness of this framework have been used over the last years
 175 for color-texture classification with multi-layer CN [46] and dynamic texture
 176 analysis with diffusion in networks [3].

177 Formally, a complex network can be defined as a graph $G = (V, E)$, where
 178 V is the set of vertices and E the set of edges connecting vertex pairs. In this
 179 work, we adopted the term graph to refer to a complex network in order to
 180 avoid confusion with the neural network term. The graphs can be directed and
 181 weighted, which is when the edge e_{ij} is directed from i to j and has a weight
 182 $w_{ij} \in \mathbb{R}$. For description, two important information about the graph can be
 183 used: the out-degree and the weighted out-degree or strength. The out-degree
 184 of a vertex i counts the number of connections from i :

$$k_i = |\{v_j; e_{ij} \in E\}|, \quad (1)$$

185 where $|S|$ denotes the cardinality of set S . The weighted out-degree s_i computes
 186 the sum of the weights of all connections from i :

$$s_i = \sum_{e_{ij} \in E} w_{ij}. \quad (2)$$

187 These measures quantify topological characteristics and different properties
 188 of the graph that are useful for the identification of hubs, scale-free property,
 189 etc.

190 2.2. Randomized Neural Networks

191 A well known problem in neural networks is that gradient descent based
 192 learning is slow and needs a huge quantity of data, specially when the number
 193 of neural weights is large. To tackle this issue, randomized neural networks
 194 [47, 48, 49, 50] were proposed. In their simplest version, these networks have
 195 a single hidden layer whose weights are random, and an output layer whose
 196 weights can be computed using a closed-form solution.

197 To mathematically describe the neural network used in this work, let $Z =$
 198 $[\vec{z}_1, \vec{z}_2, \dots, \vec{z}_N]$ be a matrix composed of the outputs of the hidden layer, accord-
 199 ing to the following equation

$$Z = \phi \left([w_1, w_2, \dots, w_Q]^T [x_1, x_2, \dots, x_N] \right) \quad (3)$$

200 where T denotes transpose operation, $\vec{x}_i = [-1, x_{i1}, x_{i2}, \dots, x_{ip}]^T$ is an input
 201 vector i with $p + 1$ attributes, $w_q = [w_{q0}, w_{q1}, \dots, w_{qp}]^T$ is the set of random
 202 weights of a determined neuron q , N is the number of feature vectors \vec{x}_i , Q is
 203 the number of neuron units of the hidden layer, and $\phi(\cdot)$ is a transfer function.

204 Next, after adding a constant value -1 to each vector \vec{z}_i for the bias weights of
 205 the output layer neurons, the aim of the closed-form training is to find a matrix
 206 M that satisfies $D = MZ$, where $D = [\vec{d}_1, \vec{d}_2, \dots, \vec{d}_N]$ is a matrix of (ground
 207 truth) label vectors, each one corresponding to its respective input \vec{x}_i . To
 208 compute M , it is possible to use the Moore-Penrose pseudo-inverse [51, 52] with
 209 the Tikhonov regularization [53, 54], thus resulting in the following equation

$$M = DZ^T(ZZ^T + \lambda I)^{-1} \quad (4)$$

210 where $0 < \lambda < 1$ and I is an identity matrix of size $(Q + 1) \times (Q + 1)$.

211 3. Proposed Approach

212 In this section, we describe the CPNN approach, which extends to dynamic
 213 textures the static texture characterization approach proposed in [41]. In the
 214 first step, the dynamic texture video is modeled into two directed graphs: spatial
 215 and temporal graphs. Then, information from these graphs are used to train
 216 the neural networks and build a signature.

217 3.1. Modeling Dynamic Texture in Directed Graphs

218 In order to analyze the dynamic textures, it is important to capture features
 219 that represent the appearance and motion properties of the video. To this end,

220 in the proposed approach, we model the dynamic texture video into two di-
 221 rected graphs: the spatial graph $G_S = (V_S, E_S)$ that represents the appearance
 222 properties and the temporal graph $G_T = (V_T, E_T)$ that contains the motion
 223 characteristics. Each pixel $i = (x_i, y_i, t_i)$ of the graphs is represented by a ver-
 224 tex $i \in V$, such that x_i and y_i are the spatial coordinates and t_i the temporal
 225 index of the pixel i .

226 The two graphs have as main difference the definition of the set of edges. In
 227 the spatial graph, in order to characterize appearance, we connect only vertices
 228 that are from the same frame. Thus, the set E_S is given by the connection of all
 229 vertices whose distance is smaller than or equal to a given radius r and whose
 230 time coordinates t_i and t_j are equal (as illustrated in Figure 1(a)),

$$e_{ij} \in E_S \iff ((x_i - x_j)^2 + (y_i - y_j)^2 + (t_i - t_j)^2) \leq r \text{ and } t_i = t_j \quad (5)$$

231 On the other hand, in the temporal graph, we focus on the relationships
 232 between frames to analyze the motion characteristics. In this way, in the set
 233 E_T , we connect all vertices whose distance is smaller than or equal to r and
 234 their time coordinates are different (as exemplified in Figure 1(b)),

$$e_{ij} \in E_T \iff ((x_i - x_j)^2 + (y_i - y_j)^2 + (t_i - t_j)^2) \leq r \text{ and } t_i \neq t_j \quad (6)$$

235 Each pixel represented by a vertex has a different gray-level, which relates
 236 to the texture patterns. To add this information to the graph topology, we
 237 transform it to a directed and weighted graph, as follows: First, for each edge
 238 a direction is defined based on the order of gray-level. Specifically, in an edge
 239 e_{ij} the vertex i points to the vertex j if $I(i) \leq I(j)$. In addition, a weight w_{ij}
 240 is defined from the difference of intensities and distance between the two pixels

241 that represent the vertices:

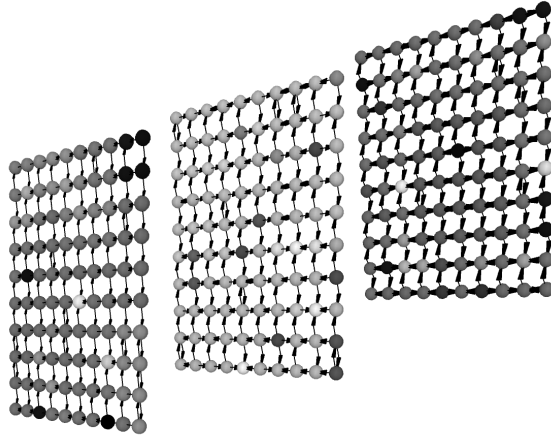
$$w_{ij} = \begin{cases} \frac{|I(i)-I(j)|}{L}, & \text{if } r = 1 \\ \frac{(\frac{dist(i,j)-1}{r-1}) + (\frac{|I(i)-I(j)|}{L})}{2}, & \text{otherwise.} \end{cases} \quad (7)$$

242 where $I(i) \in [0, 255]$ is the gray-level of the pixel i , $dist(i, j)$ is the Euclidean
 243 distance between the pixels i and j and L is the highest gray-level. When the
 244 radius is $r = 1$ then the weight w_{ij} is the difference of gray levels normalized
 245 by the maximum gray-level L , producing a value in $[0, 1]$. On the other hand,
 246 if $r > 1$ then the weight w_{ij} is given by the average of the distance between
 247 the pixels normalized by the maximum distance r , and the normalized gray
 248 scale difference, which also provides a value in $[0, 1]$. Thus, this weight function
 249 includes information about the pixel neighborhood and the difference of gray-
 250 level in order to balance the importance between geometric information and
 251 color in the texture representation [40].

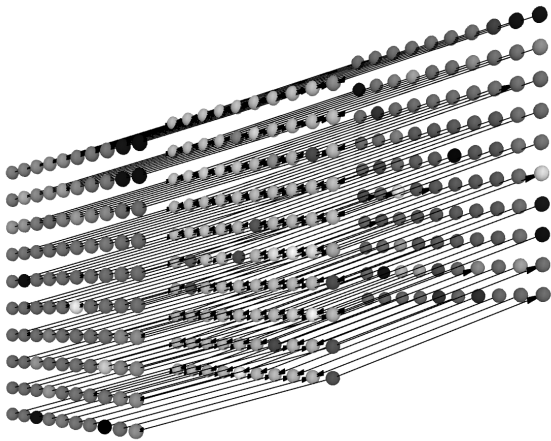
252 3.2. Proposed Signature

253 In this approach, we train the randomized neural network with topologi-
 254 cal characteristics from graphs that model the videos of dynamic textures. For
 255 training the RNNs, the out-degree k_i and the weighted out-degree s_i of each ver-
 256 tex from the modeled graph compose the input matrices. The learned weights
 257 from the output layer form the feature vector for dynamic texture representa-
 258 tion. The main steps of the proposed method are summarized in the flowchart
 259 diagram in Figure 3.

260 In order to create the matrix of input vectors for randomized neural net-
 261 work, we use the evolution of the graph for different values of modeling pa-
 262 rameter r . Therefore, for each vertex i of the graph an input vector and its
 263 corresponding output label are defined as follows: the out-degree values of
 264 the vertex for different radii $\{r_1, r_2, \dots, r_R\}$ are considered as the input vec-
 265 tor $\vec{x}_i = [k_i^1, k_i^2, k_i^3, \dots, k_i^R]$, where R is the maximum radius value. The output
 266 label is simply the gray-level $d_i = I(i)$. A matrix of input vectors $X_{(k)}$ for the
 267 out-degree and a vector of output labels D is obtained considering all vertices



(a) Spatial graph



(b) Temporal graph

Figure 1: Modeling of a three frame video in a (a) spatial graph and a (b) temporal graph (using $r = 1$).

268 of the graph. The number of columns of $X_{(k)}$ (and the size of vector D) is then
 269 the number of samples N which corresponds to all the pixels, but could also be
 270 any sub-sample of the video. In this way, it is possible to statistically analyze
 271 the topology evolution of the vertices that represent the pixels with different
 272 gray-levels. Figure 2(a) illustrates the step for obtaining the matrices of input
 273 vector and label from a spatial graph. In addition to the matrix $X_{(k)}$, we also

274 build the matrix of input vectors for the weighted out-degree $X_{(s)}$.

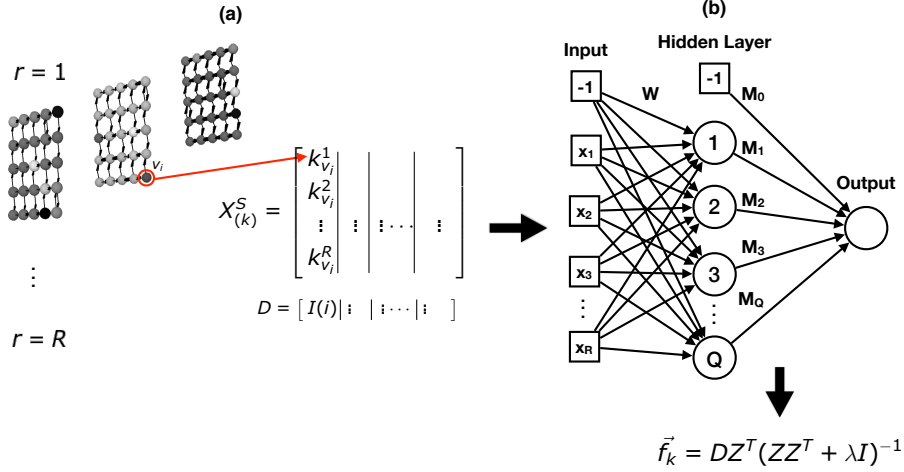


Figure 2: Construction of the output vector and label from a spatial graph by modeling dynamic textures with different values of r and using the out-degree k_i .

275 In the next step, the weights of the matrix W of the input layer of the
 276 randomized neural network are generated randomly. However, in the methods
 277 for dynamic texture analysis it is important that the feature vector be always
 278 the same for a given sample. In this sense, in order to always obtain the same
 279 weight values, we use the Linear Congruent Generator (LCG) [55, 56] with fixed
 280 parameters to generate the uniform pseudo random numbers for the matrix W ,

$$V(n+1) = (a * V(n) + b) \bmod c, \quad (8)$$

281 where V is the random sequence of length $E = Q * (p + 1)$ and started by
 282 $V(1) = E + 1$. The values of a , b and c are parameters defined as $a = E + 2$,
 283 $b = E + 3$ and $c = E^2$ (these values were adopted in [57]). Thus, the matrix
 284 W is composed of the vector V divided into Q segments of values $p + 1$. The
 285 values of the matrix W and X (each row) are normalized using standard score
 286 (zero mean and unit variance).

287 The feature vector that represents the dynamic textures is built based on
 288 matrix M , which becomes here a vector \vec{f} (since the output labels are scalar),

289 which is computed as: $\vec{f} = DZ^T(ZZ^T + \lambda I)^{-1}$, such that $\lambda = 10^{-3}$ (Figure
 290 2(b)) and the length of \vec{f} is $Q + 1$ because of the bias value. To characterize
 291 appearance and movement of dynamic textures, we propose to use the spatial
 292 graph and temporal graph as inputs to train the randomized neural network.
 293 Therefore, firstly, two randomized neural networks are trained, each one with
 294 a different matrix of input vectors $X_{(k)}^S$ and $X_{(s)}^S$ extracted from the spatial
 295 graphs. From these trained randomized neural networks, we obtain two vectors
 296 \vec{f}_k^S and \vec{f}_s^S . For the temporal graph, the same procedure is performed and two
 297 vectors are obtained \vec{f}_k^T and \vec{f}_s^T using the matrices $X_{(k)}^T$ and $X_{(s)}^T$, respectively.
 298 In this way, to represent the appearance and motion of the dynamic texture,
 299 the following concatenation is proposed:

$$\vec{\Upsilon}(Q)_R = [\vec{f}_k^S, \vec{f}_s^S, \vec{f}_k^T, \vec{f}_s^T], \quad (9)$$

300 where Q is the number of hidden layer neurons and R is the maximum radius
 301 of graph modeling, i.e., the number of radii used to construct the matrix X .
 302 Figure 3 illustrates the steps to obtain the vector $\vec{\Upsilon}(Q)_R$, which is built using
 303 a single value of Q and R . These two parameters influence the training of the
 304 neural network and, therefore, different characteristics are learned for different
 305 parameter values. Thus, the vector $\vec{\Psi}(Q)_{R_1, R_2}$ that concatenates the vectors
 306 $\vec{\Upsilon}(Q)_R$ for different values of R is used:

$$\vec{\Psi}(Q)_{R_1, R_2} = [\vec{\Upsilon}(Q)_{R_1}, \vec{\Upsilon}(Q)_{R_2}]. \quad (10)$$

307 Finally, we propose a feature vector $\vec{\Theta}(R)_{(Q_1, Q_2, Q_m)}$ that combines the vector
 308 $\vec{\Psi}(Q)_{R_1, R_2}$ for different numbers of neurons:

$$\vec{\Theta}_{Q_1, Q_2, \dots, Q_m} = [\vec{\Psi}(Q_1)_{R_1, R_2}, \vec{\Psi}(Q_2)_{R_1, R_2}, \dots, \vec{\Psi}(Q_m)_{R_1, R_2}]. \quad (11)$$

309 The overall algorithm of the proposed method is described in Algorithm 1.

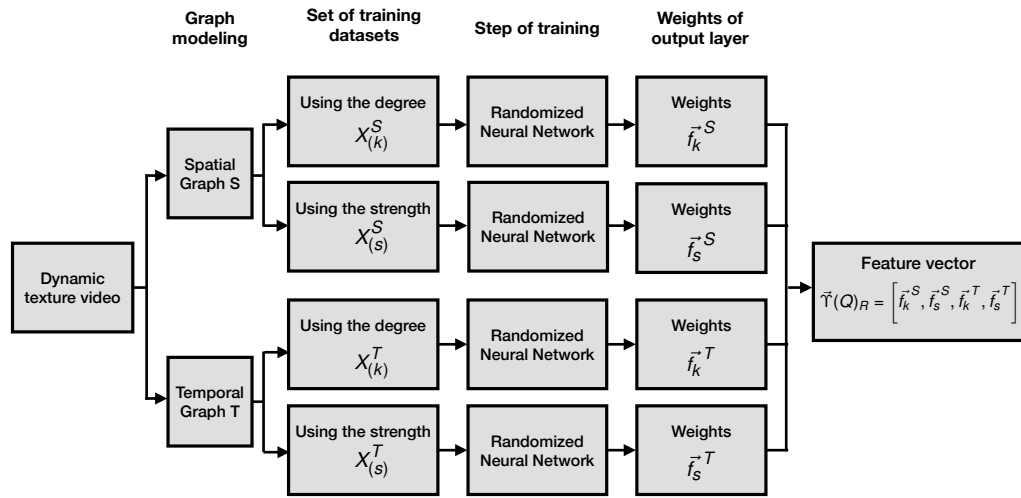


Figure 3: Flowchart diagram of the proposed method.

Algorithm 1: CPNN method

Data: Video V

Result: Feature Vector $\vec{Y}(Q)_R$

Parameter: Number of hidden neurons Q , maximum radius R

/ Feature Vector for a video V and parameters Q and R */*

$\vec{Y}(Q)_R \leftarrow \text{CPNN}(V, Q, R)$

Function CPNN(V, Q, R):

/ computes the graph measures of the vertices for each
radius and builds the input matrices */*

for $r \leftarrow 1$ **to** R **do**

$X_{(k)}^S(r, :) \leftarrow \text{SpatialGraphDegree}(V, r)$

$X_{(s)}^S(r, :) \leftarrow \text{SpatialGraphStrength}(V, r)$

$X_{(k)}^T(r, :) \leftarrow \text{TemporalGraphDegree}(V, r)$

$X_{(s)}^T(r, :) \leftarrow \text{TemporalGraphStrength}(V, r)$

end

$D \leftarrow V$ // the label vector is the gray-scale values of the
video pixels

/ trains the RNN for each input matrix and obtains the
output weights */*

$\vec{f}_k^S \leftarrow \text{trainRNN}(X_{(k)}^S, D, Q)$

$\vec{f}_s^S \leftarrow \text{trainRNN}(X_{(s)}^S, D, Q)$

$\vec{f}_k^T \leftarrow \text{trainRNN}(X_{(k)}^T, D, Q)$

$\vec{f}_s^T \leftarrow \text{trainRNN}(X_{(s)}^T, D, Q)$

$\vec{Y}(Q)_R \leftarrow [\vec{f}_k^S, \vec{f}_s^S, \vec{f}_k^T, \vec{f}_s^T]$ // combines the output weights in
a unique feature vector

return $\vec{Y}(Q)_R$

```

Function trainRNN( $X, D, Q$ ):
  X = Zscore (X) // normalize the input matrix

  W = LCG (Q,P+1,Q*(P+1)) // generate the random weights

  X = addBias (X,-1) // add the bias in the input matrix

  Z = Activation (W*X) // activation function of the hidden
    layer

  Z = addBias (Z,-1) // add bias in the Z

  lambda = 0.001

  M = (D*Z')/(Z*Z' + lambda * eye(Q+1)) // calculates the
    output weights with the Moore-Penrose pseudo-inverse

  return M

```

311 4. Validation Setup

312 To evaluate our method, two benchmark databases were used. They are:

- 313 • Dyntex++ [58]: this database, which is a compilation of the Dyntex
 314 databases [59], has 3600 videos divided into 36 classes, 100 videos per
 315 class, each one of size $50 \times 50 \times 50$. Figure 4(a) shows examples of the
 316 first frame of samples from Dyntex++.
- 317 • UCLA-50 [60]: this database is composed of 50 classes, 4 videos per class,
 318 each one of size $75 \times 48 \times 48$. Also, two variations from the UCLA-50,
 319 both proposed in [36], were used in our experiments. The first (UCLA-9)
 320 combines videos taken from different viewpoints and groups them into 9
 321 classes: smoke (4 samples), flowers (12), boiling water (8), sea (12), fire
 322 (8), water (12), fountains (20), waterfall (16) and plants (108). The second
 323 (UCLA-8) discards the class “plants” because it has a large set of samples
 324 when compared to the other classes. The first frame of some samples from
 325 the UCLA database is shown in Figure 4(b).

326 In the validation procedure, we used the 1-NN classifier (we used a imple-
 327 mentation from Weka [61]) in order to compare our results to other descriptors
 328 available in the literature. As evaluation protocol, we employed the procedure
 329 adopted in [36, 58, 25] for the UCLA-50, UCLA-9, and UCLA-8; and adopted
 330 in [40] for the Dyntex++. Thus, we used 4-fold and 10-fold cross-validation for
 331 the UCLA-50 and Dyntex++, respectively. In this case, we use 10 trials for a
 332 statistically more reliable result. On the UCLA-9 and UCLA-8 databases, we
 333 used hold-out (50% of samples for training and the remainder for test). Av-
 334 erage accuracy and standard deviation of 20 trials were used to quantify the
 335 performance of our method. The graph modeling step of the proposed method
 336 was programmed in the C language, while for training and feature extraction
 337 with the RNN was used the Matlab 9.2 software. The trained weights of the
 338 output layer (i.e., the feature vectors) computed by the proposed method in all
 339 databases are available in GitHub ¹.

340 5. Results and Discussions

341 5.1. Dynamic Texture Classification

342 Firstly, we perform a parametric analysis of our approach. For this, a feature
 343 vector is extracted of each sample from UCLA-50 and Dyntex++ databases
 344 using different values of Q and R . Figure 5 summarizes the average accuracy
 345 for the UCLA-50 and Dyntex++ databases using the feature vector $\vec{\Psi}(Q)_{R_1, R_2}$
 346 with $Q = 4$ and different values and combinations of R . In the plot, the rows
 347 represent the values of R_1 and the columns the values of R_2 . The main diagonal
 348 corresponds to a single value of R used to build the feature vector. In this last
 349 case, the highest average accuracy, 94.51%, is obtained using $R = 4$. On the
 350 other hand, when using two values of R combined, the highest average accuracy
 351 is achieved using the combination $(R_1, R_2) = (4, 10)$. This indicates that the
 352 combination of local (small radius) and regional (large radius) information from

¹<https://github.com/lucascorreiaribas/CPNN>

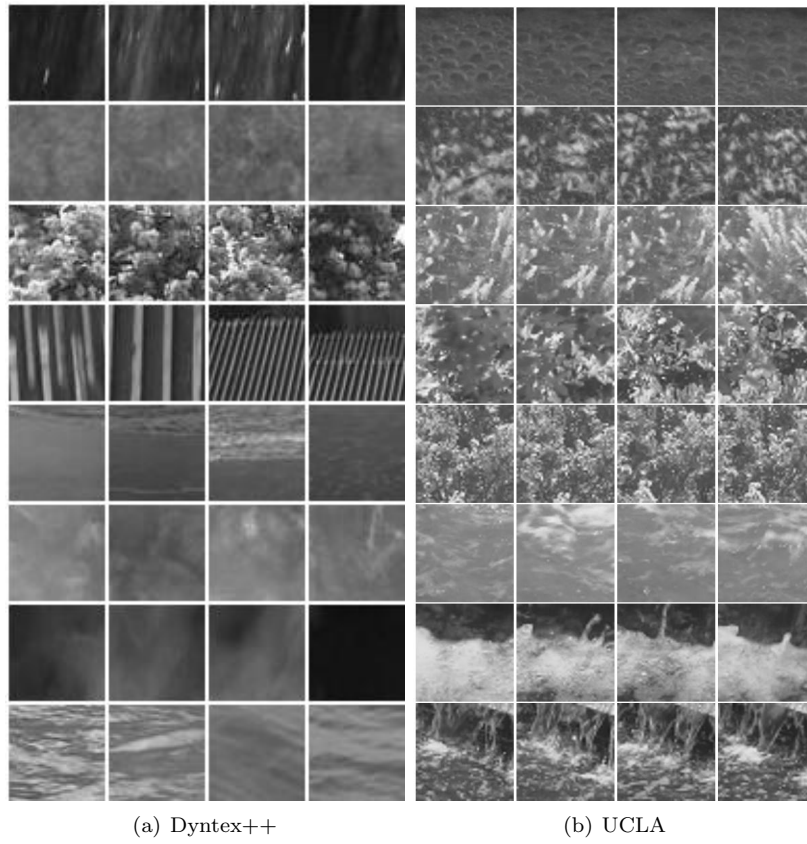


Figure 4: Initial frames of dynamic texture samples from the databases. Each row represents a class.

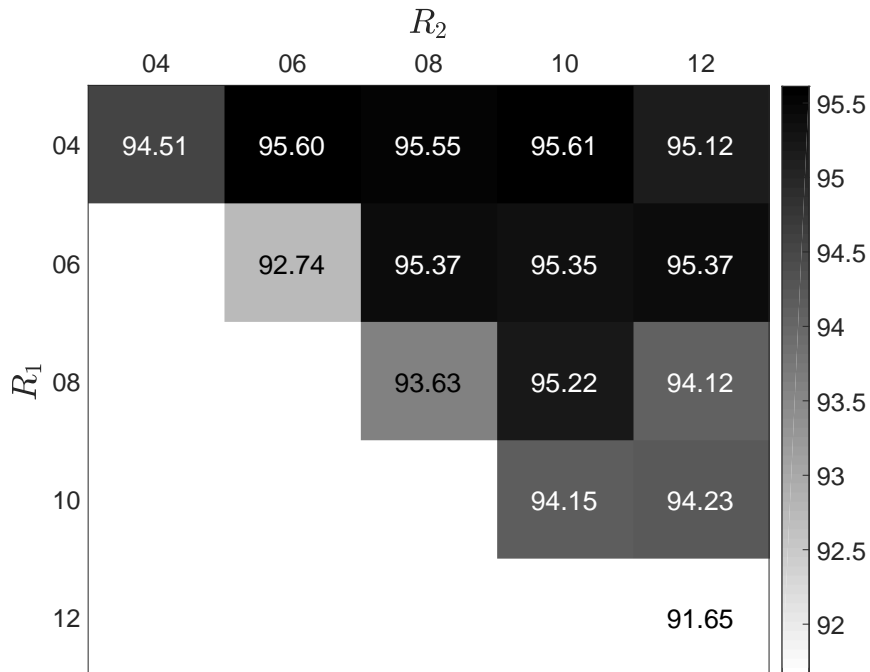


Figure 5: Accuracy for different values and combination of one or two radii, averaged on the datasets UCLA-50 and Dyntex++.

353 graphs is more discriminative.

354 Figure 6 presents the accuracies using the feature vector $\vec{\Theta}_{Q_1, Q_2, \dots, Q_m}$ with
 355 one and two values of Q on the UCLA-50 and Dyntex++ databases. It can be
 356 seen that, on the UCLA-50 database, the higher accuracies are obtained using
 357 large values of Q (all combinations with $Q = 29$), while small values of Q pro-
 358 vide better accuracies on Dyntex++ database. In a second experiment, we test
 359 the feature vector $\vec{\Theta}_{Q_1, Q_2, \dots, Q_m}$ combining three different values of Q , as it can
 360 be seen in Table 1. Note that, as we increase the values of Q in the combina-
 361 tions, the accuracy tends to stabilize and then decrease, meanwhile the size of
 362 the feature vector is increased. In this sense, the combination $\{4, 24, 29\}$ can be
 363 considered as a good trade-off between the size of feature vectors and accuracy

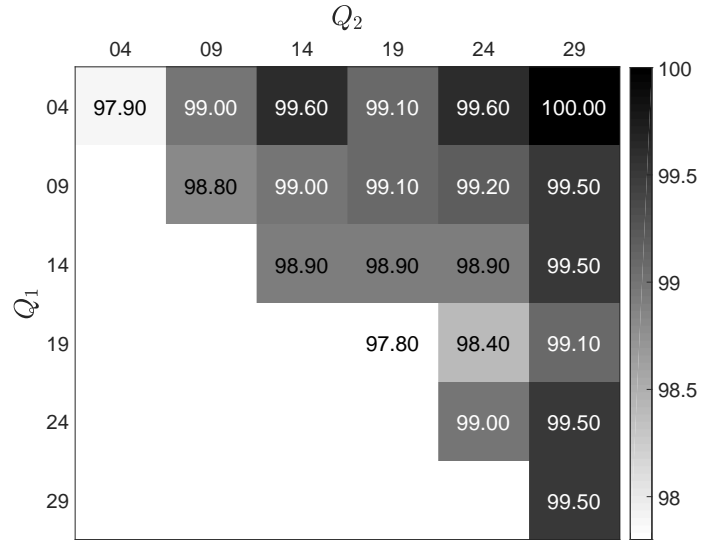
Table 1: Accuracy for the feature vector $\vec{\Theta}_{Q_1, Q_2, \dots, Q_m}$ using three value combinations for Q .

Q	# of features	UCLA	Dyntex++
{04, 09, 14}	240	99.10 (\pm 1.21)	95.06 (\pm 1.04)
{04, 09, 19}	280	99.10 (\pm 1.02)	95.08 (\pm 1.14)
{04, 09, 24}	320	99.50 (\pm 0.89)	95.19 (\pm 1.10)
{04, 09, 29}	360	99.50 (\pm 0.89)	94.71 (\pm 1.21)
{04, 14, 19}	320	99.30 (\pm 0.98)	95.38 (\pm 0.97)
{04, 14, 24}	360	99.40 (\pm 0.94)	95.32 (\pm 1.09)
{04, 14, 29}	400	99.60 (\pm 0.82)	95.05 (\pm 1.16)
{04, 19, 24}	400	99.10 (\pm 1.21)	95.10 (\pm 1.10)
{04, 19, 29}	440	99.50 (\pm 0.89)	94.86 (\pm 1.18)
{04, 24, 29}	480	99.92 (\pm 0.37)	95.03 (\pm 1.27)
{09, 14, 19}	360	99.10 (\pm 1.02)	95.18 (\pm 0.91)
{09, 14, 24}	400	99.50 (\pm 0.89)	95.18 (\pm 1.04)
{09, 14, 29}	440	99.50 (\pm 0.89)	94.90 (\pm 1.10)
{09, 19, 24}	440	98.80 (\pm 1.36)	94.81 (\pm 1.15)
{09, 19, 29}	480	99.10 (\pm 1.02)	94.69 (\pm 1.21)
{09, 24, 29}	520	99.30 (\pm 0.98)	94.87 (\pm 1.12)
{14, 19, 24}	480	98.80 (\pm 1.36)	94.94 (\pm 1.02)
{14, 19, 29}	520	99.10 (\pm 1.02)	94.72 (\pm 1.19)
{14, 24, 29}	560	99.40 (\pm 0.94)	94.72 (\pm 1.01)
{19, 24, 29}	600	99.00 (\pm 1.21)	94.24 (\pm 1.11)

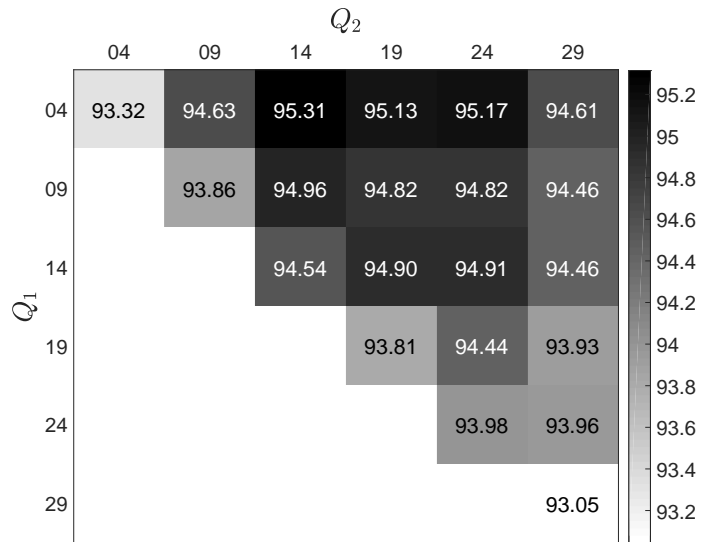
364 on both databases. However, it is important to emphasize that the method
365 obtained good results for a large number of parameter values, demonstrating
366 robustness to parameter changes. For example, the combination $\{4, 14\}$ pro-
367 duces 160 features, and the accuracies are 99.60% and 95.31% on the UCLA-50
368 and Dyntex++ databases, respectively.

369 We also compare our experimental results in the classification task to the
370 results obtained by other existing methods. Table 2 presents the accuracy ob-
371 tained for different methods on UCLA-50 database. In these experiments, the
372 results indicate that the proposed method achieved the highest accuracy, which
373 is followed by the CNN-GoogLeNet [33] approach with an accuracy of 99.50%.
374 The next accuracy of 97.15% was obtained by the Randomized Neural Network
375 based signature (RNN-DT) [35] method.

376 A statistical hypothesis test was performed to evaluate the significance of
377 the difference in performance between the proposed method and the compared



(a) UCLA-50



(b) Dyntex++

Figure 6: Classification results for the feature vector $\vec{\Theta}_{Q_1, Q_2, \dots, Q_m}$ with one single ($m = 1$) or two different ($m = 2$) values of Q on the Dyntex++ and UCLA-50 databases. The diagonal cells correspond to single values of Q , the others to combinations of two distinct values of Q .

378 methods on the UCLA-50 database. For this purpose, we employ the paired-
 379 sample t-test and Wilcoxon signed-rank test [62] from the Matlab implementa-
 380 tion with a significance level equal to $\alpha = 0.05$. To perform this test, the
 381 accuracy of the proposed method, Diffusion, DPSWNet, and RNN-DT methods
 382 were computed as the average of 20 trials. Thus, the null hypothesis is that the
 383 performance of the proposed method is equal to the compared methods, while
 384 in the alternative hypothesis the proposed method is statistically superior to
 385 the others. When our hypothesis is tested in the comparison with the Diffu-
 386 sion method, the p -value is $p = 5.687e-19$ and $p_w = 1.255e-5$ to the t-test
 387 and Wilcoxon test, respectively. Since p and $p_w < \alpha$ the null-hypothesis can
 388 be rejected in favor of alternative hypothesis, confirming that the performance
 389 (mean and median accuracy for t-test and Wilcoxon test, respectively) of the
 390 proposed method is statistically superior to the Diffusion method in UCLA-50
 391 database. The p -values for the comparison with the RNN-DT ($p = 1.844e-17$
 392 and $p_w = 3.094e-5$) and DPSWNet ($p = 6.683e-20$ and $p_w = 1.614e-5$)
 393 methods are also much lower than α in the two tests, so we can reject the
 394 null-hypothesis. Regarding the GoogLeNet method, the result of the statistical
 395 test must be taken with caution because we use the result of the original paper.
 396 Therefore, our hypothesis is that the proposed method improved the result in re-
 397 lation to the GoogleLeNet in 0.25%, so the null-hypothesis is the negation of this
 398 hypothesis. Based on the t-test and Wilcoxon test, the p -values are $p = 0.0002$
 399 and $p_w = 0.00093$, respectively. These values reject the null-hypothesis and
 400 indicate the superiority of the proposed method.

401 Table 3 provides a comparison of literature methods on the UCLA-8 and
 402 UCLA-9 databases, which consider videos taken from different viewpoints in
 403 the same class. On the UCLA-8 database, the proposed method obtained the
 404 second-highest accuracy (98.94%), which is very similar to the highest accu-
 405 racy achieved by the CNN-GoogLeNet (99.02%). On the other hand, on the
 406 UCLA-9 database, the DPSWNet [42] method achieved the highest accuracy
 407 (99.10%) while the proposed method obtained 98.19%. Table 4 presents the
 408 comparison results for the Dyntex++ database. On this database, our pro-

Table 2: Comparison in the 50-class UCLA database (1-NN classifier and 4-fold cross validation). For the methods with '*', the results are taken from [25], [24] and [40], the results with '+' are obtained from the original paper, while the others were generated.

Descriptor	ACC (%)
KDT-MD* [63]	89.50
DFS* [2]	89.50
3D-OTF* [64]	87.10
CVLBP* [30]	93.00
HLBP* [25]	95.00
MEWLSP* [24]	96.50
LBP-TOP [22]*	94.50
VLBP* [23]	89.5
DPSW [39]	94.60 (± 1.98)
CNN-GoogLeNet+ [33]	99.50
CNDT [40]	94.62 (± 1.04)
Diffusion+ [3]	98.50 (± 3.37)
DPSWNet+ [42]	98.00 (± 3.50)
RNN-DT+ [35]	97.05 (± 1.87)
CPNN	99.92 (± 0.37)

409 posed method was surpassed by the Local Binary Patterns (LBP-TOP) [22],
410 RNN-DT [35] and RI-VLBP [23]. In this way, we can affirm that our result
411 is not statistically superior to the others in these two databases. However, it
412 is important to emphasize some aspects of the methods. CNNs are currently
413 among the most powerful approaches in image analysis and have high compu-
414 tational cost. Also, they require a large number of samples for training, which
415 are drawbacks compared to our method. The RNN-DT is a state-of-the-art
416 method that shares with the CPNN the same fundamental core of using neural
417 weights as descriptors. Furthermore, our proposed approach produces a smaller
418 feature vector (480 descriptors but the combination {4,14} produces 160 de-
419 scriptors with competitive accuracies) when compared to the LBP-TOP (768
420 descriptors) and RI-VLBP (16 384 descriptors), MBSIF (6 144 descriptors) and
421 MEWLSP (1 536 descriptors). The feature vector size can be crucial in some ap-
422 plications which require low computational cost to classify the DT and memory
423 consumption.

424 We also compared the results of the proposed method with other approaches
425 in the literature that are more similar to ours. In particular, the RNN-DT

426 method uses the same neural network architecture considered in our work, but
427 in a three orthogonal planes scheme to train the model, without the graph mod-
428 eling information. The RNN-DT method obtained a slightly higher accuracy
429 in the 9-class UCLA and Dyntex++ databases, while the proposed approach
430 improves the accuracy in 2.95% and 1.2% when compared with the RNN-DT
431 method on the 50-class and 8-class UCLA databases, respectively. On the other
432 hand, the CNDT and DPSWNet methods employed different ways to model the
433 dynamic texture in graphs with topological statistical measures as descriptors.
434 In relation to these methods, with the exception of the 9-class UCLA database,
435 the proposed method obtained higher accuracies in all databases. These results
436 indicate that the combination of the graph-based description and learned repre-
437 sentation from RNNs can improve the characterization of the dynamic texture.

Table 3: Comparison of the proposed method with other dynamic texture methods in the 9-class and 8-class UCLA databases (1-NN classifier and half of the samples for training and the remainder for testing). The results with '*' are taken from [25] and [24], the results with '+' are obtained from the original paper, while the others were generated.

Descriptor	ACC (%)	
	9-class UCLA	8-class UCLA
3D-OTF* [64]	96.32	95.80
CVLBP* [30]	96.90	95.65
HLBP* [25]	98.35	97.50
MEWLSP* [24]	98.55	98.04
MBSIF* [10]	98.75	97.80
High level feature* [65]	92.67	85.65
DNGP* [12]	98.10	97.00
WMFS* [66]	96.95	97.18
Chaotic vector* [38]	85.10	85.00
VLBP* [23]	96.30	91.96
LBP-TOP* [22]	96.00	93.67
CNN-GoogLeNet+ [33]	98.35	99.02
DPSW [39]	96.33 (± 2.46)	93.41 (± 6.01)
CNDT [40]	95.61 (± 2.72)	94.32 (± 4.18)
DPSWNet+ [42]	99.10 (± 0.86)	96.55 (± 7.13)
Diffusion+ [3]	97.80 (± 1.53)	96.22 (± 4.80)
RNN-DT+ [35]	98.54 (± 1.56)	97.74 (± 2.99)
CPNN	98.19 (± 2.27)	98.94 (± 1.42)

438 In addition to 1-NN, we also consider other classifiers to evaluate the poten-

Table 4: Comparison of the proposed method and others in the Dyntex++ database (1-NN classifier and 10-fold cross validation). The results with '*' are obtained from [35], the results with '+' are taken from the original paper, while the others were generated.

Descriptor	ACC (%)
VLBP [23]*	96.14 (± 0.77)
LBP-TOP [22]*	97.72 (± 0.43)
DPSW [39]*	91.39 (± 1.29)
CNDT [40]*	83.86 (± 1.40)
DPSWNet [42] ⁺	93.50 (± 1.27)
Diffusion [3] ⁺	93.80 (± 1.08)
RNN-DT [35] ⁺	96.51 (± 0.94)
CPNN	95.03 (± 1.27)

439 tial of the methods. The classifiers are: Random Forest [67], Deep Random Vec-
440 tor Functional Link - D-RVFL [68] and Linear Discriminant Analysis - LDA [69].
441 We consider the dynamic texture descriptors with the features and source codes
442 available. Table 5 shows the accuracies obtained on the UCLA-50, UCLA-9 and
443 UCLA-8 databases. In the table, the rows represent the different DT descriptors,
444 while the columns represent the different classifiers evaluated on each database.
445 On the UCLA-50 database, the proposed method achieved the highest accuracy
446 in all classifiers, which are very similar to the RNN-DT and DPSWNet meth-
447 ods. On the other hand, on the UCLA-9 and UCLA-8 databases, the RNN-DT
448 method had a better performance, except for the UCLA-9 using the Random
449 Forest and D-RVFL classifiers, where our method had a slightly higher accuracy.
450 In particular, we can observe that on UCLA-50 database and using the D-RVFL
451 classifier, our descriptor CPNN achieved a good performance compared to other
452 descriptors, such as Diffusion, RI-VLBP, and DPSW methods. The core of the
453 CPNN method is learning features from randomized neural networks, which can
454 be viewed as a variant of the RVFL network. This can explain the good perfor-
455 mance of these methods since their features are the weights of the output layer.
456 This argument motivates the investigation for extending our method to other
457 neural network architectures with more layers, although this may penalize the
458 computational competitiveness of the method. On the Dyntex++ database, the
459 proposed method also obtained the highest accuracy for the D-RVFL classifier,

Table 5: Accuracy obtained on the UCLA databases by different dynamic texture descriptors (represented in the rows) using the Random Forest, D-RVFL and LDA classifiers.

Descriptor	Random Forest			D-RVFL			LDA		
	UCLA-50	UCLA-9	UCLA-8	UCLA-50	UCLA-9	UCLA-8	UCLA-50	UCLA-9	UCLA-8
VLBP	80.37 (1.49)	86.79 (4.03)	82.84 (6.97)	81.10 (1.10)	73.38 (4.37)	56.48 (4.79)	72.87 (1.11)	86.32 (3.18)	85.00 (6.32)
DPSWNet	97.30 (0.54)	91.53 (4.03)	93.07 (4.27)	94.15 (0.97)	86.84 (3.05)	65.45 (2.82)	98.37 (0.48)	94.90 (2.14)	92.50 (6.22)
DPSW	95.75 (1.32)	92.24 (3.21)	92.50 (4.78)	83.45 (2.11)	84.18 (3.33)	60.22 (7.35)	88.12 (1.44)	94.13 (4.09)	88.97 (6.09)
CNDT	94.37 (0.25)	92.04 (3.75)	90.00 (6.93)	91.75 (0.87)	87.65 (3.03)	63.29 (4.96)	97.25 (0.29)	92.75 (4.73)	92.16 (5.59)
Diffusion	97.00 (0.71)	93.32 (2.86)	88.52 (6.66)	52.75 (4.72)	85.25 (2.23)	62.95 (3.84)	97.87 (0.85)	94.54 (3.06)	95.34 (4.51)
RNN-DT	98.90 (0.32)	92.04 (2.35)	94.32 (3.87)	94.25 (0.50)	88.01 (2.45)	66.93 (1.72)	99.50 (0.41)	97.19 (2.43)	98.07 (3.77)
CPNN	99.05 (0.44)	93.62 (3.32)	90.68 (4.66)	96.75 (0.87)	88.11 (2.67)	64.43 (3.63)	99.87 (0.25)	96.38 (3.71)	94.89 (4.77)

Table 6: Accuracy **generated for** the Dyntex++ database by different dynamic texture descriptors (represented in the rows) using the Random Forest, D-RVFL and LDA classifiers.

Descriptor	Random Forest	D-RVFL	LDA
VLBP	84.47 (0.26)	79.48 (0.48)	89.54 (0.35)
DPSWNet	90.60 (0.25)	85.98 (0.31)	85.61 (0.25)
DPSW	90.17 (0.08)	63.13 (0.42)	83.22 (0.17)
CNDT	83.61 (0.51)	84.34 (0.26)	90.27 (0.17)
Diffusion	91.72 (0.16)	79.22 (0.22)	83.17 (0.16)
RNN-DT	95.37 (0.10)	48.22 (0.34)	88.89 (0.28)
CPNN	93.44 (0.27)	89.17 (0.34)	87.74 (0.15)

460 as can be seen in Table 6. However, for the Random Forest and LDA classifiers,
 461 our results indicate that some classifiers may not be adapted to the proposed
 462 descriptor. Indeed, the performance of the classifiers depends on several issues
 463 such as the nature of the features, tuning of the parameters, etc. Thus, in
 464 some cases, simple classifiers can obtain best performances than more complex
 465 classifiers [70].

466 Table 7 shows the processing time needed, on average, to compute a feature
 467 vector from a single dynamic texture for each method. In the tests, we computed
 468 the average of 20 executions of feature extraction of a single sample using a 3.60
 469 GHz Intel(R) Core i7, 64GB RAM, and 64-bit Operating System. The proposed
 470 method took 3.07 s and 2.47 s to extract the feature vector from the UCLA and
 471 Dyntex++ database, respectively. The RNN-DT method took the lowest time,
 472 1.35 s on the UCLA database and 1.83 s on the Dyntex++ database. Although
 473 the proposed method obtained the second-lowest time, the results demonstrate
 474 that our method is very competitive compared to the other approaches, taking
 475 a reasonable time to compute the features, and achieving high accuracies.

Table 7: Computational processing time in seconds of the proposed method and other methods to compute the feature vector.

Descriptor	UCLA	Dyntex++
VLBP	3.67	2.60
DPSW	49.02	34.13
DPSWNet	8.97	48.78
CNDT	17.68	16.45
Diffusion	6.68	4.94
RNN-DT	1.35	1.83
Proposed Method	3.07	2.47

476 *5.2. Dynamic Texture Segmentation*

477 Here, we show how our CPNN descriptors (described in Section 3) can be
 478 used for real-time dynamic texture segmentation. Experiments were done with
 479 videos captured from a moving boat on the Guerlédan lake in Brittany. In
 480 order to apply our descriptor for dynamic texture segmentation, we consider
 481 an approach based on overlapping blocks. Figure 7 summarizes the approach
 482 for dynamic texture segmentation: firstly, the video is divided into overlapping
 483 blocks; for each block a feature vector is obtained using our dynamic texture
 484 descriptor; finally the feature vector is labeled using our classification approach.

- 485 • **Overlapping blocks:** the dynamic texture video of $w \times h \times T$ pixels is
 486 divided into overlapping blocks B of size $p \times p \times q$ pixels (as can be seen in
 487 Figure 7(a)). The blocks are evenly sampled using steps of size l between
 488 each block (horizontally, vertically and temporally). The border pixels are
 489 not considered when it is not possible to fit a block.
- 490 • **Feature extraction:** a feature vector is obtained for each block of the
 491 dynamic texture video using the proposed method for dynamic texture
 492 description in Section 3.
- 493 • **Labeling:** in this step, the blocks are labeled from their feature vector.
 494 For this, we use supervised classifiers. In this way, new blocks are pre-
 495 dicted based on a classifier model trained using labeled blocks from other
 496 annotated videos. As we are using overlapping blocks, the pixels of the

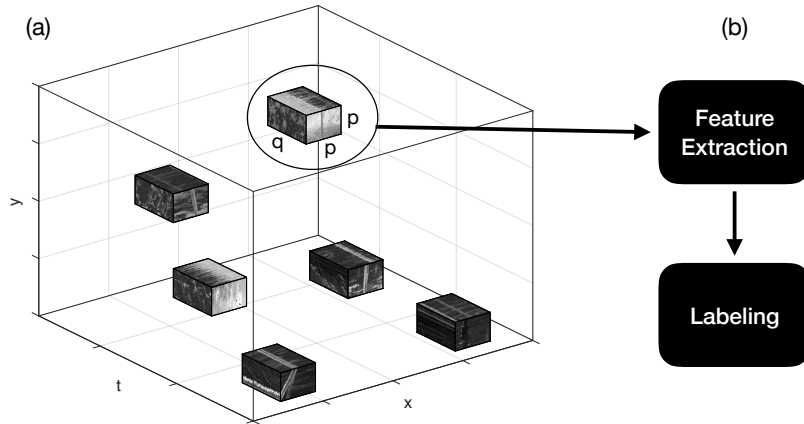


Figure 7: Illustration of the dynamic texture segmentation approach based on block labeling.

497 video belong to several blocks, thus their final label is provided by majority
 498 voting.

499 The goal is to test our dynamic texture descriptor to segment the different
 500 semantic regions of the video: water, sky and land (forest). Figure 8(a) shows
 501 an illustration of the video used in the experiments. The main challenges of
 502 this type of video are the variability of textures due to perspective, the sun
 503 reflections, and of course, the motion of the boat. The video used in the training
 504 step has $288 \times 384 \times 1200$ pixels, while the video used in the testing step has
 505 $288 \times 384 \times 1621$ pixels. The labeled blocks used for the training of the classifier
 506 were obtained randomly from the different regions of the training video. In the
 507 experiments, we tested different sizes of step between blocks ($l = 5, 8, 11$) and a
 508 size of block equal to $p = q = 30$ pixels. We also used two classifiers (Support
 509 Vector Machine (SVM) and 1-Nearest Neighbors (1-NN)) and different numbers
 510 of labeled samples to train the classifiers.

511 In Figure 9 is shown the first frame of the videos segmented into three classes,
 512 with each color representing a class: blue (sky), red (land) and transparent
 513 (water). In this figure, in the first column was used 1000 samples for training
 514 the 1-NN classifier and in the second column 2000 samples. Figure 10 shows
 515 the segmented videos using the SVM classifier. As can be seen, the proposed

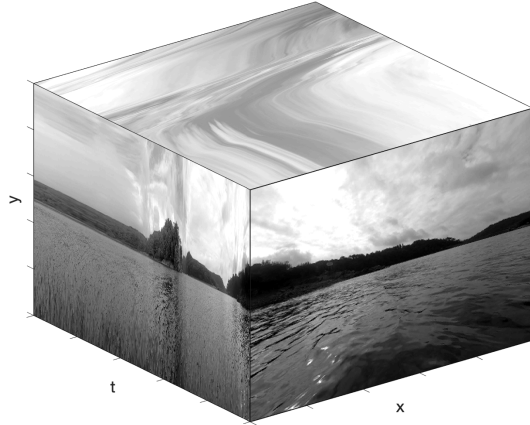


Figure 8: Space \times time cuboid view of one moving boat video used in the experiment of segmentation.

516 scheme can identify well the three different classes in all examples. However,
 517 the method has more difficulties to delimit the border regions. To improve this
 518 point, we intend to investigate new ways of applying or refine the scheme for
 519 segmentation.

520 The main advantages of our method are the computational simplicity and
 521 the fast processing time, which makes the approach promising for real-time seg-
 522 mentation task and active learning. In addition, the proposed approach uses a
 523 small number of samples for training, which is another interesting characteristic
 524 for problems with few data samples.

525 6. Conclusion

526 In this paper, we present a method for dynamic texture recognition called
 527 CPNN, which learns a representation from the graph-based features using ran-
 528 domized neural networks. This is achieved through the adoption of a Complex
 529 Network framework for modeling a video through directed graphs, which are able
 530 to efficiently model the appearance and motion characteristics of the dynamic
 531 texture. From this, graph-based features can be learned by the randomized

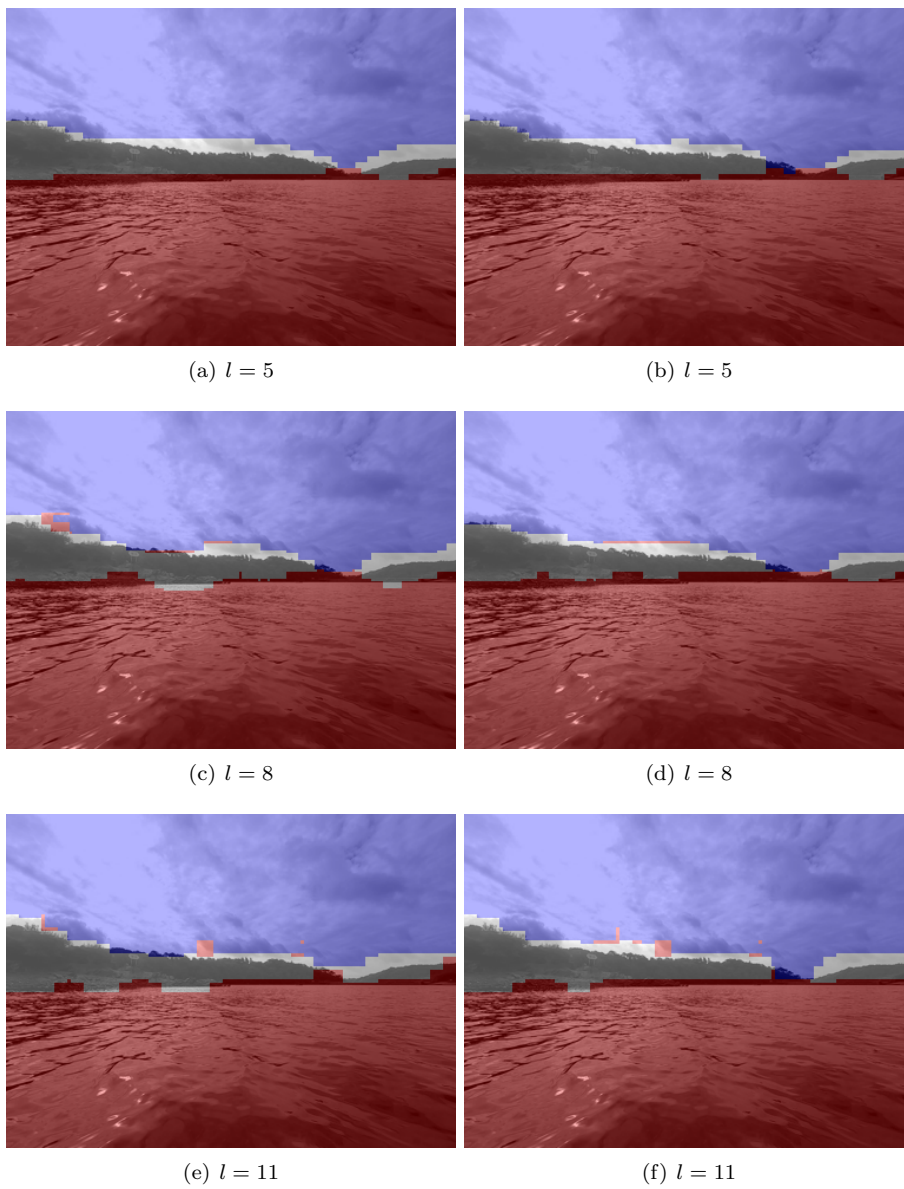
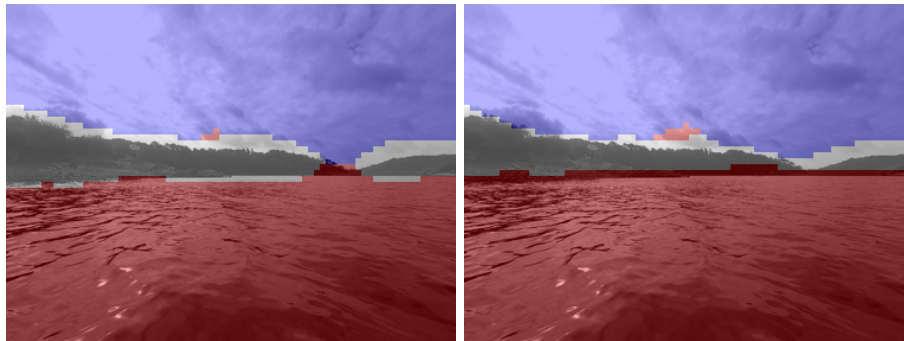
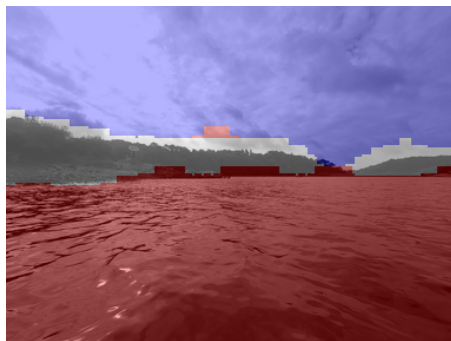


Figure 9: One frame of segmented video using our DT model with the KNN classifier, for different values of step parameter l , and different numbers of training samples (left: 1000, right: 2000).



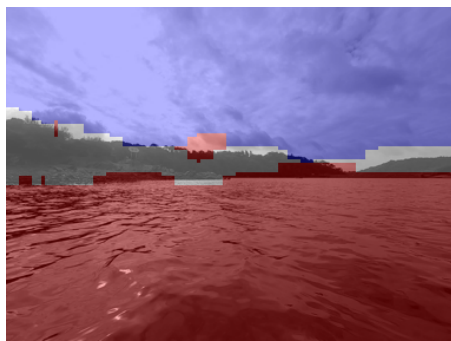
(a) $l = 5$

(b) $l = 5$



(c) $l = 8$

(d) $l = 8$



(e) $l = 11$

(f) $l = 11$

Figure 10: Same as Figure 9, but using the SVM classifier.

532 neural network, which has a simple and fast learning algorithm, producing a
533 representative feature vector through the trained weights of the output layer.
534 Based on the experiments, we adopted in the proposed method six randomized
535 neural networks of 1 hidden layer with 4, 24, and 29 number of hidden units
536 and input layers of size 4 and 10 features.

537 We have tested the CPNN method on two benchmarks for the task of dy-
538 namic texture classification. The results lead to the conclusion that our method
539 provides discriminative dynamic texture descriptors using a simple classifier.
540 Also, experiments of computational processing time demonstrated a competi-
541 tive performance of the proposed method compared to the others. Based on
542 these findings and on some experiments in dynamic texture segmentation, the
543 proposed method can be a valuable tool for real-time applications and could
544 be investigated for active learning purposes. These observations motivate the
545 future investigation of exploring the complex network frameworks and learning
546 methods for dynamic texture analysis. As limitation and future work, new man-
547 ners of creating the training set of the neural network can be explored, such as
548 the use of clustering measures, hierarchical degree and joint-degree of the ver-
549 tices. The interpretation of the physical meaning of the learned features by the
550 proposed model should also be further investigated. Another research issue is to
551 investigate other neural network architectures with more layers (e.g., D-RVFL)
552 to learn the features, although computational time may increase. Furthermore,
553 our method learns the features using a regression model from RNN. However,
554 we believe that a classification model in which each output neuron represents
555 one class can be employed to learn the features and improve the results.

556 **Acknowledgments**

557 Lucas C. Ribas gratefully acknowledges the financial support grant #2016/23763-
558 8 and #2019/03277-0, São Paulo Research Foundation (FAPESP). Jarbas Joaci
559 de Mesquita Sá Junior thanks CNPq (National Council for Scientific and Tech-
560 nological Development, Brazil) (Grant: 302183/2017-5) for the financial sup-

561 port of this work. O. M. Bruno acknowledges support from CNPq (Grant
562 #307897/2018-4) and FAPESP (grant #2014/08026-1, 2018/22214-6 and 2016/18809-
563 9). The authors wish to thank Thomas Simon and Clément Yver for allowing
564 them to use their videos from Guerlédan’s lake.

565 **References**

- 566 [1] G. Doretto, A. Chiuso, Y. Wu, S. Soatto, Dynamic textures, *International*
567 *Journal of Computer Vision* 51 (2) (2003) 91–109.
- 568 [2] Y. Xu, Y. Quan, H. Ling, H. Ji, Dynamic texture classification using dy-
569 namic fractal analysis, in: *2011 International Conference on Computer*
570 *Vision*, 2011, pp. 1219–1226.
- 571 [3] L. C. Ribas, W. N. Gonçalves, O. M. Bruno, Dynamic texture analysis with
572 diffusion in networks, *Digital Signal Processing* 92 (2019) 109–126.
- 573 [4] X. Zhao, Y. Lin, J. Heikkil, Dynamic texture recognition using volume local
574 binary count patterns with an application to 2D face spoofing detection,
575 *IEEE Transactions on Multimedia* 20 (3) (2018) 552–566.
- 576 [5] K. G. Derpanis, R. P. Wildes, Classification of traffic video based on a
577 spatiotemporal orientation analysis, in: *IEEE Workshop on Applications*
578 *of Computer Vision (WACV)*, 2011, pp. 606–613.
- 579 [6] W. Li, V. Mahadevan, N. Vasconcelos, Anomaly detection and localization
580 in crowded scenes, *IEEE Transactions on Pattern Analysis and Machine*
581 *Intelligence* 36 (1) (2014) 18–32.
- 582 [7] K. Dimitropoulos, P. Barmoutis, N. Grammalidis, Spatio-temporal flame
583 modeling and dynamic texture analysis for automatic video-based fire de-
584 tection, *IEEE Transactions on Circuits and Systems for Video Technology*
585 25 (2) (2015) 339–351.

- 586 [8] R. P. Wildes, J. R. Bergen, Qualitative spatiotemporal analysis using an
587 oriented energy representation, in: European Conference on Computer Vi-
588 sion, Springer, 2000, pp. 768–784.
- 589 [9] K. G. Derpanis, R. P. Wildes, Dynamic texture recognition based on distri-
590 butions of spacetime oriented structure, in: 2010 IEEE Computer Society
591 Conference on Computer Vision and Pattern Recognition, IEEE, 2010, pp.
592 191–198.
- 593 [10] S. R. Arashloo, J. Kittler, Dynamic Texture Recognition Using Multiscale
594 Binarized Statistical Image Features, *IEEE Transactions on Multimedia*
595 16 (8) (2014) 2099–2109.
- 596 [11] C. Feichtenhofer, A. Pinz, R. P. Wildes, Bags of spacetime energies for
597 dynamic scene recognition, in: Proceedings of the IEEE Conference on
598 Computer Vision and Pattern Recognition, 2014, pp. 2681–2688.
- 599 [12] A. R. Rivera, O. Chae, Spatiotemporal directional number transitional
600 graph for dynamic texture recognition, *IEEE Transactions on Pattern Anal-
601 ysis and Machine Intelligence* 37 (10) (2015) 2146–2152.
- 602 [13] T. T. Nguyen, T. P. Nguyen, F. Bouchara, Prominent local representa-
603 tion for dynamic textures based on high-order Gaussian-gradients, *IEEE
604 Transactions on Multimedia*.
- 605 [14] X. Zhao, Y. Lin, L. Liu, J. Heikkilä, W. Zheng, Dynamic texture classifica-
606 tion using unsupervised 3d filter learning and local binary encoding, *IEEE
607 Transactions on Multimedia* 21 (7) (2019) 1694–1708.
- 608 [15] T. T. Nguyen, T. P. Nguyen, F. Bouchara, A novel filtering kernel based
609 on difference of derivative Gaussians with applications to dynamic texture
610 representation, *Signal Processing: Image Communication* 98 (2021) 116394.
- 611 [16] R. Polana, R. Nelson, Temporal texture and activity recognition, in:
612 M. Shah, R. Jain (Eds.), *Motion-Based Recognition*, Vol. 9 of *Compu-
613 tational Imaging and Vision*, Springer Netherlands, 1997, pp. 87–124.

- 614 [17] C.-H. Peh, L.-F. Cheong, Synergizing spatial and temporal texture, IEEE
615 Transactions on Image Processing 11 (10) (2002) 1179–1191.
- 616 [18] R. Péteri, D. Chetverikov, Dynamic texture recognition using normal
617 flow and texture regularity, in: Pattern Recognition and Image Analysis,
618 Springer, 2005, pp. 223–230.
- 619 [19] S. Fazekas, D. Chetverikov, Dynamic texture recognition using optical flow
620 features and temporal periodicity, in: International Workshop on Content-
621 Based Multimedia Indexing (CBMI), 2007, pp. 25–32.
- 622 [20] T. T. Nguyen, T. P. Nguyen, F. Bouchara, X. S. Nguyen, Directional beams
623 of dense trajectories for dynamic texture recognition, in: International Con-
624 ference on Advanced Concepts for Intelligent Vision Systems, Springer,
625 2018, pp. 74–86.
- 626 [21] L. N. Couto, C. A. Barcelos, Singular patterns in optical flows as dynamic
627 texture descriptors, in: Iberoamerican Congress on Pattern Recognition,
628 Springer, 2018, pp. 351–358.
- 629 [22] G. Zhao, M. Pietikainen, Dynamic texture recognition using local binary
630 patterns with an application to facial expressions, IEEE Transactions on
631 Pattern Analysis and Machine Intelligence 29 (6) (2007) 915–928.
- 632 [23] G. Zhao, M. Pietikäinen, Dynamic texture recognition using volume local
633 binary patterns, in: Dynamical Vision, Springer Berlin Heidelberg, 2006,
634 pp. 165–177.
- 635 [24] R. D. Tiwari, V. Tyagi, Dynamic texture recognition using multiresolution
636 edge-weighted local structure pattern, Computers & Electrical Engineering
637 62 (2017) 485–498.
- 638 [25] D. Tiwari, V. Tyagi, A novel scheme based on local binary pattern for
639 dynamic texture recognition, Computer Vision and Image Understanding
640 150 (2016) 58–65.

- 641 [26] G. Zhao, T. Ahonen, J. Matas, M. Pietikainen, Rotation-invariant image
642 and video description with local binary pattern features, *IEEE transactions*
643 *on image processing* 21 (4) (2011) 1465–1477.
- 644 [27] J. Luo, Z. Tang, H. Zhang, Y. Fan, Y. Xie, Ltgh: A dynamic texture feature
645 for working condition recognition in the froth flotation, *IEEE Transactions*
646 *on Instrumentation and Measurement* 70 (2021) 1–10.
- 647 [28] T. T. Nguyen, T. P. Nguyen, F. Bouchara, X. S. Nguyen, Momental di-
648 rectional patterns for dynamic texture recognition, *Computer Vision and*
649 *Image Understanding* 194 (2020) 102882.
- 650 [29] B. Raman, D. Sadhya, et al., Dynamic texture recognition using local tetra
651 pattern-three orthogonal planes (LTrP-TOP), *The Visual Computer* 36 (3)
652 (2020) 579–592.
- 653 [30] D. Tiwari, V. Tyagi, Dynamic texture recognition based on completed vol-
654 ume local binary pattern, *Multidimensional Systems and Signal Processing*
655 27 (2) (2016) 563–575.
- 656 [31] X. Qi, C.-G. Li, G. Zhao, X. Hong, M. Pietikäinen, Dynamic texture and
657 scene classification by transferring deep image features, *Neurocomputing*
658 171 (2016) 1230–1241.
- 659 [32] S. R. Arashloo, M. C. Amirani, A. Noroozi, Dynamic texture represen-
660 tation using a deep multi-scale convolutional network, *Journal of Visual*
661 *Communication and Image Representation* 43 (2017) 89–97.
- 662 [33] V. Andrearczyk, P. F. Whelan, Convolutional neural network on three or-
663 thogonal planes for dynamic texture classification, *Pattern Recognition* 76
664 (2018) 36–49.
- 665 [34] X. Zhao, Y. Lin, L. Liu, Dynamic texture recognition using 3d random fea-
666 tures, in: *ICASSP 2019-2019 IEEE International Conference on Acoustics,*
667 *Speech and Signal Processing (ICASSP), IEEE, 2019, pp. 2102–2106.*

- 668 [35] J. J. M. Sá Junior, L. C. Ribas, O. M. Bruno, Randomized neural network
669 based signature for dynamic texture classification, *Expert Systems with*
670 *Applications* 135 (2019) 194–200.
- 671 [36] A. Ravichandran, R. Chaudhry, R. Vidal, View-invariant dynamic texture
672 recognition using a bag of dynamical systems, in: *2009 IEEE Conference*
673 *on Computer Vision and Pattern Recognition*, 2009, pp. 1651–1657.
- 674 [37] A. Ravichandran, R. Chaudhry, R. Vidal, Categorizing dynamic textures
675 using a bag of dynamical systems, *IEEE Transactions on Pattern Analysis*
676 *and Machine Intelligence* 35 (2) (2012) 342–353.
- 677 [38] Y. Wang, S. Hu, Chaotic features for dynamic textures recognition, *Soft*
678 *Computing* 20 (5) (2016) 1977–1989.
- 679 [39] W. N. Gonçalves, O. M. Bruno, Dynamic texture analysis and segmenta-
680 tion using deterministic partially self-avoiding walks, *Expert Systems with*
681 *Applications* 40 (11) (2013) 4283 – 4300.
- 682 [40] W. N. Gonçalves, B. B. Machado, O. M. Bruno, A complex network ap-
683 proach for dynamic texture recognition, *Neurocomputing* 153 (2015) 211 –
684 220.
- 685 [41] L. C. Ribas, J. J. M. Sá Junior, L. F. S. Scabini, O. M. Bruno, Fusion
686 of complex networks and randomized neural networks for texture analysis,
687 *Pattern Recognition* 103 (2020) 107189.
- 688 [42] L. C. Ribas, O. M. Bruno, Dynamic texture analysis using networks gen-
689 erated by deterministic partially self-avoiding walks, *Physica A: Statistical*
690 *Mechanics and its Applications* 541 (2020) 122105.
- 691 [43] A.-L. Barabási, R. Albert, Emergence of scaling in random networks, *sci-*
692 *ence* 286 (5439) (1999) 509–512.
- 693 [44] M. Girvan, M. E. Newman, Community structure in social and biological
694 networks, *Proceedings of the national academy of sciences* 99 (12) (2002)
695 7821–7826.

- 696 [45] D. J. Watts, S. H. Strogatz, Collective dynamics of small-world networks,
697 nature 393 (6684) (1998) 440–442.
- 698 [46] L. F. Scabini, R. H. Condori, W. N. Gonçalves, O. M. Bruno, Multilayer
699 complex network descriptors for color–texture characterization, Informa-
700 tion Sciences 491 (2019) 30–47.
- 701 [47] W. F. Schmidt, M. A. Kraaijveld, R. P. W. Duin, Feedforward neural net-
702 works with random weights, in: Proceedings., 11th IAPR International
703 Conference on Pattern Recognition. Vol.II. Conference B: Pattern Recog-
704 nition Methodology and Systems, 1992, pp. 1–4.
- 705 [48] Y.-H. Pao, Y. Takefuji, Functional-link net computing: theory, system ar-
706 chitecture, and functionalities, Computer 25 (5) (1992) 76–79.
- 707 [49] Y.-H. Pao, G.-H. Park, D. J. Sobajic, Learning and generalization charac-
708 teristics of the random vector functional-link net, Neurocomputing 6 (2)
709 (1994) 163–180.
- 710 [50] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory
711 and applications, Neurocomputing 70 (1) (2006) 489–501.
- 712 [51] E. H. Moore, On the reciprocal of the general algebraic matrix, Bulletin of
713 the American Mathematical Society 26 (1920) 394–395.
- 714 [52] R. Penrose, A generalized inverse for matrices, Mathematical Proceedings
715 of the Cambridge Philosophical Society 51 (3) (1955) 406–413.
- 716 [53] A. N. Tikhonov, On the solution of ill-posed problems and the method of
717 regularization, Dokl. Akad. Nauk USSR 151 (3) (1963) 501–504.
- 718 [54] D. Calvetti, S. Morigi, L. Reichel, F. Sgallari, Tikhonov regularization and
719 the L-curve for large discrete ill-posed problems, Journal of Computational
720 and Applied Mathematics 123 (1) (2000) 423 – 446.
- 721 [55] D. H. Lehmer, Mathematical methods in large scale computing units, An-
722 nals Comp. Laboratory Harvard University 26 (1951) 141–146.

- 723 [56] S. K. Park, K. W. Miller, Random number generators: good ones are hard
724 to find, *Communications of the ACM* 31 (10) (1988) 1192–1201.
- 725 [57] J. J. M. Sá Junior, A. R. Backes, ELM based signature for texture classi-
726 fication, *Pattern Recognition* 51 (2016) 395–401.
- 727 [58] B. Ghanem, N. Ahuja, Maximum margin distance learning for dynamic
728 texture recognition, in: *Proceedings of the 11th European Conference on*
729 *Computer Vision: Part II, ECCV’10*, Springer-Verlag, Berlin, Heidelberg,
730 2010, pp. 223–236.
- 731 [59] R. Péteri, S. Fazekas, M. J. Huiskes, DynTex: A comprehensive database of
732 dynamic textures, *Pattern Recognition Letters* 31 (12) (2010) 1627–1632.
- 733 [60] P. Saisan, G. Doretto, Y. N. Wu, S. Soatto, Dynamic texture recogni-
734 tion, in: *Proceedings of the 2001 IEEE Computer Society Conference on*
735 *Computer Vision and Pattern Recognition. CVPR 2001, Vol. 2, 2001*, pp.
736 II–58–II–63 vol.2.
- 737 [61] G. Holmes, A. Donkin, I. H. Witten, Weka: A machine learning work-
738 bench, in: *Proceedings of ANZIS’94-Australian New Zealand Intelligent*
739 *Information Systems Conference, IEEE, 1994*, pp. 357–361.
- 740 [62] F. Wilcoxon, Individual comparisons by ranking methods, in: *Break-*
741 *throughs in statistics*, Springer, 1992, pp. 196–202.
- 742 [63] A. B. Chan, N. Vasconcelos, Classifying video with kernel dynamic tex-
743 tures, in: *IEEE Conference on Computer Vision and Pattern Recognition*
744 *(CVPR), 2007*, pp. 1–6.
- 745 [64] Y. Xu, S. Huang, H. Ji, C. Fermüller, Scale-space texture description on
746 SIFT-like textons, *Computer Vision and Image Understanding* 116 (9)
747 (2012) 999–1013.
- 748 [65] Y. Wang, S. Hu, Exploiting high level feature for dynamic textures recog-
749 nition, *Neurocomputing* 154 (2015) 217–224.

- 750 [66] H. Ji, X. Yang, H. Ling, Y. Xu, Wavelet domain multifractal analysis for
751 static and dynamic texture classification, *IEEE Transactions on Image Pro-*
752 *cessing* 22 (1) (2013) 286–299.
- 753 [67] L. Breiman, Random forests, *Machine learning* 45 (1) (2001) 5–32.
- 754 [68] R. Katuwal, P. N. Suganthan, Stacked autoencoder based deep random
755 vector functional link neural network for classification, *Applied Soft Com-*
756 *puting* 85 (2019) 105854.
- 757 [69] R. A. Fisher, The use of multiple measurements in taxonomic problems,
758 *Annals of Eugenics* 7 (7) (1936) 179–188.
- 759 [70] D. R. Amancio, C. H. Comin, D. Casanova, G. Travieso, O. M. Bruno, F. A.
760 Rodrigues, L. da Fontoura Costa, A systematic comparison of supervised
761 classifiers, *PloS one* 9 (4) (2014) e94137.