Maximum Margin Distance Learning for Dynamic Texture Recognition

Bernard Ghanem and Narendra Ahuja

Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA
{bghanem2,ahuja}@vision.ai.uiuc.edu

Abstract. The range space of dynamic textures spans spatiotemporal phenomena that vary along three fundamental dimensions: spatial texture, spatial texture layout, and dynamics. By describing each dimension with appropriate spatial or temporal features and by equipping it with a suitable distance measure, elementary distances (one for each dimension) between dynamic texture sequences can be computed. In this paper, we address the problem of dynamic texture (DT) recognition by learning linear combinations of these elementary distances. By learning weights to these distances, we shed light on how "salient" (in a discriminative manner) each DT dimension is in representing classes of dynamic textures. To do this, we propose an efficient maximum margin distance learning (MMDL) method based on the Pegasos algorithm [1], for both class-independent and class-dependent weight learning. In contrast to popular MMDL methods, which enforce restrictive distance constraints and have a computational complexity that is cubic in the number of training samples, we show that our method, called *DL-PEGASOS*, can handle more general distance constraints with a computational complexity that can be made linear. When class dependent weights are learned, we show that, for certain classes of DTs , spatial texture features are dominantly "salient", while for other classes, this "saliency" lies in their temporal features. Furthermore, *DL-PEGASOS* outperforms state-of-the-art recognition methods on the UCLA benchmark DT dataset. By learning class independent weights, we show that this benchmark does not offer much variety along the three DT dimensions, thus, motivating the proposal of a new DT dataset, called DynTex++.

1 Introduction

A dynamic texture (DT) sequence captures a stochastic spatiotemporal phenomenon. The randomness reflects in the spatial and temporal changes in the image signal. This may be caused by a variety of physical processes, e.g., involving objects that are small (smoke particles) or large (snowflakes), or rigid (grass, flag) or nonrigid (cloud, fire), moving in 2D or 3D, etc. Even though the overall global motion of a DT may be perceived by humans as being simple and coherent, the underlying local motion is governed by a complex stochastic model. For example, a scene of "translating" clouds conveys visually identifiable global dynamics; however, the implosion and explosion of the cloud segments during the motion result in very complicated local dynamics. Irrespective of the nature of the physical phenomena, the usual objective of DT modeling in computer vision and graphics is to capture the nondeterministic, spatial and temporal variation in images. The study of DTs poses numerous challenges, especially for traditional motion models that fail to capture their stochastic nature. These challenges arise from the need to capture the large number of objects involved, their complex motions, and their intricate interactions. A good model must accurately and efficiently capture both the appearance and global dynamics of a DT. Despite the diverse types of DTs in nature, we see that they belong to a three dimensional DT space. In this space, each dimension isolates a single aspect that describes the variation of an individual DT. These dimensions are, therefore, broad categories of variation for DTs, in general. However, they are not generally independent, since for some cases of DT, it is not possible to fix two dimensions and vary the third independently. This interdependence is attributed to the physical nature of the phenomena being imaged. In what follows, we will describe each of these dimensions and give their respective ranges. Then, we will designate the portion of the DT space, where this paper operates. Note that the first two dimensions describe the spatial variation and the spatial organization of a DT, while the third describes its temporal variations.

1. **Spatial Texture Element:** This dimension describes the spatial variation of a DT as observed from each frame independently. Texture elements (usually denoted as *texels*) are the spatially repetitive groups of pixels that share statistically similar appearance and structural properties. The spectrum of texture elements varies from the simplest form at the microscopic level (i.e. particles) to the most complex at the macroscopic level (i.e. whole objects). At one extreme, this spectrum has DTs that show clouds, smoke, or water in motion, while at the other, there are DTs of birds, animals, or humans moving. The majority of DT work has focused on pixel or subpixel objects (i.e. microscopic), whereby the pixel is assumed to be the texture element whose motion is to be modeled.
2. **Spatial Texture Layout:** This dimension describes the spatial layout of the texture elements in a DT, as well as, their spatial layering. A DT's spatial layout determines how its texture elements are organized within each frame, especially in terms of their spatial placement. In this sense, there are DTs with homogenously placed/spaced texture elements, as well as, DTs where the placement distribution is non-uniform. Moreover, the spatial layering of a DT refers to the "density" (or translucency) of a DT. For simplicity, spatial layering of a DT can be viewed as the alpha matte of the texture elements, in each frame, when visualized infront of a background layer. The values of this alpha matte take values in $[0, 1]$. For opaque DTs, spatial layering is not an issue, since the background does not appear at all (i.e. the alpha matte is either 0 or 1). For translucent DTs (e.g. clouds and smoke), this layering is essential. The majority of DT work has focused on DTs with opaque texture elements that cover the whole spatial extent of the video.

3. **Dynamics:** This dimension describes the temporal variation of a DT as observed by the frame-to-frame variation in its texture elements and their layering/layout. DT dynamics represent temporal changes in features (e.g. intensity values and linear transformations of these values) describing the texture elements and their layout. Note that the dynamics of a DT is a global motion representation that incorporates the dynamics of individual texture elements and their spatiotemporal interactions. Being a DT means that the dynamics of texture elements are statistically similar and temporally stationary. In other words, texture elements in the same DT all "move" in a similar fashion and their "motions" are not time dependent (i.e. statistically stationary). As such, the models of DT dynamics either make use of physical models (e.g. Navies-Stokes equations [2]) or assume a general parametric model whose parameters are learned by fitting the model to the observed DT frames (e.g. a linear dynamical system [3]). The majority of DT work has concentrated on the latter form of models, where linear/nonlinear models have been proposed to model variations in the intensity values of DTs.

In this paper, we cater to opaque DTs consisting of pixel-based texture elements, whose dynamics can be represented by a linear parametric model [3]. We address the problem of DT recognition, which is motivated by critical real-life applications, especially the detection of the onset of emergencies (e.g. fire). Recognition is done by learning linear combinations of distances between DT sequences, so that classes of DTs are maximally separated. These distances quantify how different two DT sequences are with respect to the three dimensions mentioned above. By learning weights to these distances, we shed light on how "salient" (in a discriminative fashion) each dimension (i.e. spatial and/or temporal) is in representing a single DT class or a whole DT database.

2 Related Work

DT recognition involves the analysis of both image appearance and temporal changes in appearance. For an overview of recent techniques developed for DT recognition, we refer the reader to [4]. Numerous DT recognition methods have stemmed from representing the global spatiotemporal variations of a DT as a linear dynamical system (LDS) [3]. In [5], Doretto et al. use the LDS model parameters and the Martin distance measure [6] to perform nearest neighbor recognition. In [7], a kernel function between two LDS models was proposed and used in a support vector machine (SVM) framework to perform DT recognition. More recent work has addressed shift and view invariant DT recognition [8,9]. The latter work extends the use of the popular bag-of-features model to the non-Euclidean space of LDS models.

Other recognition methods have used a multiplicity of spatiotemporal descriptors to represent a DT sequence. In [10], Peteri et al. propose a DT recognition algorithm based on six translation invariant features. Recent work by Zhao et al. proposed using local binary patterns (LBP) [11] and volume local binary patterns (VLBP) to recognize DT sequences [12,13]. The latter two methods are based on local descriptors, which do not incorporate the global dynamics that characterize a DT.

Despite the merits of these methods, they all either focus on one dimension of the DT space defined before or assume that these dimensions contribute equally and in the same manner for all DT classes. These assumptions are quite restrictive and fail to characterize the discriminative properties of many DTs. To the best of our knowledge, this paper is the first to address the problem of combining the discriminative properties of the three DT dimensions. Here, we provide an intuitive example that motivates why this is important in DT recognition. On one hand, the fire DT class is easily distinguished from other DT classes, primarily due to its highly discriminative dynamics, as compared to its spatial texture appearance. On the other hand, DTs such as moving leaves and grass have a more "salient" spatial texture element.

We infer the contributions of the DT dimensions by using a multiplicity of DT descriptors, each of which operates in a given dimension. We elaborate on these descriptors and motivate their selection later. Since these descriptors are of different dimensions and belong to different spaces, we model the distance between two DT sequences as a weighted sum of the elementary distances between their respective descriptors. Learning these weights in a maximum margin setting will determine the contributions of the DT dimensions, in such a way that maximizes DT class discrimination. Learning weighted distance functions in a maximum margin framework is not new, as it has been successfully applied to image classification and retrieval [14,15] and more recently to region-based object recognition [16]. These approaches impose the following distance constraint: an image is closer to all other images in its class than to images of all other classes. In feature space, this forces classes to be significantly compact, which tends not to be the case for most real data. This "compactness" assumption is quite restrictive and does not generalize well to object classes that share properties (e.g. cow vs. horse). Furthermore, this assumption produces a number of distance constraints/variables that is cubic in the number of training images, since all relevant distance triplets are used. Our method generalizes this "compactness" assumption whereby each DT sequence is only closer to a *representative* set of DTs within its class than to a *comparative* set of DTs outside this class. By taking the *representative* set of a DT to include its k nearest neighbors within its class and its *comparative* set to include all other DTs outside its class, we allow for less compact DT classes and much fewer distance constraints. To reduce computational complexity, we solve the primal version of the maximum margin problem in a way similar to the Pegasos algorithm [1].

Here, we note that distance weight learning finds some similarities with multiple kernel learning (MKL), which has been recently applied to object detection [17,18]. In MKL, the kernels define similarities between elements and are, by definition, symmetric and positive definite kernels. Although similarities can be formed from certain distances (e.g. by parametric negative exponentiation), these distances need not be symmetric and the parameters used to form the similarities need to be set wisely. This method also suffers from a computational drawback, since it requires expensive optimization techniques to learn the kernel mixing coefficients. Moreover, the MKL framework does not readily accommodate the distance constraints required in maximum margin distance learning (MMDL).

Contributions: The contributions of this work are three fold. **(1)** We propose to learn the individual contributions/weights of all three DT dimensions, in regards to DT class discrimination. **(2)** To learn these weights, we propose an efficient MMDL method based on the Pegasos algorithm, whose complexity can be made linear in the number of training samples. **(3)** A new DT dataset, called DynTex++, is compiled to replace the current UCLA benchmark dataset.

This paper is organized as follows. In Section 3, we give an overview of the DT recognition problem, in an MMDL framework. Section 4 provides a detailed description of our proposed solution and algorithm, while Section 5 shows experimental validation of this algorithm, when applied to the UCLA and DynTex++ datasets.

3 Problem Overview

In this paper, we seek to learn how the different dimensions of the DT space can be linearly combined to best discriminate between DT classes. Learning these linear combinations for a given DT class or a group of DT classes sheds light on the relative importance of each DT dimension. We choose a suitable descriptor to represent each dimension, which is characterized by a corresponding elementary distance. Since these descriptors need not belong to vector spaces, the elementary distances are can be of different forms. In this framework, the distance between two DT sequences is modeled as a positively weighted sum of their elementary distances. These weights are learned in a maximum margin fashion, so that DT classes are maximally separated. We consider the case of class independent and class dependent weights.

We assume a set of M training DT sequences (from N classes) is given with corresponding labels in $\{1, \ldots, N\}$. Let $\ell(.)$ denote the labeling function, whereby $\ell(v_i)$ is the label of the DT sequence v_i. The DT sequence v_i has F different DT descriptors[1], which characterize the three different DT dimensions. We define the f^{th} elementary distance from v_j to v_i as $d_f\left(v_i \rightarrow v_j\right)$. Here, we note that these elementary distances need not be symmetric. As such, the combined distance from v_j to v_i is defined as $D_{\boldsymbol{w}_{\ell(v_i)}}\left(v_i \rightarrow v_j\right) = \sum_{f=1}^{F} w_{\ell(v_i)}^{f} d_f\left(v_i \rightarrow v_j\right)$. More compactly, we can combine the elementary distances in vector format to obtain $D_{\boldsymbol{w}_{\ell(v_i)}}\left(v_i \rightarrow v_j\right) = \boldsymbol{w}_{\ell(v_i)}^{T} \boldsymbol{d}\left(v_i \rightarrow v_j\right)$. Here, $w_{\ell(v_i)}^{f}$ is the weight that characterizes the f^{th} elementary distance for class $\ell(v_i)$. Here, we are considering class dependent weights; however, class independent weights are similarly incorporated by dropping the class label from $\boldsymbol{w}_{\ell(v_i)}$.

In order to best separate the DT classes, we assume that each DT of a given class is closer to a *representative* set of DTs within this class than a *comparative* set of DTs outside this class. Let $\mathcal{R}(v_i)$ define the *representative* set corresponding to DT v_i and $\mathcal{C}(v_i)$ define its *comparative* set. Under this assumption,

[1] In this paper, $F = 3$, but the method generalizes to any number of descriptors.

a set of distance constraints arises for each DT v_i, defined as follows. For all $i \neq j, \ell(v_i) = \ell(v_j) \neq \ell(v_k)$, $v_j \in \mathcal{R}(v_i)$, and $v_k \in \mathcal{C}(v_i)$ we have:

$$D_{\boldsymbol{w}_{\ell(v_i)}}(v_i \rightarrow v_j) \leq D_{\boldsymbol{w}_{\ell(v_i)}}(v_i \rightarrow v_k) \Leftrightarrow \boldsymbol{w}_{\ell(v_i)}^T \triangle \boldsymbol{d}(v_i, v_j, v_k) \geq 0 \qquad (1)$$

where $\boldsymbol{d}(v_i, v_j, v_k) = \boldsymbol{d}(v_i \rightarrow v_k) - \boldsymbol{d}(v_i \rightarrow v_j)$ is the distance difference corresponding to the DT triplet v_i, v_j, and v_k. The total number of these constraints is $\sum_i^M |\mathcal{R}(v_i)||\mathcal{C}(v_i)|$. Clearly, this number and thus the scale of the optimization needed to learn $\boldsymbol{w}_{\ell(v_i)}$ depends on the nature of $\mathcal{R}(.)$ and $\mathcal{C}(.)$. In fact, it is bounded by $\Theta(M^3)$ from above and $\Theta(M)$ from below.

Let $\mathbf{A}_c \in \mathbb{R}^{L \times F}$ denote the matrix whose rows are composed of all the distance difference vectors $\triangle\boldsymbol{d}(v_i, v_j, v_k)$ for all DTs v_i where $\ell(v_i) = c$. The distance constraints in Eq. (1) can be formalized as $\mathbf{A}_c\boldsymbol{w}_c \succeq \mathbf{0}$. We embed these constraints in a maximum margin framework, as shown in Eq. (2). In this framework, the cost function includes two terms that work towards minimizing the classification bias and variance. The second term is the average hinge loss cost of the L distance constraints. This cost uses a margin of 1 instead of 0. Although using L_1 regularization is known to lead to sparser solutions, we choose an L_2 regularization term on $\boldsymbol{w}_c$ instead, as it is more robust to noise and outliers and the number of feature descriptors F is relatively too small to benefit from a sparse solution.

$$\min_{\boldsymbol{w}_c \succeq \mathbf{0}} \ \frac{\lambda}{2}\|\boldsymbol{w}_c\|_2^2 + \frac{1}{L}\sum_{i=1}^{L} \max\left(0, 1 - \boldsymbol{w}_c^T \boldsymbol{a}_c(i)\right) \qquad (2)$$

where $\boldsymbol{a}_c(i)$ is the i^{th} row in $\mathbf{A}_c$. It is important to point out that when solving for class independent weights the matrix of distance constraints becomes a concatenation of all $\mathbf{A}_c$ matrices with $c \in \{1, \ldots, N\}$. Furthermore, note that class information need not be provided so long as relative dissimilarities/rankings are. In other words, even when class labels are not given, our method can still be applied, if pairwise distance inequalities are known. So, a statement like "dynamic texture A looks more similar to dynamic texture B than C" can be directly translated to a distance constraint.

The formulation in Eq. (2) is the same one used in the Pegasos algorithm [1], except for the non-negativity constraint on $\boldsymbol{w}_c$. In the next section, we will show how the original Pegasos method can be modified to efficiently solve for $\boldsymbol{w}_c$, to incorporate different forms of $\mathcal{R}(.)$ and $\mathcal{C}(.)$, and to reduce the number of distance constraints used in each Pegasos iteration. In fact, we choose to use this formulation/method instead of the one used in [14,15,16], since the latter does not lend itself suitable for variations in the *representative* and *comparative* sets and it requires a custom solver to handle a large number of distance constraints.

After solving for $\boldsymbol{w}_c$ of each class, a test DT sequence is classified as the class, which satisfies the most (or violates the least) number of distance constraints generated by the test DT. More specifically, for each class in the training set, a logistic regression classifier[2] is learned based on the combined distances of

[2] For a test sequence v_p, $f(v_p|c) = \left(1 + \exp\left(\alpha_0 + \sum_{v_i:\ell(v_i)=c} \alpha_i D_{\boldsymbol{w}_c}(v_i \rightarrow v_p)\right)\right)^{-1}$ defines the logistic regression classifier of class c.

training samples to samples within this class, as done in [16]. The test DT is assigned to the class, whose regression classifier evaluates to the maximum value among all classes. In the case of class independent weight learning, a simple k-nearest neighbor (kNN) classifier can be employed to classify the test DT.

Elementary Distances

In what follows, we present and justify the set of feature descriptors ($F = 3$) that we choose to represent the three DT dimensions of a DT sequence.

1. **Spatial Texture Element:** This DT dimension is described by a histogram of Local Binary Patterns (LBP), which provides a simple yet powerful local depiction of intensity variation. Each frame in a DT is described by an LBP histogram. As such, the elementary distance between two DTs along this dimension is the minimum distance between LBP histograms from these two DTs. To compare histograms, we use the Earth Mover's Distance (EMD) [19], which though more computationally expensive than other distances (e.g. ℓ_2 norm or χ^2), it provides a more accurate histogram distance. This spatial texture descriptor has been successfully utilized in DT recognition [12] and extended to video sequences in [20]. Recently, it has also proven to be useful in improving human detection performance [21].
2. **Spatial Texture Layout:** This DT dimension is described by a Pyramid of Histograms of Oriented Gradients (PHOG), which provide a powerful depiction of local spatial layout. In building the PHOG of a DT frame, we assume uniform weighting for each histogram at a given pyramid level and we normalize with respect to the number of histograms at each pyramid level. We only use two levels in the pyramid. Similar to the LBP descriptor, we use EMD to compute distances between histograms. Prior work has used this descriptor extensively in detecting objects, especially human detection [22], as well as, image retrieval [23].
3. **Dynamics:** To describe the global temporal variations of a DT sequence, we model it as a Linear Dynamical System (LDS) [3]. An LDS model is parameterized by the matrix pair $(\mathbf{A}, \mathbf{C})$, which govern feature generation and state transition. We assume a model size of 25, in our experiments. The LDS model and its variants have been extensively applied to DT recognition, most recently in [8,9]. The elementary distance between two LDS models is the Martin distance between ARMA processes [6].

Since each elementary distance above spans a different range of values, proper normalization is called for. After computing the elementary distances between DT sequences in the training set, we normalize each distance type by its mean (μ) offset by a multiple of its standard deviation (σ). In our experiments, we normalize each elementary distance by its corresponding ($\mu + 3\sigma$).

4 Learning Maximum Margin Weights

In this section, we give a detailed description of the learning algorithm used to compute $\boldsymbol{w}_c$ in Eq. (2). **Algorithm** 1 summarizes the learning process, which is a modified version of the original Pegasos algorithm [1]. *DL-PEGASOS* can handle general definitions for $\mathcal{R}(.)$ and $\mathcal{C}(.)$, since they can be data-driven and/or application specific. Furthermore, these definitions can even be dependent on $\boldsymbol{w}_c$, which explains why $\mathcal{R}(v_i)$ and $\mathcal{C}(v_i)$ must be updated at each iteration of this algorithm (refer to STEPS 2-3). In this case, Eq. (2) is no longer convex and so *DL-PEGASOS* becomes a stochastic, projected[3] subgradient descent method that alternates between **(i)** performing a Pegasos iteration when the sets $\mathcal{R}(v_i)$ and $\mathcal{C}(v_i)$ are fixed for all DT sequences v_i and **(ii)** updating these sets for a fixed Pegasos solution of $\boldsymbol{w}_c$. A study of convergence for *DL-PEGASOS* is kept for future work; however, empirical analysis is very promising.

Algorithm 1. *Distance Learning PEGASOS (DL-PEGASOS)*

```
   Input : $\mathcal{R}(.)$, $\mathcal{C}(.)$, $\{\boldsymbol{d}(v_i \to v_j) : \ell(v_i) = c\}$, $\lambda$, $T$, $m$
1  Initialization: $\boldsymbol{w}_c^{(0)} \in B_\lambda^+ = \{\boldsymbol{x} : \|\boldsymbol{x}\|_2 \le \frac{1}{\sqrt{\lambda}}, \boldsymbol{x} \succeq \mathbf{0}\}$
2  for $t = 0, \dots, T$ do
3      • determine $\mathcal{R}(v_i)$ and $\mathcal{C}(v_i)$ $\forall v_i$ such that $\ell(v_i) = c$ (use $\boldsymbol{w}_c^{(t)}$ if needed)
4      • determine $\mathbf{A}_c \in \mathbb{R}^{L \times F}$
5      // original PEGASOS iteration
6      • Randomly choose $C_t \subseteq \{1, \dots, L\}$, where $|C_t| = m$
7      • Set $C_t^+ = \{i \in C_t : \boldsymbol{a}_c^T(i)\boldsymbol{w}_c^{(t)} < 1\}$ and $\eta_t = \frac{1}{\lambda t}$
8      • Compute subgradient: $\boldsymbol{\nabla}_t = \lambda \boldsymbol{w}_c^{(t)} - \frac{1}{|C_t^+|}\sum_{i \in C_t^+} \boldsymbol{a}_c(i)$
9      • Do subgradient descent step: $\boldsymbol{w}_c^{(t+\frac{1}{2})} = \boldsymbol{w}_c^{(t)} - \eta_t \boldsymbol{\nabla}_t$
10     • Project onto $B_\lambda^+$: $\boldsymbol{w}_c^{(t+1)} = \min\left\{1, \frac{1/\sqrt{\lambda}}{\left\|\left[\boldsymbol{w}_c^{(t+\frac{1}{2})}\right]_+\right\|_2}\right\}\left[\boldsymbol{w}_c^{(t+\frac{1}{2})}\right]_+$
11 end
12 return $\boldsymbol{w}_c^{(T)}$
```

In our MMDL formulation, the distance constraint matrix $\mathbf{A}_c$ is directly dependent on the definition of $\mathcal{R}(.)$ and $\mathcal{C}(.)$. One popular definition is to equate $\mathcal{R}(v_i)$ to the set of all DTs within class c and $\mathcal{C}(v_i)$ to the set of all DTs outside class c (refer to Fig. 1(a)). This definition was used in [14,15,16]. This is quite restrictive, since it assumes that classes in feature space must be significantly compact (i.e. the minimum distance between any sample in class B to class A is at least the maximum distance between any two samples in class A). This is usually not the case for most real data. Based on this definition, the total number of distance constraints is $L = \Theta(M^3)$, which quickly becomes intractable for reasonably sized datasets. As a result, heuristic pruning measures were taken to

[3] The projection onto B_λ^+ is necessary due to the non-negativity constraint on $\boldsymbol{w}_c$. The $[.]_+$ operator returns a vector whose negative coordinates are truncated to zero.

reduce this number [15,14]; however, it remains $\Theta(M^3)$. A major problem with these measures is their immutability, since relevant constraints that are pruned at the beginning can never be added back to the learning process. Therefore, a need arises for another definition of $\mathcal{R}(.)$ and $\mathcal{C}(.)$ that is less restrictive (i.e. a more general representation of real data) and less computationally demanding.

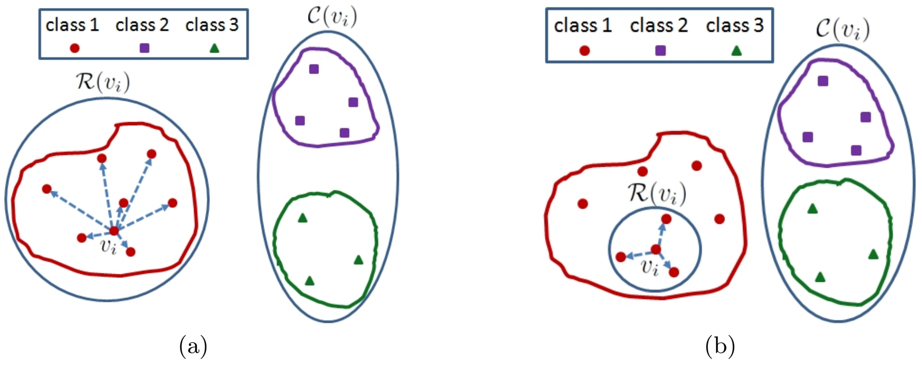


Fig. 1. Shows examples of two definitions for $\mathcal{R}(v_i)$ and their impact on the relative positioning of classes in feature space. For illustration purposes, we assume an L_2 distance is used between features. 1(a) is an example of the definition used in [14,15,16]. 1(b) is an example of the definition used here. Note how the classes need to be more separated (or equivalently more compact) in 1(a) than 1(b).

Although our MMDL method can handle a general structure for $\mathcal{R}(.)$ and $\mathcal{C}(.)$, in this paper, we set $\mathcal{R}(v_i)$ to the k nearest neighbors of v_i within its class. This is based on the intuition that a simple kNN classifier can be easily employed to classify v_i. In this case, STEPS 2-3 in **Algorithm** 1 are equivalent to finding v_i's nearest neighbors according to $\boldsymbol{w}_c^{(t)}$. Note that the value of k need not be the same for every class c. A similar scheme can be applied to set $\mathcal{C}(v_i)$; however, since $M \gg N$ and to avoid overhead computation, we do not compute the nearest neighbors of v_i outside class c. Instead, we simply let $\mathcal{C}(v_i)$ be the set of <u>all DTs</u> outside class c (refer to Fig. 1(b)). Since $k \ll M$, the total number of distance constraints now is $L = \Theta(M^2)$. However, only m out of L constraints are actually used in a single iteration and m is usually much smaller than L. In fact, we show empirical results where the total number of constraints per *DL-PEGASOS* iteration can be reduced to $m = \Theta(M)$, without loss in recognition performance. Since a random set of these relevant constraints is chosen every iteration, the immutability problem facing previous methods is also alleviated. Moreover, the computational complexity of *DL-PEGASOS*, with $\mathcal{R}(.)$ and $\mathcal{C}(.)$ defined as above, is $\Theta\left(T(\frac{2F+k}{N}M + Fm)\right)$, which includes computing and sorting the combined distances $D_{\boldsymbol{w}_c}$. While previous MMDL methods suffer from $\Theta(M^3)$ complexity, our method is at worst $\Theta(M^2)$ and on average $\Theta(M)$.

5 Experimental Results

In this section, we present experimental results that validate the *DL-PEGASOS* algorithm[4] in terms of DT recognition. We first learn class-independent and class-dependent weights for the UCLA benchmark dataset [5]. Realizing that recognition performance on this dataset has saturated and that it lacks DT diversity, a new, easily accessible benchmark is essential. We organize the DynTex++ dataset to be this next benchmark and evaluate our algorithm on it.

5.1 UCLA Dataset

The UCLA dynamic texture dataset contains 50 classes of gray-scale dynamic texture, each of which is comprised of 4 DT sequences. Since these 50 classes contain the same DTs at different viewpoints, they can be grouped together to form 9 classes, as in [9]. Each DT sequence includes 75 frames of 160×110 pixels. Here, the DT sequences are cropped to show the representative dynamics alone, thus, leading to frames of 48×48 pixels.

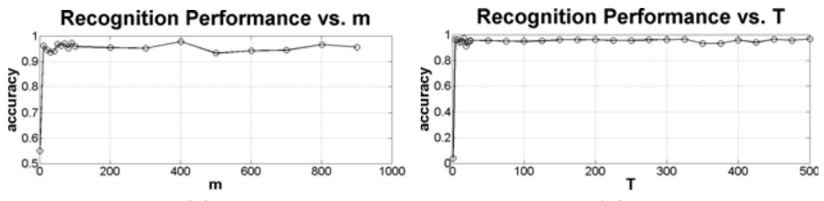
50-class breakdown: In the case of the 50 DT classes, the state-of-the-art recognition result (97.5%) was achieved by using kernel support vector machines (SVM's) [24]. Here, four cross-fold validation was performed, so the training set included $M = 150$ DT sequences (i.e. 3 sequences for each class). Applying *DL-PEGASOS* with $m = 150$ (i.e. $\Theta(M)$) and $T = 25$ iterations, we obtain an average recognition performance of 99% when both class dependent and class independent weights were learned. The class independent weights for the LBP, PHOG, and LDS descriptors are $w_1 = 1.95$, $w_2 = 1.12$, and $w_3 = 1.33$ respectively. This clearly indicates that the discrimination between DTs in this dataset is dominated by their spatial texture features, whereby using these features alone leads to a recognition rate of 90%. This reinforces the conclusion of [7], whose authors also reported on the dominant discriminative power of static texture in the UCLA DT dataset. In what follows, we will evaluate *DL-PEGASOS* on the 9-class breakdown of this dataset, since it poses a greater challenge.

9-class breakdown: In the case of the 9 DT classes, the state-of-the-art recognition result (80%) was achieved by using a bag-of-words model on LDS features [9][5], which lends itself useful to view-invariant recognition. For comparison, we adopt the same experimental setup as in [9]. We train on 50% of the dataset (i.e. $M = 100$) and test on the rest, with the recognition rates recorded as the average rate over 20 trials (i.e. random bisection of the classes in the dataset). First, we study the effect of the *DL-PEGASOS* free parameters (i.e. m and T) on the average recognition performance. Fig. 2(a) plots the recognition rate of class independent *DL-PEGASOS* when m is varied, while T is fixed to 25 iterations. Since $k = 1$, the total number of distance constraints is about 7000, from

[4] All experiments were executed using MATLAB 7.6 on a 2.4 GHz, 4GB RAM PC. Some *DL-PEGASOS* parameters were kept constant: **(i)** $k = 1$ nearest neighbors for $\mathcal{R}(.)$ and **(ii)** $\lambda = 0.05$.

[5] In [9], only 8 classes were considered, since the "plants" class was removed.

which m distance constraints are randomly chosen at each iteration. It is evident that recognition rate very quickly stabilizes ($\sim 95\%$), thus, indicating that most distance constraints do not play a significant role in discriminating between DT classes. This seems intuitive, since most constraints are easily satisfied for DTs that are significantly different in DT feature space. We also conclude that m can be reduced to $\Theta(M)$, without loss of performance. Similarly, Fig. 2(b) plots the recognition rate as T is increased, while m is fixed to 100. Clearly, the stable rate ($\sim 95\%$) is reached in a very small number of iterations.



(a) (b)

Fig. 2. Plots the recognition performance of *DL-PEGASOS* versus m (the number of distance constraints per iteration) and T (the maximum number of iterations) when class dependent weights are learned. To obtain the recognition rates in 2(a), we use $T = 25$. To obtain the recognition rates in 2(b), we use $m = 100$.

By setting $m = 100$ and $T = 25$, we obtain an average recognition rate of 95.6%, which significantly outperforms the state-of-the-art (80%) on this dataset. Fig. 3 shows the average confusion matrix for this experiment. The confused classes tend to have very similar appearance and/or dynamics, especially "fire" + "smoke", "flowers" + "plants" and "fountains" + "waterfall". In regards to time complexity, each complete trial ran in under 0.6 seconds. This time <u>does not</u> include feature extraction or pairwise elementary distance computation.

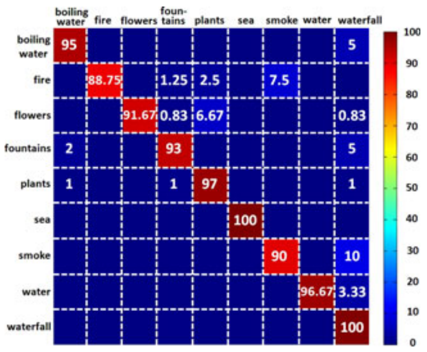


Fig. 3. Shows the confusion matrix for the 9-class experiment

Here, we mention that the recognition performance of class dependent *DL-PEGASOS* (82%) is significantly less than the class independent performance above. This is indicative of overfitting due to the small number of DTs per class. However, it is worthwhile to examine the values of $\boldsymbol{w}_c$, since they shed light on which DT dimension(s) are the most discriminative for a given class. From the weights in Table 1, we notice that some of our intuitions about what discriminates certain DTs are validated. For example, classes defined primarily by their spatial texture appearance (e.g. "flowers", "plants", and "sea") have

Table 1. Class dependent weights for the 9-class recognition experiment

	boiling water	**fire**	**flowers**	**fountains**	**plants**	**sea**	**smoke**	**water**	**waterfall**
w_1 (LBP)	0.21	1.22	**10.58**	0.12	**2.95**	**6.27**	4.23	7.13	4.73
w_2 (PHOG)	**7.81**	0.17	1.06	0.83	0.19	2.95	1.99	1.61	0.93
w_3 (LDS)	7.31	**7.07**	1.45	**10.18**	0.14	1.08	5.93	4.70	7.12

dominant w_1 values. Other classes that are primarily defined by their motion have dominant w_3 values (e.g. "fire" and "fountains"). Interestingly, the "boiling water" class is the only class where w_2 is the largest weight. This is due, in part, because the spatial texture is irregular and highly varying over time, while the overall layout remains stable. The other classes rely on a combination of these dimensions for their discriminative power.

5.2 DynTex++ Dataset

As mentioned before, the UCLA dataset is currently the benchmark for DT recognition, even though a much larger and more diverse datasets (the DynTex dataset [25]) exists. The UCLA dataset remains the benchmark due to the following reasons. **(i)** Its DT sequences have already been pre-processed from their raw form, whereby each sequence is cropped to show its representative dynamics in absence of any static or dynamic background. **(ii)** Only a single DT is present in each DT sequence. **(iii)** In each DT sequence, no panning or zooming is performed. **(iv)** Ground truth labels of the DT sequences are provided. Although some researchers have applied their recognition algorithms on the DynTex dataset (e.g. [20]), it is difficult to manage/use because it lacks the above four properties, in its present form. Therefore, we propose the compilation of a new dataset, called DynTex++.

Compiling the DynTex++ Dataset: The goal here is to organize the raw data in the DynTex dataset in order to provide a richer benchmark that is publicly available (http://vision.ai.uiuc.edu/~bghanem2/DynTex++.htm) for future DT analysis, in the same way the UCLA dataset is currently. The original dataset is already publicly available ($\sim 2GB$ of data); however, only the raw AVI videos are provided. We proceeded to filter, pre-process, and label these DT sequences. While DynTex contains a total of 656 video sequences, DynTex++ uses only 345 of them. We eliminated sequences that contained more than one DT, contained dynamic background, included panning/zooming, or did not depict much motion. The remaining sequences were then hand labeled as one of $N = 36$ classes (e.g. "flying birds", "waterfall", "vehicle traffic"). They were not uniformly distributed among the N classes. We preprocessed them so each class contained the same number of subsequences.

The preprocessing proceeded as follows: **(i)** Each sequence is spatially downsampled by a factor of 0.75 and converted to grayscale. **(ii)** Since it is infeasible to manually crop these sequences, we randomly selected a large (1000) set of subsequences of fixed size ($50 \times 50 \times 50$), each of which is attributed a relevance score that represents how much motion it entails. This score is the average optical flow [26] energy in the subsequence. By doing this, static background subsequences are eliminated from consideration and the more relevant DT subsequences remain. **(iii)** From each class, we selected 100 subsequences with the highest scores (uniformly chosen from the sequences constituting this class), thus, resulting in a dataset of $M = 3600$ subsequences. For more details on DynTex++, refer to the **supplementary material**.

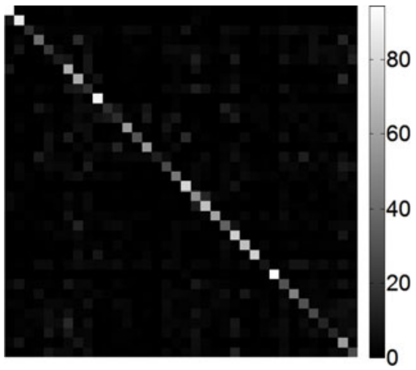


Fig. 4. shows the confusion matrix for DT recognition on DynTex++

***DL-PEGASOS* on DynTex++:** We apply our approach to the DynTex++ dataset, using an experimental setup similar to the one in the 9-class experiment on the UCLA dataset. In this case, we set $m = 2000$ and $T = 100$. We obtain an average recognition rate of 63.7%, with the average confusion matrix shown in Fig. 4. Each trial took under 15 seconds to run to completion.

6 Conclusions and Acknowledgments

In this paper, we formulate DT recognition in a maximum margin distance learning framework, where the distance between two DTs is a linear combination of three elementary distances representing DT space. These distance weights are efficiently learned by our proposed *DL-PEGASOS* algorithm, whose computational complexity is linear in the number of training samples. We validated our approach by outperforming the state-of-the-art on the UCLA benchmark, as well as, applying it the newly compiled DynTex++ dataset. The support of the Office of Naval Research under grant N00014-09-1-0017 and the National Science Foundation under grant IIS 08-12188 is gratefully acknowledged.

References

1. Shalev-Shwartz, S., Singer, Y., Srebro, N.: Pegasos: Primal estimated sub-gradient solver for svm. In: ICML (2007)
2. Ghanem, B., Ahuja, N.: Extracting a fluid dynamic texture and the background from video. In: CVPR (2008)
3. Soatto, S., Doretto, G., Wu, Y.N.: Dynamic textures. IJCV 51, 91–109 (2003)
4. Chetverikov, D., Peteri, R.: A brief survey of dynamic texture description and recognition. In: International Conference on Computer Recognition Systems (2005)
5. Saisan, P., Doretto, G., Wu, Y.N., Soatto, S.: Dynamic texture recognition. In: CVPR, pp. 58–63 (2001)
6. Martin, R.J.: A metric for arma processes. IEEE Trans. on Signal Processing 48, 1164–1170 (2000)

7. Chan, A.B., Vasconcelos, N.: Probabilistic kernels for the classification of auto-regressive visual processes. CVPR 1, 846–851 (2005)
8. Woolfe, F., Fitzgibbon, A.W.: Shift-invariant dynamic texture recognition. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3952, pp. 549–562. Springer, Heidelberg (2006)
9. Ravichandran, A., Chaudhry, R., Vidal, R.: View-invariant dynamic texture recognition using a bag of dynamical systems. In: CVPR, pp. 1651–1657 (2009)
10. Peteri, R., Chetverikov, D.: Dynamic texture recognition using normal flow and texture regularity. In: Iberian Conference on Pattern Recognition and Image Analysis (2005)
11. Ojala, T., Pietikainen, M., Maenpaa, T.: Multiresolution gray scale and rotation invariant texture analysis with local binary patterns. TPAMI 24, 971–987 (2002)
12. Zhao, G., Pietikainen, M.: Local binary pattern descriptors for dynamic texture recognition. In: ICPR, vol. 2, pp. 211–214 (2006)
13. Zhao, G., Pietikainen, M.: Dynamic texture recognition using local binary patterns with an application to facial expressions. TPAMI 29, 915–928 (2007)
14. Frome, A., Singer, Y., Sha, F., Malik, J.: Image retrieval and classification using local distance functions. In: NIPS (2006)
15. Frome, A., Singer, Y., Sha, F., Malik, J.: Learning globally-consistent local distance functions for shape-based image retrieval and classification. In: ICCV (2007)
16. Gu, C., Lim, J.J., Arbelaez, P., Malik, J.: Recognition using regions. In: CVPR, pp. 1030–1037. IEEE, Los Alamitos (2009)
17. Varma, M., Ray, D.: Learning the discriminative power-invariance trade-off. In: ICCV (2007)
18. Vedaldi, A., Gulshan, V., Varma, M., Zisserman, A.: Multiple kernels for object detection. In: ICCV (2009)
19. Rubner, Y., Tomasi, C., Guibas, L.J.: A metric for distributions with applications to image databases. In: ICCV (1998)
20. Zhao, G., Pietikainen, M.: Dynamic texture recognition using volume local binary patterns. In: ECCV, Workshop on Dynamical Vision, pp. 12–23 (2006)
21. Wang, X., Han, T.X., Yan, S.: An hog-lbp human detector with partial occlusion handling. In: ICCV (2009)
22. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR, pp. 886–893 (2005)
23. Bosch, A., Zisserman, A., Munoz, X.: Representing shape with a spatial pyramid kernel. In: CIVR (2007)
24. Chan, A.B., Vasconcelos, N.: Classifying video with kernel dynamic textures. In: CVPR, vol. 1 (2007)
25. Peteri, R., Huiskes, M., Fazekas, S.: DynTex: the Centre for Mathematics and Computer Science (CWI), Amsterdam (2006), `http://www.cwi.nl/projects/dyntex/`
26. Gautama, T., Hulle, M.A.V.: A phase-based approach to the estimation of the optical flow field using spatial filtering. TNN 13, 1127–1136 (2002)