Convolutional Neural Network on Three Orthogonal Planes for Dynamic Texture Classification

Vincent Andrearczyk, Paul F. Whelan

PII: S0031-3203(17)30437-5
DOI: 10.1016/j.patcog.2017.10.030
Reference: PR 6342

To appear in: *Pattern Recognition*

Received date: 4 April 2017
Revised date: 30 September 2017
Accepted date: 23 October 2017

Highlights

- A new CNN framework is introduced to analyze DTs on three orthogonal planes.
- Multiple texture specific CNNs are developed.
- An analysis of the contribution of each plane is conducted as well as the domain transferability.
- Experiments on various DT classification datasets show the superiority of our approach over existing ones.

Convolutional Neural Network on Three Orthogonal Planes for Dynamic Texture Classification

Vincent Andrearczyk[a,*], Paul F. Whelan[a]

[a] *Vision Systems Group, School of Electronic Engineering, Dublin City University, Glasnevin, Dublin 9, Ireland*

Abstract

Dynamic Textures (DTs) are sequences of images of moving scenes that exhibit certain stationarity properties in time such as smoke, vegetation and fire. The analysis of DT is important for recognition, segmentation, synthesis or retrieval for a range of applications including surveillance, medical imaging and remote sensing. Convolutional Neural Networks (CNNs) have recently proven to be well suited for texture analysis with a design similar to filter banks. We develop a new DT analysis method based on a CNN method applied on three orthogonal planes. We train CNNs on spatial frames and temporal slices extracted from the DT sequences and combine their outputs to obtain a competitive DT classifier trained end-to-end. Our results on a wide range of commonly used DT classification benchmark datasets prove the robustness of our approach. Significant improvement of the state of the art is shown on the larger datasets.

Keywords: Dynamic texture, image recognition, Convolutional Neural Network, filter banks, spatiotemporal analysis

1. Introduction

Dynamic Texture (DT) is an extension of texture in the temporal domain, introducing temporal variations such as motion and deformation. Doretto et al. [1] describe a DT as a sequence of images of moving scenes that exhibits certain

*Corresponding author
Email address: `vincent.andrearczyk3@mail.dcu.ie` (Vincent Andrearczyk)

stationary properties in time. Examples of natural DTs include smoke, clouds, trees and waves. The analysis of DTs, including classification, segmentation, synthesis and indexing for retrieval is essential for a large range of applications such as surveillance, medical image analysis and remote sensing. New methods, mainly derived from static texture approaches are required to incorporate the analysis of temporal changes to the spatial analysis. The major difficulties in DT analysis are due to the wide range of appearances and dynamics of DTs and to the simultaneous analysis and combination of spatial and temporal properties.

Deep learning has been shown to produce state of the art results on various computer vision tasks. In particular, Convolutional Neural Networks (CNNs) are very powerful for image segmentation and classification and have shown great adaptability to texture analysis by discarding the overall shape analysis [2, 3, 4]. This paper presents a new CNN framework for the analysis of DTs and an application to DT classification. The idea of our method is to extend our previous work on Texture CNN (T-CNN) [2] to the analysis of DTs by analyzing the sequences on three orthogonal planes. We train neural networks to recognize DT sequences based on the spatial distribution of pixels and on their evolution and dynamics over time. A sequence can be alternatively seen as a stack of frames (xy) along the temporal axis t, a stack of xt slices along the y axis and a stack of yt slices along the x axis. This is partly inspired by the extension of Local Binary Pattern on Three Orthogonal Planes (LBP-TOP) by Zhao and Pietikäinen [5]. While 3D filters [6] could be implemented to analyze the DT sequences, using 2D filters on three orthogonal planes significantly reduces the complexity and has been shown to extract sufficient complementary spatiotemporal information for robust DT classification [5]. Besides, the analysis of 2D planes enables the networks to be pre-trained on large image databases and fine-tuned with multiple slices extracted along a given spatial or temporal axis per sequence. An overview of the pipeline of the proposed method is illustrated in Figure 1 and explained in more detail in Section 3. We first extract slices from the DT sequences to train an independent T-CNN on each plane. We then sum the outputs of all the slices on all the planes to obtain the class with maximum

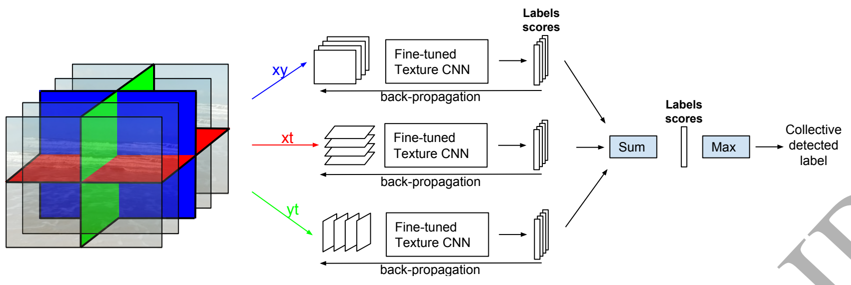


Figure 1: Overview of the proposed method for the classification of a DT sequence based on T-CNNs on three orthogonal planes in an ensemble model approach. The CNNs separately classify frames extracted from an input DT sequence on three planes. The outputs of the last fully-connected layers are summed and the highest score gives the collective classification decision.

prediction score during the testing phase in an ensemble model fashion [7].

The main contributions of this paper are as follows:

(a) We introduce a new framework to analyze DTs on three orthogonal planes.

(b) We develop deeper CNNs for textures as well as networks adapted to small images based on the original T-CNN in [2].

(c) We experiment on three DT databases with seven benchmarks and present comparative results with the literature.

(d) We conduct an analysis of the contribution and complementarity of each plane in the learning and classification of DT sequences as well as the domain transferability of trained parameters.

The rest of the paper is organized as follows. In Section 2, we describe the related work. We present our CNN method for DT analysis in Section 3. In Section 4, we test our method on three widely used DT databases derived in seven benchmarks with considerable differences including the number of classes, number and size of images and the experimental protocol. We compare our results to the state of the art and show a significant improvement.

2. Related work

2.1. Deep learning for texture images

Basu et. al [8] argue that the dimensionality of texture datasets is too large for deep networks to shatter them without explicitly extracting hand-crafted features beforehand. However, [2, 3, 4] show that the domain transferability of deep networks enables pre-trained networks to perform very well on texture classification and segmentation tasks as a simple feature extractor [3] or in an end-to-end training and testing scheme [2, 4]. As explained in [2], a CNN approach is well suited to texture analysis as it can densely pool texture features. Indeed, features learned by the first layer are similar to Gabor filters with mostly edge-like kernels and deeper features can be used as a more complex filter bank approach, widely used in texture analysis [9].

Texture descriptors are extracted in a similar manner as a CNN approach in [10]. The trainable filters are replaced by hand-crafted scaled and rotated wavelets to perform a scattering transform. The obtained descriptors are invariant to transformations such as translation, scaling and rotation and robust to small deformations. Rotation invariance is developed in a shallow CNN for texture classification in [11] by tying the weights of multiple rotated versions of filters. Deep neural networks were applied to texture analysis in [3] as a feature extractor, in which the output of the last convolution layer is used in a framework including PCA and SVM classifier. They largely improve the state of the art on texture classification benchmarks with very deep networks. A bilinear CNN model is developed in [12] for fine-grained recognition, combining two CNNs to extract and classify translationally invariant local pairwise features in a neural network framework. This method also performs well on texture datasets as shown in [4] and can generalize several widely used orderless texture descriptors. A T-CNN is developed in [2] which includes an energy layer that extracts the dense response to intermediate features in the network, obtaining significant results on texture classification tasks with reduced complexity compared to classic CNNs. The complementarity of texture and shape analysis is shown using convolutional networks for feature extraction in [3] and in an end-to-end CNN training scheme in [2]. A fully convolutional approach is developed for the segmentation of texture regions in [13].

2.2. Classic Dynamic Texture approaches

Difficulties in DT analysis arise from the large range of phenomena resulting from natural scenes and recording methods including scale, illumination and rotation variation as well as static and moving cameras. Most classic DT analysis approaches that attempt to overcome all or some of these difficulties can be classified into four categories, namely statistical, motion based, model based and spatiotemporal filtering.

Statistical approaches [5, 14, 15, 16, 17] mainly adapt standard spatial texture methods such as Gray Level Co-occurrence matrix (GLCM) [18] and Local Binary Pattern (LBP) [19] to extend their analysis to the spatiotemporal domain. LBP has been widely investigated and developed in both static and dynamic texture analysis due to its computation simplicity, invariances and good performance. Zhao and Pietikäinen [5] extend the LBP to DTs by extracting patterns on Three Orthogonal Planes (LBP-TOP): xy which is the classic spatial texture LBP as well as xt and yt which consider temporal variations of the pixel intensities. Several approaches were derived from the LBP-TOP. The Local Phase Quantization on Three Orthogonal Planes (LPQ-TOP) [16], based on quantizing the phase information of the local Fourier transform which, is particularly robust to blurred images. The LBP-TOP is combined to a Weber local descriptor in [17] to describe the spatial appearance of DTs for segmentation, combined to a motion analysis for the temporal information. More recently, Tiwari and Tyagi [14] incorporate a noise resistance feature to the LBP-TOP with a Weber's law threshold in the creation of the patterns. The same authors develop another local threshold in [15] and extract patterns at multiple scales in combination with local and global image statistics and weighted by gradient magnitude. The main drawbacks of the LBP-based approaches are the poor robustness to camera motion and the limited amount of information that the hand-crafted patterns can extract.

Motion based methods exploit the statistical properties of the motion extracted between consecutive frames of the DT sequences. Several motion extraction methods are used for the analysis of DT including Local Motion Pattern (LMP) [20], Complete Flow (CF) [17, 21] and more commonly Normal Flow (NF) [22, 23, 24]. Statistical features are then extracted from the motion to describe its spatial distribution such as GLCM [21, 22], Fourier spectrum and difference statistics [22], histograms [17, 20] and other statistics calculated on the motion field itself [23] and on its derivatives [24].

Model based methods [1, 25, 26, 27] aim at estimating the parameters of a Linear Dynamical System (LDS) using a system identification theory in order to capture the spatial appearance and dynamics of a scene. Originally designed for the synthesis problem, the estimated parameters can be used for classification [1]. However, the model based approach raises several difficulties such as the distance between models lying in a non-linear space of LDSs with a complicated manifold as well as a poor invariance to rotation, scale and illumination and assumes a single, well segmented DT. In [27] Fujita and Nayar bypass the non-linear space, segmentation and presence of multiple DTs problems by using impulse response of state variables learned during the system identification which capture the fundamental dynamical properties of DTs. Ravichandran et al. [25] overcome the view-invariance problem using a Bag-of-dynamical Systems (BoS) similar to a Bag-of-Features (BoF) with LDSs as feature descriptors. Afsari et. al [26] define a new distance between models in the non-linear space based on the notion of aligning LDSs. It enables them to compute the mean of a set of LDSs for an efficient classification, particularly adapted to large-scale problems. A different model based approach is proposed in [28], in which 2D and 3D fractal analyses are performed on the intensity, first and second order measurements and normal flow.

Filtering approaches for texture analysis were extended to the spatiotemporal domain for the analysis of DTs [29, 30, 31, 32]. Derpanis and Wildes [29] use spacetime oriented filters to extract interesting features which describe intrinsic properties of the DTs such as unstructured, static motion and transparency. A spatiotemporal directional number transitional graph was proposed in [33]. Graph-based descriptors are computed from the response to a set of spatiotemporal oriented local filters to represent the signature of DTs. In [30], Qiao and Wang used 3D Dual Tree Complex Wavelet combining the spatial orientation and motion selectiveness of wavelets. Other approaches use Wavelet Domain Multifractal Analysis (WDMA) [31] and Spatiotemporal Oriented Energy (SOE) [32].

Finally, many methods combine two or more approaches for their complementarity [17, 20, 21, 23, 28, 34]. In [35], Ghanem and Ahuja combine LBP, Pyramid of Histogram of Oriented Gradients and an LDS model approach to jointly analyze the spatial texture, the spatial layout and the dynamics of DT sequences. More recently, Yang et. al [34] use an ensemble SVM to combine static features such as LBP Gabor filters with temporal model based features.

The described shallow approaches lack of informativity and abstraction and require features specifically hand-crafted for a given problem which may not generalize to new data.

2.3. Deep learning for videos and Dynamic Textures

Following the breakthrough of deep learning in image recognition, several attempts have been made to apply CNNs to videos and/or DT analysis.

End-to-end convolutional networks with 3D filters are developed in [6], capturing spatial and temporal information for human action recognition. In [36] Tran et. al generalize the 3D convolution and pooling to cope with large video datasets. In [37], several CNN architectures are evaluated on a video classification task including classic single-frame CNN, two-stream single frame networks with various fusion approaches and 3D convolution filters. While the analysis of a few sequences benefit from a combination with motion analysis, the latter increases the sensitivity to camera motion.

Simonyan and Zisserman [38] develop a two-stream CNN including spatial and temporal analysis for action recognition. In their architecture, a spatial network is trained to recognize still video frames while a motion network is trained on dense optical flow. The two networks are trained separately and their softmax scores are combined by late fusion. Donahue et al. [39] develop a long-term recurrent CNN to learn compositional representations in space and time. While this method is appropriate for activity recognition, image captioning, and video description, it is not as relevant for DT analysis in which one is more interested in probabilities on the variation of pixel intensities (in time and space) than in the sequential detection of particular events. Shao et al. [40] use the semantic selectiveness of spatial convolutional filters to discard the background followed by a combination of spatial and temporal convolutions in a CNN architecture for crowd video understanding. In [41], the same authors use a fully convolutional network on spatial and temporal slices and on the motion extracted on these slices with various fusion strategies in order to create a per-pixel collectiveness map from crowd videos. While these approaches share some similarities with our proposed method, they are designed for the analysis of crowded scenes with assumptions (e.g. background) and methods (e.g. collectiveness map) which do not apply to DT recognition.

While the deep learning approaches for video analysis introduced so far were mainly developed for human action recognition, several recent neural network approaches have focused on classifying DTs. Culibrk and Sebe [42] use temporal gradients and Group Of Pictures (GOP) as inputs to CNNs to classify DTs. The authors dilate and warp the 50×50 images to the AlexNet input size 227×227, resulting in a waste of computation and a network architecture which is not adapted to the input images. Similar to the spatial approach in [3], Qi et. al [43] use a pre-trained CNN to extract features from the sequences. Mid-level features are extracted from each frame of the sequence to compute and classify the first and second order statistics. Recently, a CNN-like architecture was extended to DT in [44] using PCA to learn filter banks on three orthogonal planes. DT features at multiple scales are pooled via histogram count of the quantized responses and classified with linear SVM. Although it shares the idea of pooling filter banks in convolutional networks with our method, the network architecture and the training, descriptors pooling and classification strategies are entirely different. The results reported in [44] suggest that this method is better suited to small datasets with limited variation (UCLA [1]) than larger and more complex datasets (DynTex [45]). Finally, in [46], a variant of stacked autoencoder is trained to learn video dynamics which can be more complex than those based on LDS estimation. The deep encoded dynamics can be used similarly to classic model based methods to synthesize and recognize DTs and segment motions.

3. Methods

This section describes our CNN method for DT analysis and presents a new network adapted to small input image sizes as well as deep architectures based on [2].

3.1. Texture CNN

The Texture-CNN (T-CNN) approach[1] was introduced in our previous work [2] and applied to a tissue classification task in [47]. T-CNN is a network of reduced complexity adapted to the analysis of texture images by making use of dense features extracted at intermediate layers of the CNN and discarding the overall shape analysis of a classic convolutional network by introducing an energy layer. The energy layer pools the average response of all the feature maps into a vector of size equal to the number of input feature maps.
The vector output $\mathbf{e} \in \mathbb{R}^K$ of an energy layer is calculated as:

$$\mathbf{e}[k] = \frac{1}{N \times M} \sum_{i=0}^{N} \sum_{j=0}^{M} x_{i,j}^k, \tag{1}$$

where $x_{i,j}^k$ is the value of the k^{th} input feature map at spatial location (i,j). k enumerates the K input feature maps and N, M are respectively the number of

[1]An implementation of the T-CNN-3 to train and test on the kth-tips-2b texture database is available here: `https://github.com/v-andrearczyk/caffe-TCNN`

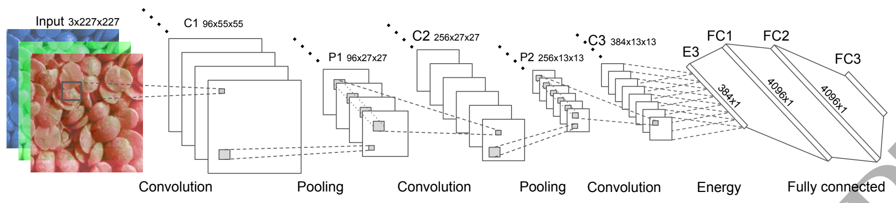


Figure 2: Texture CNN architecture using three convolution layers (T-CNN-3) [2].

rows and columns of the input feature maps. For fixed square input sizes, this layer can be implemented with an average pooling layer of kernel size equal to the input feature map size.

This approach achieves better results than a classic CNN (AlexNet), using the same architecture for the first and final layers and with a number of trainable parameters nearly three times smaller. As shown in [2, 47], the T-CNN based on AlexNet obtains the best results on various texture classification datasets with three convolution layers (T-CNN-3). This architecture is motivated by the fact that complex and large structures and shapes emerge in deep layers rather than simple texture descriptors in intermediate layers as demonstrated in [48]. The architecture of the T-CNN-3 used in this paper for the DynTex database is depicted in Figure 2. Maximum pooling is used to reduce the size of the data throughout the network, learn small transformation invariances and increase the receptive field of the neurons. The activation (non-linearity) of the neurons is performed by Rectified Linear Units (ReLUs) and is normalized by Local Response Normalization (LRN). Dropout is used with a 0.5 probability. More details on the architecture are provided in Table 1 and in [2]. Another network based on the same textural approach is developed to enable the analysis of smaller input sizes for the Dyntex++ and UCLA datasets as described below. In this paper, we experiment on a deeper T-CNN architecture based on GoogleNet [49]. The same idea of energy layer is used after the inception 4d block as detailed in Table 2.

3.2. Small Texture CNN (T-CNN50)

A CNN is used in [42] to analyze DT sequences of the Dyntex++ database. The authors dilate and warp the input images in order to match the input size of AlexNet (227×227). This approach is not optimal both in terms of complexity of the network and capability to train and learn from the small images.

Instead, we have designed a new CNN architecture based on T-CNN described previously to analyze small input images such as those of the Dyntex++ sequences (50×50) and UCLA sequences (48×48). While the input size of the T-CNN does not need to be fixed due to the energy layer, it is necessary to develop another architecture for these small input sizes for the following reasons. We want to keep the receptive field of the neurons small enough to tile the input image in such a way that the energy layer will behave similarly to a filter bank that spans the input image. If we kept the architecture of the T-CNN, the effective receptive field of the neurons of the third convolution layer would be larger than the input image itself. The architecture of this new network (that we refer to as *T-CNN50* in this paper) is detailed in Table 1. Similarly to the analysis of larger images, we develop a deeper network based on GoogleNet and T-CNN by reducing the receptive fields while maintaining the depth.

An important aspect of CNNs is the domain transferability i.e. pre-training a network on a very large database and fine-tuning on a smaller target dataset shows that the pre-trained features generalize well to new types of data in different domains [50] (see Section 3.4). Also, one can generally transfer some or all the kernels learned by any network to another network if the kernel sizes are equal, including a CNN with smaller input size. However, the kernel sizes of the T-CNN and the T-CNN50 being different, we cannot initialize the T-CNN50 using the kernels learned by the T-CNN on ImageNet. Therefore, we pre-train the T-CNN50 (based on AlexNet and GoogleNet) on ImageNet [51] with the images resized to 50×50.

Table 1: Architectures of the T-CNN and T-CNN50 based on AlexNet, where c is the number of color channels and N is the number of classes. ReLU and LRN is used similarly to [49].

Layer type	**Output sizes**		**kernel, pad, stride**		**trainable param.**	
	T-CNN	T-CNN50	T-CNN	T-CNN50	T-CNN	T-CNN50
crop	$c \times 227 \times 227$	$c \times 48 \times 48$	-	-	0	0
Conv (C1)	$96 \times 55 \times 55$	$96 \times 48 \times 48$	11, 0, 4	5, 2, 1	$c \times 11616 + 96$	$c \times 2400 + 96$
ReLU	$96 \times 55 \times 55$	$96 \times 48 \times 48$	-	-	0	0
Pool (P1)	$96 \times 27 \times 27$	$96 \times 24 \times 24$	3, 0, 2	2, 0, 2	0	0
LRN	$96 \times 27 \times 27$	$96 \times 24 \times 24$	-	-	0	0
Conv (C2)	$256 \times 27 \times 27$	$256 \times 24 \times 24$	5, 2, 1	3, 1, 1	$614,656$	$221,440$
ReLU	$256 \times 27 \times 27$	$256 \times 24 \times 24$	-	-	0	0
Pool (P2)	$256 \times 13 \times 13$	$256 \times 12 \times 12$	3, 0, 2	2, 0, 2	0	0
LRN	$256 \times 13 \times 13$	$256 \times 12 \times 12$	-	-	0	0
Conv (C3)	$384 \times 13 \times 13$	$384 \times 12 \times 12$	3, 1, 1	3, 1, 1	$885,120$	$885,120$
ReLU	$384 \times 13 \times 13$	$384 \times 12 \times 12$	-	-	0	0
Energy	384	384	-	-	0	0
Fully-con. (FC1)	4096	3000	-	-	$1,576,960$	$1,155,000$
ReLU	4096	3000	-	-	0	0
Dropout (0.5)	4096	3000	-	-	0	0
Fully-con. (FC2)	4096	3000	-	-	$16,781,312$	$9,003,000$
ReLU	4096	3000	-	-	0	0
Dropout (0.5)	4096	3000	-	-	0	0
Fully-con. (FC3)	N	N	-	-	$4096 \times N + N$	$3000 \times N + N$
Softmax	N	N	-	-	0	0

3.3. Dynamic Texture CNN

The main idea of our method is to use CNNs on three orthogonal planes. Our method thus learns and pools dense features which are expected to be repetitive in the spatial and temporal domains. This approach is partly inspired by the LBP-TOP [5] and influenced by the ensemble models [7] which extract useful diverse knowledge from the training data by learning different models in parallel and averaging their prediction. The overall framework of our Dynamic Texture CNN (DT-CNN) is detailed in Figure 1. We refer to the developed DT recognition frameworks as DT-AlexNet and DT-GoogleNet depending on which T-CNN architecture is used.

Table 2: Architectures of the T-CNN and T-CNN50 based on GoogleNet, where c is the number of color channels and N is the number of classes. The ReLU and LRN layers are used in the same way as [49].

Layer type	depth	**Output sizes**		**kernel, pad, stride**	
		T-CNN	T-CNN50	T-CNN	T-CNN50
crop	-	$c \times 224 \times 224$	$c \times 48 \times 48$	-	-
Conv (C1)	1	$64 \times 112 \times 112$	$64 \times 48 \times 48$	7, 3, 2	5, 2, 1
Pool (P1)	-	$64 \times 56 \times 56$	$64 \times 24 \times 24$	3, 0, 2	2, 0, 2
Conv (C2)	2	$196 \times 56 \times 56$	$192 \times 24 \times 24$	3, 1, 1	3, 1, 1
Pool (P2)	-	$256 \times 28 \times 28$	$256 \times 12 \times 12$	3, 0, 2	2, 0, 2
Incep (3a)	2	$480 \times 28 \times 28$	$480 \times 12 \times 12$	-	-
Incep (3b)	2	$512 \times 28 \times 28$	$512 \times 12 \times 12$	-	-
Pool (P3)	-	$512 \times 14 \times 14$	$512 \times 6 \times 6$	3, 0, 2	3, 0, 2
Incep (4a)	2	$512 \times 14 \times 14$	$512 \times 6 \times 6$	-	-
Incep (4b)	2	$512 \times 14 \times 14$	$512 \times 6 \times 6$	-	-
Incep (4c)	2	$512 \times 14 \times 14$	$512 \times 6 \times 6$	-	-
Incep (4d)	2	$528 \times 14 \times 14$	$528 \times 6 \times 6$	-	-
Energy	-	528	528	-	-
Fully-con. (FC1)	1	1024	1024	-	-
Fully-con. (FC1)	1	1024	1024	-	-
Dropout (0.4)	-	1024	1024	-	-
Softmax	-	N	N	-	-

3.3.1. Slicing the Dynamic Texture data

We must process (slice) the sequences of DT to enable the CNNs to train and test on the three orthogonal planes. The diagram explaining the pre-processing of a DT sequence is shown in Figure 3.

XY plane (spatial): We represent a sequence of DT with d frames of size $h \times w$ as a matrix $S \in \mathbb{R}^{h\times w\times d\times c}$ with h (height), w (width) and d (depth) respectively in the x, y and t axes and c the number of color channels (i.e., three for rgb, one for grayscale). In the spatial plane, we simply extract m slices (frames) equally spaced in the temporal axis from S. We resize all the frames using bilinear interpolation to the size $n \times n$ to obtain a sequence $S_{xy} \in \mathbb{R}^{n\times n\times m\times c}$ with $m \leq \min(d, h, w)$ and $n \leq \min(d, h, w)$.

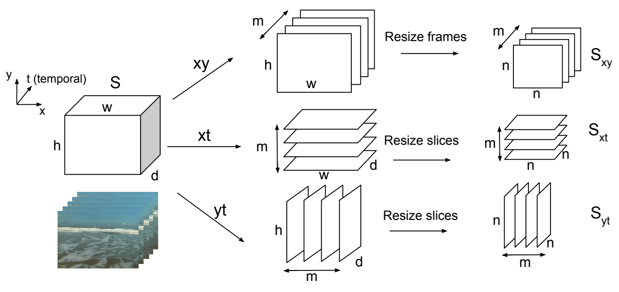


Figure 3: Diagram of DT sequence slicing in three orthogonal planes.

XT and YT planes (temporal): From the same sequence S, we extract m slices in the xt and yt planes, equally spaced on the y axis and x axes respectively. We resize the slices to the size $n \times n$ to obtain the sequences $S_{xt} \in \mathbb{R}^{n \times n \times m \times c}$ and $S_{yt} \in \mathbb{R}^{n \times n \times m \times c}$ with $m \leq \min(d, h, w)$ and $n \leq \min(d, h, w)$. A slice in the xt and yt planes reflects the evolution of a row and a column of pixels respectively over time throughout the sequence.
After pre-processing the sequences, we obtain three sets of sequences of slices which represent the same sequences in three different planes. Examples of spatial and temporal slices are shown in Figure 4.

3.3.2. Training on three planes

In order to train the CNNs, we split the sequences into training and testing sets. Details on the training and testing splits are provided in the experimental setups in Section 4.1. A dataset containing M original sequences is split into N training sequences and $(M - N)$ testing sequences. In each plane, we have $N \times m$ training and $(M - N) \times m$ testing slices, with m the number of slices per sequence. In each plane, the training slices extracted from it are used to fine-tune an independent neural network. In the testing phase, we classify all the slices in each plane and combine the outputs as explained in the following section. The networks used in our experiments are detailed in Sections 3.1 and 3.2 while the hyperparameters are described in Section 4.2.

3.3.3. Sum collective score

Because we train an independent network for each of the three orthogonal planes, we must combine the outputs in the testing phase. We operate a collective score by summing the outputs of the CNNs. First, since each sequence is represented in each plane by a stack of slices, we obtain the score in each plane for a particular sequence by summing the outputs of all the slices of a same sequence.

If we have m slices per sequence, the score vector of a sequence in a plane p is given as:

$$\mathbf{s}^p = \sum_{i=1}^{m} \mathbf{s}_i^p, \tag{2}$$

where $\mathbf{s}_i^p \in \mathbb{R}^N$ is the output (non-normalized classification score) of the last fully-connected layer of the i^{th} slice of the input sequence on plane p, with $p \in \{xy, xt, yt\}$. N is the number of classes. Note that all the scores $\mathbf{s}_i^p$ with $i = \{1, ..., N\}$ are obtained with the same fine-tuned network for a particular plane p and that each plane uses an independently fine-tuned network.
We then obtain a global score for a particular sequence by summing over the three planes.

$$\mathbf{s} = \sum_{p=\{xy,xt,yt\}} \mathbf{s}^p \tag{3}$$

Note that in the experiments we also sum over two planes or single planes to analyze their contribution and complementarity.
The collectively detected label l for a sequence is the one for which the sum score $\mathbf{s}$ is maximum:

$$l = \arg\max_i(\mathbf{s}[i]), \tag{4}$$

where $i = \{1, ..., m\}$ enumerates the DT classes. This ensemble model approach combines three weak neural network classifiers to create a more accurate one as suggested in [7]. We use a data fusion approach as it requires three network classifiers to recognize three data types derived from the sequences (i.e. sequence slices in three different planes). By using a sum collective score, we take into consideration the classification confidence given by the output vector of the last fully-connected layer. We define confidence in this context as the magnitude of the output neurons of the convolutional network. Each neuron gives a score similar to a non-normalized probability for the input image to belong to a certain class.

Also, the confidence sum of the raw output of the last fully-connected layer gives better results than the sum of the softmax normalized probability output. Using the raw output, large non-normalized scores can be attributed to a sequence by a single plane for a particular class if the confidence is high. This is similar to an automatic weighting strategy based on the detection confidence of each networks. We also confirmed experimentally that the results obtained with this sum collective score are better than using a majority or a Borda count voting scheme.

3.4. Domain Transfer

We use CNNs pre-trained on ImageNet to transfer the knowledge learned from this large image dataset to our three planes analysis. While we are only able to pre-train on the spatial (xy) axis since ImageNet does not contain videos, we find out that it transfers relatively well to the analysis of temporal axes (xt and yt). As suggested in [50], transferring features from the relatively distant task of spatial images recognition is better than using random features for the recognition of temporal images. The pre-trained networks are fine-tuned on the spatial or temporal target datasets by stochastic gradient descent. The learning rate is increased in the last fully-connected layer (10 times larger than other trainable layers) to remodel the pre-trained features for the target classification task. Although the improvement in terms of accuracy is minor, using pre-trained networks also makes the training significantly faster (two to three times faster depending on the datasets and networks).

Finally, we are aware of existing video datasets such as UCF101 (human actions [52]) and Sports-1M [37]. However, our method is designed for the

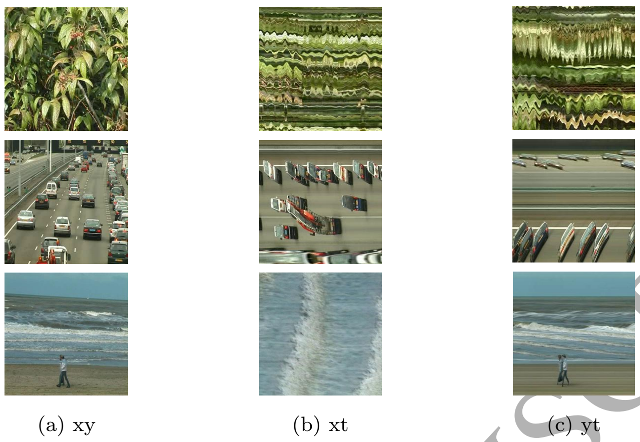


Figure 4: Examples of DT slices in three orthogonal planes of foliage, traffic and sea sequences from the DynTex database. (a) xy (spatial), (b) xt (temporal) and (c) yt (temporal).

classification of segmented DTs. The temporal slices are extracted at multiple spatial locations of the sequence and should exhibit the dynamic of the analyzed DT. Therefore, we do not pre-train our networks on such video datasets as slicing such sequences would extract multiple types of dynamic with potentially only a few slices that represent the dynamic of interest.

4. Experiments

In this section, we present the datasets and protocols used in our experiments and provide implementation details. The results obtained with our method are compared with the state of the art.

4.1. Datasets

In our experiments, we use three datasets organized in seven benchmarks to test our algorithm and compare our results to the state of the art. These experiments cover most of the datasets used in the literature. They include many DT types, various number of training samples, balanced and unbalanced classes, static and moving cameras, rotation, illumination and scale variation as well as several validation setups (leave-one-out, N-folds cross-validation and random splits).

The DynTex Database [45] is a diverse collection of high-quality DT videos. Three sub-datasets are compiled from the DynTex database. For all of them, we use the processed and downsampled (352×288) color images. All sequences are at least 250 frames long, we therefore use the first 250 frames of all sequences to extract the slices. We use the same experimental setup as in [43] with a leave-one-out classification. The three sub-dataset are defined as follows.

DynTex alpha contains 60 DT sequences equally divided into three categories: “sea” (20), “grass” (20) and “trees” (20), where the numbers represent the number of sequences in each class.

DynTex beta contains 162 sequences divided into 10 classes: “sea” (20), “vegetation” (20), “trees” (20), “flags” (20), “calm water” (20), “fountains” (20), “smoke” (16), “escalator” (7), “traffic” (9) and “rotation” (10), where the numbers represent the number of sequences in each class.

The *DynTex gamma* dataset [45] contains 264 sequences grouped into 10 classes: “flowers” (29), “sea” (38), “naked trees” (25), “foliage” (35), “escalator” (7), “calm water” (30), “flags” (31), “grass” (23), “traffic” (9) and “fountains” (37), where the numbers represent the number of sequences in each class.

Dyntex++ [35] is derived from the DynTex database with images cropped and processed. It consists of 36 classes of 100 sequences. Each sequence contains 50 grayscale images of size 50×50. We reproduce the experimental setup from [14, 21, 35]. 50 sequences are randomly selected from each class as the training set, and the other 50 sequences are used in testing. This process is repeated 20 times to obtain the average classification rate.

The UCLA database [1] contains 50 classes of four DT sequences each. Each DT sequence includes 75 grayscale frames of size 160×110 which are cropped to the size 48×48 to show representative dynamics. The classes can be grouped together to form more challenging sub-datasets. We reproduce the three mostly used setups which are defined as follows.

UCLA 50-class: For this experiment, the setup is reproduced from [35, 28, 14]. A 4-fold cross-validation is performed with three sequences of each class used for training (i.e., 150 sequences), the remaining one for testing (i.e., 50 sequences) in each of the four splits.

UCLA 9-class: For the 9-class case, we reproduce the experimental setup from [14, 28, 35]. The sequences taken from different viewpoints are grouped into 9 classes: "boiling water" (8), "fire" (8), "flowers" (12), "fountains" (20), "plants" (108), "sea" (12), "smoke" (4), "water" (12) and "waterfall" (16), where the numbers represent the number of sequences in each class. In each class, 50% of the sequences are used for training (i.e., 100 sequences), the other 50% for testing (i.e., 100 sequences). We report the average classification rate over 20 trials with random splits that are created once and used unchanged throughout the experiments.

UCLA 8-class is similar to the 9-class setup except that the "plant" category is removed since it contains many more sequences than the other classes. The number of remaining sequences is 92 (i.e., 46 for training and 46 for testing). We apply the same 20 trials as in the 9-class setup. This experimental setup is reproduced from [14, 28].

4.2. Implementation details

Our networks are implemented with Caffe [53][2]. As explained in section 3, we use different architectures to adapt the T-CNNs to the large difference of image sizes in the various experiments. However, we report the results without distinction as we use the same method described in Section 3.3 for all the experiments.

The hyperparameters of the networks slightly depend on the size of the datasets, setups and input sizes and are summarized in Table 3. The number of slices per sequence is equal in the three planes and the results are relatively robust to changes of this number. For Dyntex++ and UCLA, we use a number

[2]An implementation of the method is available at the following GitHub repository: `https://github.com/v-andrearczyk/DT-CNN`

Table 3: Hyperparameters used for training both T-CNNs based on AlexNet and GoogleNet on different datasets. From left to right: initial learning rate, factor gamma by which the learning rate is multiplied at every step, weight decay, momentum, batch size, number of iterations and steps.

hyperparam.	lr	γ	weight decay	momentum	batch size	nb. iter	steps
Dyn++	0.01	0.01	0.0005	0.9	64	25000	5000,20000
UCLA-9	0.01	0.01	0.0005	0.9	64	4000	1000,3000
Dyn, UCLA-50, UCLA-8	0.0001	0.1	0.004	0.9	64	2000	1500

Table 4: Average training and testing runtime (in second) of the proposed DT-CNN on one plane. The number of testing sequences (and number of slices in brackets) classified in the given runtime is reported in the second row. The runtime for DynTex-alpha, -beta and -gamma is the same.

	Dyntex++		DynTex		UCLA-50		UCLA-9		UCLA-8	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
test seq. (slices)		1800 (90000)		1 (10)		50 (2400)		100 (4800)		46 (2208)
DT-AlexNet	523	12.1	94.9	0.015	56.9	0.359	113	0.654	53.4	0.292
DT-GoogleNet	3047.7	39.3	176.8	0.037	222.3	1.12	435.9	2.14	215.9	1.01

of slices equal to the height, width and depth of the sequences, i.e., 50 and 48 respectively. The sequences of the DynTex database being much larger ($352 \times 288 \times 250$), we use only 10 slices per plane. We also resize all the slices after extraction to 227×227. We train our networks using stochastic gradient descent with softmax and multinomial logistic loss.

The training and testing runtime using the Caffe implementation on a single GTX 1080 GPU is reported in Table 4 for one training/testing split for each dataset.

4.3. Results

The results of our DT-CNN are compared with the state of the art in Table 5. We include the methods that obtain the best results on each dataset, namely: 9-planes mask Directional Number transitional Graph with SVM classifier (denoted DNG) [33], Dynamic Fractal Spectrum (denoted DFS) [28], spatial Transferred ConvNet Features (denoted s-TCoF) and concatenation of spatial and temporal Transferred ConvNet Features (denoted st-TCoF) [43] and Mul-

Table 5: Accuracy results (%) on DT datasets of the proposed DT-CNN and of the state of the art.

Methods	Dyntex++	DynTex-alpha	DynTex-beta	DynTex-gamma	UCLA-50	UCLA-9	UCLA-8
DL-PEGASOS [35]	63.7	-	-	-	99	95.6	-
LBP-TOP [5, 54, 43]	71.2	96.67	85.8	84.85	86.1	-	96.8
DNG [33]	93.8	-	-	-	-	**99.6**	**99.4**
DFS [28]	89.9	-	-	-	**100**	97.5	99
WLBPC [14]	95.01	-	-	-	96.5	97.17	97.61
MEWLSP [15]	98.48	-	-	-	96.5	98.55	98.04
s-TCoF [43]	-	**100**	99.38	95.83	-	-	-
st-TCoF [43]	-	98.33	98.15	98.11	-	-	-
PCANet-TOP [44]	-	95	90.74	-	88.64	-	-
DT-AlexNet	98.18	**100**	99.38	**99.62**	99.5	98.05	98.48
DT-GoogleNet	**98.58**	**100**	**100**	**99.62**	99.5	98.35	99.02

tiresolution Edge Weighted Local Structure Pattern (denoted MEWLSP) [15]. The results of other methods from the literature are also reported for comparison as follows: DL-PEGASOS [35], LBP-TOP [5], Webers law based LBP with center pixel (WLBPC) and PCANet-TOP [44]. The results of LBP-TOP are provided in [54] and [43] with the original author's implementation [5] for the former and their own implementation for the latter. The results reported for the other mentioned methods from the literature are taken from the respective publications. Our method obtains consistent high accuracy results on all the datasets and overall outperforms all other methods in the literature. The UCLA dataset contains few training samples per class and therefore our method does not outperform the current state of the art. However, a significant improvement is shown on the larger datasets DynTex and Dyntex++. The consistency of high accuracy results across all the datasets shows the robustness of our method. The deep DT-GoogleNet outperforms the shallower DT-AlexNet even though more challenging datasets would be required to fully demonstrate the power of using deep architectures. In the following, we discuss the results of each experiment in more detail.

DynTex alpha: The classification rates of our DT-AlexNet and DT-GoogleNet are 100% on this dataset with high inter-class variability and low intra-class variability and each sequence has at least one other sequence of the same class

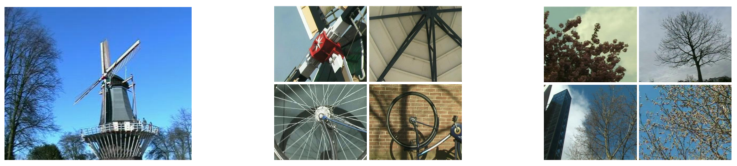


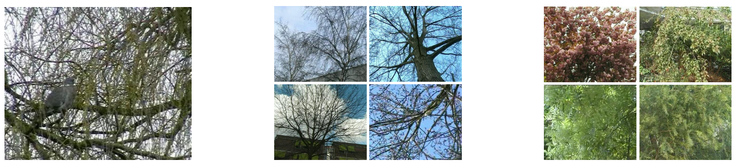
Figure 5: Misclassified sequence of the DynTex beta dataset and sequences examples from the true class and from the detected one. (a) misclassified sequence, (b) true class "rotation" and (c) detected class "trees".

which is very similar. The spatial TCoF (s-TCoF) in [43] also obtains a 100% classification on this sub-dataset.

DynTex beta: The recognition rate of our DT-AlexNet is 99.38% on DynTex-beta and 100% with DT-GoogleNet, which improves the best result obtained by the s-TCoF [43]. The only sequence misclassified with DT-AlexNet is shown in Figure 6a with some sequences of the true class (6b) and some of the wrong detected class (6c). The background of this misclassified sequence contains mainly a blue sky and trees and occupies the major spatial part of the sequence.

DynTex gamma: On the DynTex-gamma dataset, only one sequence is misclassified (out of 264) by our method (with both DT-CNNs), improving the state of the art [43] from 98.11% to 99.62%. The 79$^{\text{th}}$ sequence of true class "naked trees" is misclassified as class "foliage" with a very similar appearance and dynamics. As shown in Figure 6, the tree in the misclassified sequence is more dense as compared to other trees in the same class "naked trees", which causes the confusion. This result is satisfying as DynTex-gamma contains relatively high intra-class variations (e.g., "fountains", "flags") and shows inter-class similarities, e.g. "naked trees" vs. "foliage" and "sea" vs. "calm water" both in the spatial appearance and dynamics.

Dyntex++: In this experiment, our DT-GoogleNet method also outperforms the best results in the literature [15]. The classification rate of each of the 36 classes with DT-AlexNet is shown in Figure 7. The most misclassifications occur



(a) misclassified sequence (b) true class (c) detected class

Figure 6: Misclassified sequence of the DynTex gamma dataset and sequences examples from the true class and from the detected one. (a) misclassified sequence, (b) true class “naked trees” and (c) detected class “foliage”.

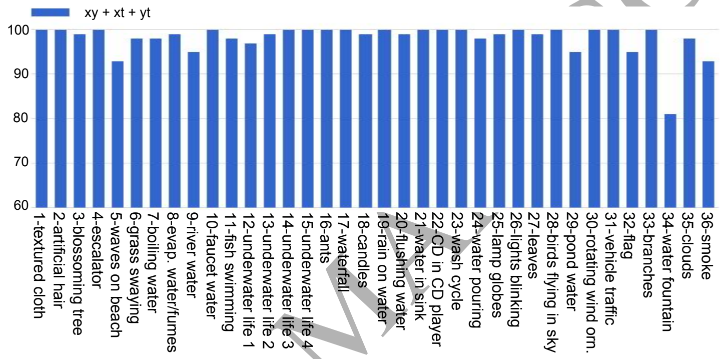


Figure 7: Classification rate of individual classes of the Dyntex++ dataset with our proposed DT-CNN (DT-AlexNet).

for the classes with high intra-class variation “water fountain” and “smoke”, with sequences which, with a closer look, do not always exhibit the expected DT due to the automatic splitting process of the original sequences from DynTex described in [35]. We do not illustrate the confusion matrix due to the large number of classes, yet we notice by visualizing it that there are no dominant categories with which these misclassified sequences are confused (i.e., confusions are spread over several classes). This experiment shows the effectiveness of our deep network method with a relatively high number of training samples and classes as well as high intra-class variation.

The UCLA database contains only four sequences per primary category (50 classes case). Therefore, the number of training sequences per primary category

Table 6: Confusion matrix of the proposed DT-CNN on UCLA 9-class.

	boil	fire	flow.	pl.	fount	sea	sm.	wat.	wfall
boil		gray]1.00	gray]1.00	gray]1.00	gray]1.00	gray]1.00	gray]1.00	gray]1.00	gray]1.00
fire	gray]1.00		gray]1.00	gray]1.00	gray]1.00	gray]1.00	gray]0.970.03	gray]1.00	gray]1.00
flower	gray]1.00	gray]1.00		gray]1.00	gray]0.920.08	gray]1.00	gray]1.00	gray]1.00	gray]1.00
fount	gray]1.00	gray]1.00	gray]1.00		gray]1.00	gray]1.00	gray]1.00	gray]1.00	gray]0.960.04
plants	gray]1.00	gray]1.00	gray]1.00	gray]1.00		gray]1.00	gray]1.00	gray]1.00	gray]1.00
sea	gray]1.00	gray]1.00	gray]1.00	gray]1.00	gray]1.00		gray]1.00	gray]0.970.03	gray]1.00
smoke	gray]1.00	gray]1.00	gray]1.00	gray]1.00	gray]1.00	gray]1.00		gray]0.750.25	gray]1.00
water	gray]1.00	gray]1.00	gray]1.00	gray]1.00	gray]1.00	gray]1.00	gray]0.970.03		gray]1.00
wfalls	gray]1.00	gray]1.00	gray]1.00	gray]0.990.01	gray]1.00	gray]1.00	gray]1.00	gray]1.00	

is three, which is much lower than the other datasets. CNNs require a high number of training samples to adjust the weights in order to generalize the recognition task to new unknown data and avoid overfitting the training data. The database being not highly challenging with low intra-class variation, our method (with both DT-AlexNet and DT-GoogleNet) is still able to reach nearly 100% classification and is close to the state of the art. However, due to the training size, we do not outperform the best shallow methods in the literature on the three sub-datasets as described below.

UCLA-50: We achieve 99.5% classification with only one sequence misclassified out of 200 with both DT-CNNs. The best performance in the literature is 100%, achieved in [28], though our DT-CNNs largely outperform this approach by over 8% on the Dyntex++ dataset.

UCLA-9: Our method obtains 98.05% and 98.35% classification rate with DT-AlexNet and DT-GoogleNet respectively, close to the state of the art (99.6%) [33]. The confusion matrix obtained with DT-AlexNet is shown in Table 6 for which one should keep in mind that the number of sequences per class varies. One can notice that the number of training samples of a class largely influences the recognition rate of the test samples of that same class. The "smoke" class contains only four samples and obtains the lowest 75% classification. These results confirm that our method performs best on large datasets with a high number of training samples per class due to the deep learning scheme.

UCLA-8: Our method achieves a classification rate of 98.48% with DT-AlexNet and 99.02% with DT-GoogleNet, close to the state of the art on this sub-dataset of 99.4% [33]. The confusion matrix in Table 7 shows that the few confusions with DT-AlexNet mainly occur for classes exhibiting high similari-

Table 7: Confusion matrix of the proposed DT-AlexNet on UCLA 8-class.

	boil	fire	flower	fount	sea	smoke	water	wfalls
boil		gray]0.990.01	gray]1.00	gray]1.00	gray]1.00	gray]1.00	gray]1.00	gray]1.00
fire	gray]0.990.01		gray]1.00	gray]1.00	gray]1.00	gray]1.00	gray]1.00	gray]1.00
flower	gray]1.00	gray]1.00		gray]1.00	gray]1.00	gray]1.00	gray]1.00	gray]1.00
fount	gray]1.00	gray]1.00	gray]1.00		gray]1.00	gray]1.00	gray]1.00	gray]1.00
sea	gray]1.00	gray]1.00	gray]1.00	gray]1.00		gray]0.980.02	gray]0.980.02	gray]0.990.01
smoke	gray]1.00	gray]1.00	gray]1.00	gray]1.00	gray]1.00		gray]1.00	gray]1.00
water	gray]1.00	gray]1.00	gray]1.00	gray]1.00	gray]1.00	gray]1.00		gray]1.00
wfalls	gray]1.00	gray]1.00	gray]1.00	gray]0.990.01	gray]1.00	gray]0.980.02	gray]0.990.01	

Table 8: Accuracy results (%) on DT datasets of the proposed DT-AlexNet using various combinations of planes.

Methods	Dyntex++	DynTex-alpha	DynTex-beta	DynTex-gamma	UCLA-50	UCLA-9	UCLA-8
xy	94.28	**100**	98.77	98.11	99	95.7	98.04
xt	96.57	**100**	95.68	97.35	98.5	97.4	95.76
yt	96.28	**100**	95.06	97.73	93	97.15	95.65
$xy + xt$	97.57	**100**	**99.38**	99.24	**99.5**	97.4	98.26
$xy + yt$	97.71	**100**	**99.38**	**99.62**	99	97.8	**98.59**
$xt + yt$	97.84	**100**	97.53	98.11	98	97.95	96.85
$xy + xt + yt$	**98.18**	**100**	**99.38**	**99.62**	**99.5**	**98.05**	98.48

ties in the spatial appearance and/or dynamics ("sea", "smoke", "water" and "waterfalls").

Finally, the Area Under the ROC Curve (AUC) was calculated as some of the datasets are imbalanced (UCLA). The micro and macro average ROC curves across classes were calculated for each run or fold. The micro and macro curves were then averaged again across runs or folds. The AUCs of both micro and macro ROC averages were above 99% for both DT-AlexNet and DT-GoogleNet on all the datasets. On the most imbalanced dataset UCLA-9, DT-AlexNet obtains a micro and macro average AUC of 0.9934 and 0.994 respectively.

4.4. Contribution of planes

The results obtained with DT-AlexNet using different combinations of planes are reported in Table 8.

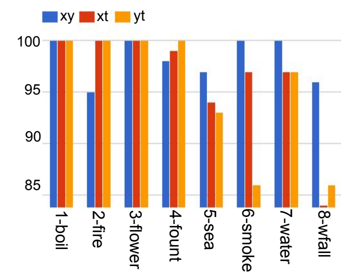
4.4.1. Single plane analysis

Firstly, while the spatial analysis of the xy plane is better overall than the temporal ones (xt and yt), the temporal analyses are close to, and at times outperform, the spatial analysis.

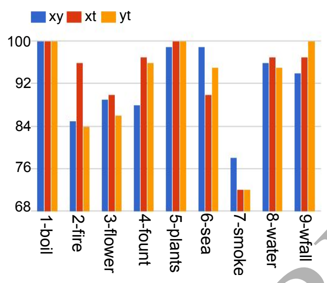
Note that the performance of the xt and yt planes for some classes depends a lot on the angle at which the DT videos are captured as high temporal information is sometimes captured in one or the other plane. For instance, translational motions such as waves and traffic may be contained in only one of the temporal planes if the direction of the motion is parallel/perpendicular to one of the spatial axes. Furthermore, an independent CNN is fine-tuned on each plane and thus learns differently. Therefore, we notice differences between the temporal analyses of the xt and yt planes.

The good results obtained with both single temporal planes go against early conclusions made in the literature stating that DT recognition mostly relies on the spatial analysis while the motion analysis can only provide minor complementary information [21, 35]. This statement is based on experimental results and on the human perception of DTs. While it might be true for the experiments conducted in [21, 35] and for a human being able to differentiate most DTs with a single frame, we show that it does not generalize to all the analysis methods, and that deep network approaches are excellent at finding their own ways to analyze things. Indeed, when it might be very difficult for a human to recognize sequences of temporal slices like those shown in Figures 4b and 4c, our method is able to accurately classify sequences given only the temporal slices. Similar observations were made in [5] in which the LBP histograms on all three single planes result in similar accuracies on the DynTex dataset. One should note that in both our method and [5], the analysis of the xt and yt planes are not purely temporal as they reflect the evolution of 1D spatial lines of pixels over time. Yet these planes exhibit temporal variations and these types of approaches are generally referred to as temporal analysis.

In our experiments, the spatial analysis performs 1% to 3% better than the temporal ones for most of the sub-datasets (DynTex beta, DynTex gamma, UCLA-50 and UCLA-8). However, the temporal analysis outperforms the spatial one on the Dyntex++ and the UCLA-9 sub-datasets. The accuracy of each class using single plane analysis is shown in Figures 8a, 8b, 9a, 9b and

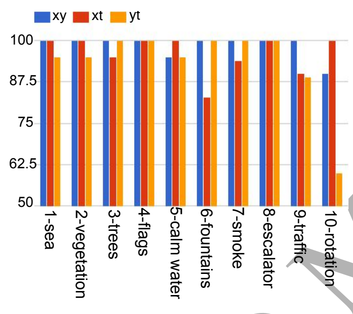


(a) ucla-8

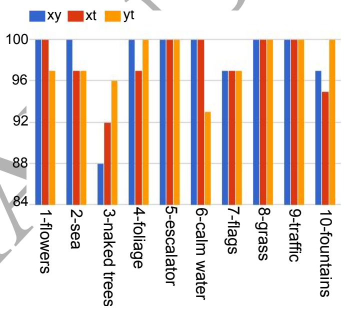


(b) ucla-9

Figure 8: Classification rate of individual classes using single xy, xt and yt planes with DT-AlexNet on the (a) ucla-8 and (b) ucla-9 sub-datasets.



(a) DynTex beta



(b) DynTex gamma

Figure 9: Classification rate of individual classes using single xy, xt and yt planes with DT-AlexNet on the (a) DynTex beta and (b) DynTex gamma sub-datasets.

10 for respectively the UCLA-8, UCLA-9, DynTex-beta, DynTex-gamma and Dyntex++ datasets. It is difficult to find consistent results across datasets to conclude and generalize what recognition the spatial analysis is able to achieve better than the temporal ones.

570 *4.4.2. Combination of planes*

The combination of all the planes results in a consistent increase of accuracy as compared to the sole spatial analysis. Similarly to [5], these results show the complementarity of the spatial and temporal planes. In [5], the authors assign

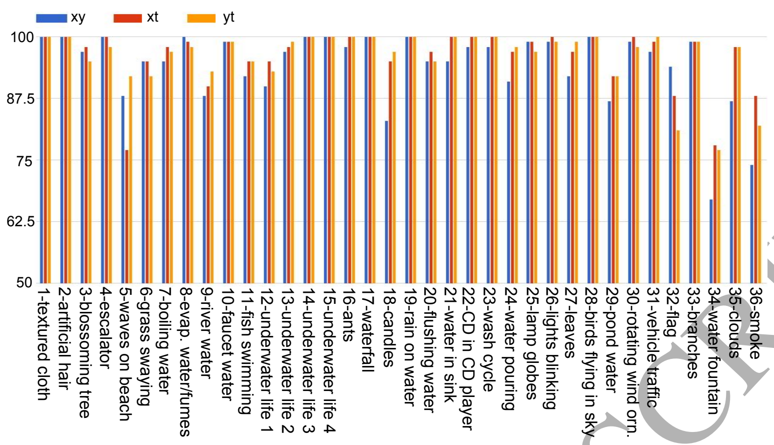


Figure 10: Classification rate of individual classes of the Dyntex++ dataset using single xy, xt and yt planes with DT-AlexNet.

weights to the planes depending on the test results in order to increase the recognition rate. We want to keep a fully automated system and prefer not to report such experiments. Also, we do not assign weights based on the training loss which varies drastically or another measure of performance calculated on the training set because the learned parameters do not always generalize well to the test set; e.g., a low training loss may result in low test accuracy if the network overfits the training data.

The xy plane captures the spatial information while the other two planes mainly capture the dynamics of the sequences. As expected, combining the spatial plane to a single temporal plane, either xt or yt performs generally better than the sole spatial plane or the two temporal planes combined. The best overall performance is achieved by the combination of the three planes which again shows their complementarity.

4.5. Domain transferability and visualization

In this section, we will start with a comparison of DT-AlexNet using networks from scratch and pre-trained. The results are compared in Figure 11. We

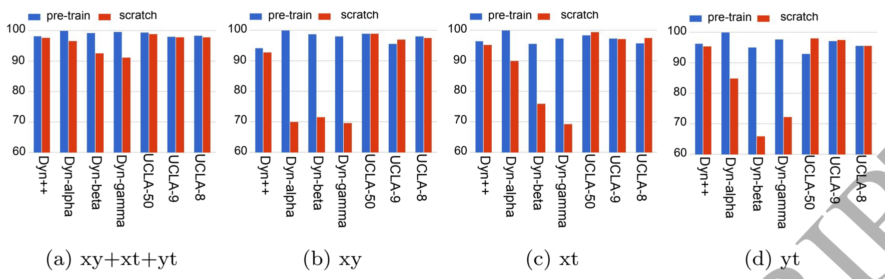


Figure 11: Classification rate of DT-AlexNet with networks trained from scratch vs pre-trained on ImageNet with the following planes: (a) xy+xt+yt (b) xy, (c) xt and (d) yt.

notice that using pre-trained networks only slightly improves the accuracy of our method using the combination of three plans (see Figure 11a). The networks are able to learn from scratch due to the relatively large number of samples resulting from the slicing approach. As shown in Figure 11, the networks used for Dyntex++ and UCLA (T-CNN50) learn better from scratch than the larger T-CNNs used for the DynTex datasets. It is particularly true for single plane methods (Figures 11b to 11d). Having fewer parameters, they learn better from small datasets with less overfitting and thus do not benefit as much from the pre-training.

Moreover, one could expect the spatial analysis to benefit extensively more from the pre-training than the temporal analysis as the source (ImageNet) and target (spatial textures) domains are closer. Yet, the temporal planes benefit almost similarly to the spatial one which demonstrates the transferability of the learned parameters across domains. Note that it also shows that the learning is not biased by an overlap (similar images and classes) between the pre-training ImageNet dataset and the DT datasets. We will now comment on the kernels learned by our method and the features that they detect. First of all, we notice that in ImageNet, some classes contain images with high texture content (tone variation and repetitive patterns) such as “cheetah”, “chainlink fence”, “theater curtains” and “grille”. As shown in [2], pre-training T-CNN with such classes transfers better on texture datasets than pre-training with object-like classes.

We also notice here that inputting spatial or temporal slices from the DynTex database to T-CNN pre-trained on ImageNet often highly activates neurons that learned to recognize these texture classes. Thus, the network is able to learn texture-like patterns already from non segmented textures (e.g. cheetas) and this is why it transfers well to (spatial and temporal) texture images. Moreover, the response at the last convolution layer of the T-CNN to both spatial and temporal DT slices is more dense than the response to an object image. This is also expected and motivates the use of the energy layer (average pooling) for texture and DT analysis.

We also compare the features that the neurons of the fine-tuned and scratch T-CNNs respond most to using the gradient ascent visualization method [55]. We notice, as one would expect, that the neurons of the intermediate layers respond to more complex patterns in fine-tuned networks than those trained from scratch. However, there is very little difference in complexity between the maximum activation of the neurons of the last fully-connected layer. The neuron mainly detect features that exhibit simple repeated patterns. It shows that the dense orderless pooling strategy of the energy layer discards the overall shape information to focus on the repetition of more simple patterns.

Finally, note that for some datasets, the temporal slices xt and yt could be probabilistically expected to exhibit the same dynamic. It would be the case for instance with random rotations of the field of view across different sequences and/or for DT sequences without dominant orientation of motion. In such scenario, it could be useful to combine the temporal slices xt and yt into a single analysis i.e. using two fine-tuned networks instead of three and obtain more rotation invariance. This approach did not result in an increase of accuracy in the experiments proposed in this paper.

5. Conclusion

We developed a new method designed for the analysis of DT sequences based on CNNs applied on three orthogonal planes. We have shown that training independent CNNs on three orthogonal planes and combining their outputs in an ensemble model manner performs well on DT classification by learning to jointly recognize spatial and dynamic patterns. We based this work on our previously described T-CNN (specifically designed for texture images) and developed a new network for small images (Dyntex++ and UCLA) as well as deeper T-CNN networks adapted from GoogleNet to texture analysis. We experimented with our method on the most used DT datasets in the literature, yet a major problem remains the lack of larger and more challenging DT datasets to fully exploit the power of deep learning. Our deepest approach (DT-GoogleNet) obtains slightly higher accuracy than the shallower one (DT-AlexNet) and established a new state of the art on the DynTex and Dyntex++ databases. It achieved less than 1% lower accuracy than the state of the art on the UCLA database which contains fewer training samples, making the CNN training difficult. The neural network learning process enables our method to analyze and learn from many diverse datasets and setups with a good invariance to rotation, illumination, scale, sequence length and camera motion. Thus, our method obtains high accuracy in all the tested datasets (>98%) whereas previous methods in the literature are more specialized in the analysis of one particular database.

We also showed high accuracy results using single temporal planes, at times outperforming the spatial analysis. It demonstrates both the domain transferability of the features learned only on spatial images and the importance of temporal analysis in DT recognition. Finally, we demonstrated the complementarity of the spatial and temporal analyses with the best results obtained by the combination of the three planes over single and two planes analyses.

6. Future research

Our DT-CNN method can be used with any network architecture, yet little difference is observed between shallow and deep models due to the saturation of the DT datasets. A more challenging (including intra-class variation, complex spatiotemporal processes, camera motion, occlusion, etc.) and larger DT database is required to demonstrate the advantage of deep models over shallow ones. The high abstraction and generalization capabilities demonstrated by the consistent high results in this paper and by deep learning results in various complex recognition tasks, should be confirmed in such challenging scenario.

Finally, an interesting line of research would be to incorporate the analysis of the three planes into a single network architecture with an early or slow fusion instead of the late fusion adopted here or by allowing connections in intermediate layers.

References

[1] G. Doretto, A. Chiuso, Y. N. Wu, S. Soatto, Dynamic textures, International Journal of Computer Vision 51 (2) (2003) 91–109.

[2] V. Andrearczyk, P. F. Whelan, Using filter banks in convolutional neural networks for texture classification, Pattern Recognition Letters 84 (2016) 63–69.

[3] M. Cimpoi, S. Maji, I. Kokkinos, A. Vedaldi, Deep filter banks for texture recognition, description, and segmentation, International Journal of Computer Vision 118 (1) (2016) 65–94.

[4] T.-Y. Lin, S. Maji, Visualizing and understanding deep texture representations, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2791–2799.

[5] G. Zhao, M. Pietikainen, Dynamic texture recognition using local binary patterns with an application to facial expressions, Pattern Analysis and Machine Intelligence, IEEE Transactions on 29 (6) (2007) 915–928.

[6] S. Ji, W. Xu, M. Yang, K. Yu, 3D convolutional neural networks for human action recognition, IEEE transactions on pattern analysis and machine intelligence 35 (1) (2013) 221–231.

[7] L. K. Hansen, P. Salamon, Neural network ensembles, IEEE Transactions on Pattern Analysis & Machine Intelligence (10) (1990) 993–1001.

[8] S. Basu, M. Karki, S. Mukhopadhyay, S. Ganguly, R. Nemani, R. DiBiano, S. Gayaka, A theoretical analysis of deep neural networks for texture classification, in: Neural Networks (IJCNN), 2016 International Joint Conference on, IEEE, 2016, pp. 992–999.

[9] T. Randen, J. H. Husoy, Filtering for texture classification: A comparative study, IEEE Transactions on pattern analysis and machine intelligence 21 (4) (1999) 291–310.

[10] L. Sifre, S. Mallat, Rotation, scaling and deformation invariant scattering for texture discrimination, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 1233–1240.

[11] D. Marcos, M. Volpi, D. Tuia, Learning rotation invariant convolutional filters for texture classification, in: Pattern Recognition (ICPR), 2016 23rd International Conference on, IEEE, 2016, pp. 2012–2017.

[12] T.-Y. Lin, A. RoyChowdhury, S. Maji, Bilinear CNN models for fine-grained visual recognition, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1449–1457.

[13] V. Andrearczyk, P. F. Whelan, Texture segmentation with fully convolutional networks, arXiv preprint arXiv:1703.05230.

[14] D. Tiwari, V. Tyagi, Improved Weber's law based local binary pattern for dynamic texture recognition, Multimedia Tools and Applications (2016) 1–18.

[15] D. Tiwari, V. Tyagi, Dynamic texture recognition using multiresolution edge-weighted local structure pattern, Computers & Electrical Engineering`doi:10.1016/j.compeleceng.2016.11.008`.

[16] E. Rahtu, J. Heikkilä, V. Ojansivu, T. Ahonen, Local phase quantization for blur-insensitive image analysis, Image and Vision Computing 30 (8) (2012) 501–512.

[17] J. Chen, G. Zhao, M. Salo, E. Rahtu, M. Pietikäinen, Automatic dynamic texture segmentation using local descriptors and optical flow, Image Processing, IEEE Transactions on 22 (1) (2013) 326–339.

[18] R. M. Haralick, K. Shanmugam, et al., Textural features for image classification, IEEE Transactions on systems, man, and cybernetics 3 (6) (1973) 610–621.

[19] T. Ojala, M. Pietikäinen, D. Harwood, A comparative study of texture measures with classification based on featured distributions, Pattern recognition 29 (1) (1996) 51–59.

[20] P. Gao, C. L. Xu, Extended statistical landscape features for dynamic texture recognition, in: Computer Science and Software Engineering, 2008 International Conference on, Vol. 4, IEEE, 2008, pp. 548–551.

[21] V. Andrearczyk, P. F. Whelan, Dynamic texture classification using combined co-occurrence matrices of optical flow, in: Irish Machine Vision & Image Processing Conference proceedings IMVIP 2015, 2015.

[22] C.-H. Peh, L.-F. Cheong, Synergizing spatial and temporal texture, Image Processing, IEEE Transactions on 11 (10) (2002) 1179–1191.

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

[24] S. Fazekas, D. Chetverikov, Dynamic texture recognition using optical flow features and temporal periodicity, in: Content-Based Multimedia Indexing, 2007. CBMI'07. International Workshop on, IEEE, 2007, pp. 25–32.

[25] A. Ravichandran, R. Chaudhry, R. Vidal, Categorizing dynamic textures using a bag of dynamical systems, Pattern Analysis and Machine Intelligence, IEEE Transactions on 35 (2) (2013) 342–353.

[26] B. Afsari, R. Chaudhry, A. Ravichandran, R. Vidal, Group action induced distances for averaging and clustering linear dynamical systems with applications to the analysis of dynamic scenes, in: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE, 2012, pp. 2208–2215.

[27] K. Fujita, S. K. Nayar, Recognition of dynamic textures using impulse responses of state variables, in: Proc. Third International Workshop on Texture Analysis and Synthesis (Texture 2003), 2003, pp. 31–36.

[28] Y. Xu, Y. Quan, H. Ling, H. Ji, Dynamic texture classification using dynamic fractal analysis, in: Computer Vision (ICCV), 2011 IEEE International Conference on, IEEE, 2011, pp. 1219–1226.

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

[30] Q. Yu-long, W. Fu-shan, Dynamic texture classification based on dual-tree complex wavelet transform, in: Instrumentation, Measurement, Computer, Communication and Control, 2011 First International Conference on, IEEE, 2011, pp. 823–826.

[31] H. Ji, X. Yang, H. Ling, Y. Xu, Wavelet domain multifractal analysis for static and dynamic texture classification, Image Processing, IEEE Transactions on 22 (1) (2013) 286–299.

[32] C. Feichtenhofer, A. Pinz, R. Wildes, Bags of spacetime energies for dynamic scene recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 2681–2688.

[33] A. Ramirez Rivera, O. Chae, Spatiotemporal directional number transitional graph for dynamic texture recognition, Pattern Analysis and Machine Intelligence, IEEE Transactions on 37 (10) (2015) 2146–2152.

[34] F. Yang, G.-S. Xia, G. Liu, L. Zhang, X. Huang, Dynamic texture recognition by aggregating spatial and temporal features via ensemble svms, Neurocomputing 173 (2016) 1310–1321.

[35] B. Ghanem, N. Ahuja, Maximum margin distance learning for dynamic texture recognition, in: Computer Vision–ECCV 2010, Springer, 2010, pp. 223–236.

[36] D. Tran, L. Bourdev, R. Fergus, L. Torresani, M. Paluri, Learning spatiotemporal features with 3D convolutional networks, arXiv preprint arXiv:1412.0767.

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

[38] K. Simonyan, A. Zisserman, Two-stream convolutional networks for action recognition in videos, in: Advances in Neural Information Processing Systems, 2014, pp. 568–576.

[39] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, T. Darrell, Long-term recurrent convolutional networks for visual recognition and description, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 2625–2634.

[40] J. Shao, C.-C. Loy, K. Kang, X. Wang, Slicing convolutional neural network for crowd video understanding, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 5620–5628.

[41] J. Shao, C. C. Loy, K. Kang, X. Wang, Crowded scene understanding by deeply learned volumetric slices, IEEE Transactions on Circuits and Systems for Video Technology.

[42] D. Culibrk, N. Sebe, Temporal dropout of changes approach to convolutional learning of spatio-temporal features, in: Proceedings of the ACM International Conference on Multimedia, ACM, 2014, pp. 1201–1204.

[43] X. Qi, C.-G. Li, G. Zhao, X. Hong, M. Pietikäinen, Dynamic texture and scene classification by transferring deep image features, Neurocomputing 171 (2016) 1230–1241.

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

[45] R. Péteri, S. Fazekas, M. J. Huiskes, DynTex: A comprehensive database of dynamic textures, Pattern Recognition Letters 31 (12) (2010) 1627–1632.

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

[47] V. Andrearczyk, P. F. Whelan, Deep learning in texture analysis and its application to tissue image classification, in: A. Depeursinge, O. S. Al-Kadi, J. R.

Mitchell (Eds.), Biomedical Texture Analysis: Fundamentals, Tools and Challenges, Elsevier, 2017, Ch. 4, pp. 95–129.

[48] D. Bau, B. Zhou, A. Khosla, A. Oliva, A. Torralba, Network Dissection: Quantifying Interpretability of Deep Visual Representations, in: Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on, 2017.

[49] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.

[50] J. Yosinski, J. Clune, Y. Bengio, H. Lipson, How transferable are features in deep neural networks?, in: Advances in Neural Information Processing Systems, 2014, pp. 3320–3328.

[51] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, IEEE, 2009, pp. 248–255.

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

[53] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, Caffe: Convolutional architecture for fast feature embedding, arXiv preprint arXiv:1408.5093.

[54] E. Norouznezhad, M. T. Harandi, A. Bigdeli, M. Baktash, A. Postula, B. C. Lovell, Directional space-time oriented gradients for 3D visual pattern analysis, in: Computer Vision–ECCV 2012, Springer, 2012, pp. 736–749.

[55] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, H. Lipson, Understanding neural networks through deep visualization, arXiv preprint arXiv:1506.06579.

Vincent Andrearczyk received a double Masters degree in electronics and signal processing from ENSEEIHT, France and Dublin City University, in 2012 and 2013 respectively. He is currently working toward the PhD degree in Dublin City University. His research focuses on deep learning for texture and dynamic texture analysis.

Paul F. Whelan is a Professor with the School of Electronic Engineering at Dublin City University. His research interests include image segmentation, and its associated quantitative analysis (specifically mathematical morphology, colour-texture analysis) with applications in computer/machine vision and medical imaging (specifically computer aided detection and diagnosis). http://paulwhelan.eu/