Kernelized Similarity Learning and Embedding for Dynamic Texture Synthesis

Shiming Chen, Peng Zhang, Xinge You, *Senior Member, IEEE*, Qinmu Peng, *Member, IEEE*, Xin Liu, *Member, IEEE*, Zehong Cao, *Member, IEEE* and Dacheng Tao, *Fellow, IEEE*

***Abstract*—Dynamic texture (DT) exhibits statistical stationarity in the spatial domain and stochastic repetitiveness in the temporal dimension, indicating that different frames of DT possess high similarity correlation that is a critical prior knowledge. However, existing methods cannot effectively learn a promising synthesis model for high-dimensional DT from a small number of training data. In this paper, we propose a novel DT synthesis method, which makes full use of similarity prior knowledge to address this issue. Our method bases on the proposed kernel similarity embedding, which not only can mitigate the high-dimensionality and small sample issues, but also has the advantage of modelling nonlinear feature relationship. Specifically, we first raise two hypotheses that are essential for DT model to generate new frames using similarity correlation. Then, we integrate kernel learning and extreme learning machine into a unified synthesis model to learn kernel similarity embedding for representing DT. Extensive experiments on DT videos collected from internet and two benchmark datasets, i.e., Gatech Graphcut Textures and Dyntex, demonstrate that the learned kernel similarity embedding can effectively exhibit the discriminative representation for DT. Accordingly, our method is capable of preserving long-term temporal continuity of the synthesized DT sequences with excellent sustainability and generalization. Meanwhile, it effectively generates realistic DT videos with fast speed and low computation, compared with the state-of-the-art methods. The code and more synthesis videos are available at our project page https://shiming-chen.github.io/Similarity-page/Similarit.html.**

***Index Terms*—Dynamic texture (DT), kernel similarity embedding, extreme learning machine, similarity prior knowledge.**

I. Introduction and Motivation

DYNAMIC texture, exhibiting statistical stationarity in the spatial domain and stochastic repetitiveness in the temporal dimension, is one of the dynamic patterns in computer vision [1], [2], e.g., moving vehicles, falling water, flaming fire, rotating windmill. Due to the demands of dynamic patterns synthesis in video technology applications (e.g., texture recognition [3], video segmentation [4], [5] and super-resolution [6]), synthesizing DTs has gradually become an interesting topic in computer graphic and computer vision [7]–[12]. DT synthesis aims to infer a generating process from a DT example, which then allows producing an infinitely varying stream of similar looking texture videos.

In general, DT synthesis methods can be categorized into two groups: non-neural-networks-based methods and neural-networks-based methods. The first group methods are popular approaches for DT synthesis in the early stage, and it can be further classified as physics-based methods [13]–[15], nonparametric methods [16], [17] and dynamic system (DS) modeling methods [1], [8], [18]–[20]. The second group methods automatically learn the texture distribution with effective representation of neural networks [7], [9], [11], [12], [21], [22]. However, DT is high-dimensional data and lacks enough samples (DT sample typically only has one single training video with a short length of sequence). The non-neural-network-based methods usually seek to reduce the dimensionality of DT for modeling, which may account for information loss and it is hard to design a proper dimensionality-reduction algorithm. Meanwhile, neural-network-based methods fail to effectively fit their large number of parameters when learning from a small number of training samples. To overcome these challenges, we propose a novel insight for DT synthesis that prior knowledge is mined and exploited, i.e., similarity prior knowledge.

In fact, the similarity correlation between frame-to-frame is an explicit expression of statistical stationarity and stochastic repetitiveness of DT. It is a critical representation to distinguish DTs from other videos. Similarity representation serves as the learning objective of metric learning for the discriminative model [3], [23], [24], which suggests the importance of similarity correlation for representation. Some researchers also attempted to mine the potential similarity knowledge of samples to improve the performance of the discriminative model [25]–[31], which suggests that similarity correlation is critical prior knowledge as important as the class labels and other annotation information. Moreover, the similarity correlation can explicitly capture the homogeneous and heterogeneous correlation between different frames of DT. However, to the best of our knowledge, there are no studies on DT synthesis to consider the similarity prior knowledge to address the high-dimensionality and small sample issues, and this is the focus

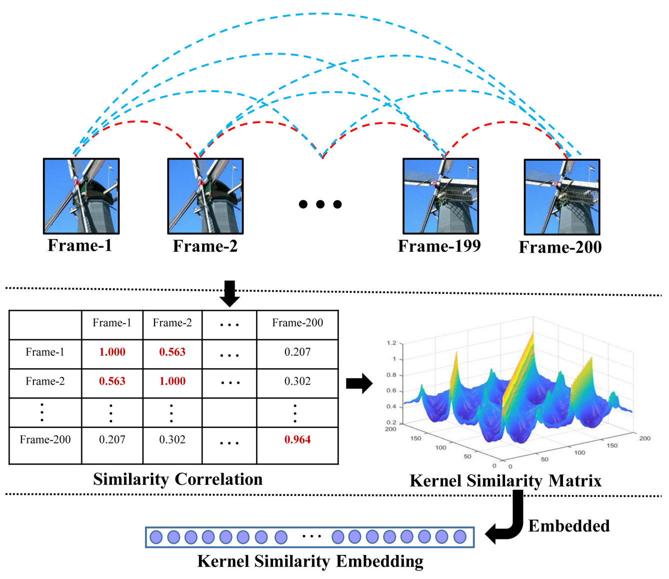
This work was supported in part by the National Natural Science Foundation of China (61571205 and 61772220), the Key Program for International S&T Cooperation Projects of China (2016YFE0121200), the Special Projects for Technology Innovation of Hubei Province (2018ACA135), the Key Science and Technology Innovation Program of Hubei Province (2017AAA017), the Natural Science Foundation of Hubei Province (2018CFB691), fund from Science, Technology and Innovation Commission of Shenzhen Municipality (JCYJ20180305180804836 and JSGG20180507182030600). (Corresponding author: Xinge You.)

S. Chen, P. Zhang, X. You and Q. Peng are with the Department of Electronic Information and Communication, Huazhong University of Science and Technology, Wuhan 430074, China. (e-mail:gchenshiming@gmail.com; youxg@hust.edu.cn; pengqinmu@hust.edu.cn).

X. Liu is with the Department of Computer Science and Technology, Huaqiao University, Xiamen 361021, China (e-mail: xliu@hqu.edu.cn).

Z. Cao is is with the Discipline of ICT, University of Tasmania, TAS 7001, Australia (e-mail:zehong.cao@utas.edu.au).

D. Tao is with the UBTECH Sydney Artificial Intelligence Centre, Faculty of Engineering and Information Technologies, School of Information Technologies, The University of Sydney, Darlington, NSW 2008, Australia (e-mail:dacheng.tao@sydney.edu.au).



	Frame-1	Frame-2	⋯	Frame-200
Frame-1	1.000	0.563	⋯	0.207
Frame-2	0.563	1.000	⋯	0.302
⋮	⋮	⋮		⋮
Frame-200	0.207	0.302	⋯	0.964

Fig. 1: The core idea of the proposed method. DT exhibits statistical stationarity in the spatial domain and stochastic repetitiveness in the temporal dimension, indicating that different frames of DT possess high similarity correlation. Meanwhile, this correlation can be expressed by kernel similarity matrix and embedded into kernel similarity embedding.

of the present paper.

To make full use of similarity prior knowledge, we embed it into the representation of the generative model for DT synthesis. Thus, we raise two hypotheses: 1) the content of texture video frames varies over time-to-time while the more closed frames should be more similar, and 2) the transition between frame-to-frame can be modeled as a linear or nonlinear function to capture the similarity correlation. These hypotheses are essential for the DT model to generate new frames according to current frames using a similarity correlation of different frames. Fortunately, kernel function implicitly embraces an exciting property that it can elegantly represent the similarity of two inputs [32]. Thus our core idea is that the statistical stationarity in the spatial domain and the stochastic repetitiveness in the temporal dimension of DTs can be partially captured by similarity correlation between frame-to-frame. This correlation can be further elegantly exhibited by kernel similarity matrix that is embedded into kernel similarity embedding for representation, as demonstrated in Figure 1. Furthermore, extreme learning machine (ELM) as an emergent technology which overcomes some challenges faced by other computational intelligence techniques, and it has recently attracted the attention of more researchers [33]–[37]. Therefore, we attempt to make full advantage of ELM and jointly utilize kernel learning to learn kernel similarity embedding for improving DT synthesis.

In this work, we propose a novel DT synthesis method to generate high-quality, long-term DT sequences with fast speed and low computation. It integrates kernel learning and ELM into a powerfully unified synthesis model to learn kernel similarity embedding for representing statistical stationarity in the spatial domain and stochastic repetitiveness in the temporal dimension of DT sequences. Specifically, we preprocess every input DT sequence S_N (N is the length of DT sequence), which is divided into two parts: explanatory frames and response frames. Then, our method uses kernel function to replace the feature mapping function of the hidden layer of extreme learning machine, and thus the kernel similarity embedding is easily learned after training. Finally, the DT sequence is iteratively generated via the trained model of our method.

To summarize, this study makes the following salient contributions:

- We raise two efficient hypotheses to benefit the DT system, which are essential for the DT system to generate new frames according to current frames using similarity correlation of different frames.
- We propose a novel DT synthesis method, which learns kernel similarity embedding to synthesize realistic video sequences with good sustainability and generalization.
- We introduce kernel similarity embedding to mine and exploit similarity prior knowledge of DT and analyze its availability in intuitive and theoretical insights.
- We carry out extensive experiments on benchmark datasets to demonstrate that our method shows consistent improvement over the state-of-the-art methods.

The remainder of this paper is organized as follows. Section II provides an overview of the background and related works of DT synthesis. The proposed method based on kernel similarity embedding is elaborated in Section III. The performance and evaluation are given in Section IV. Section V presents the discussion. Section VI provides a summary and the outlook for future research.

II. Background and Related Work

The goal of DT synthesis is to generate an infinitely varying stream of similar appearance texture videos. It aims to accurately learn a transition function f from the input texture sequence $\{y_t\}_{t=1,\ldots,N} \in R^m$ (N is the length of sequence, m is the dimensionality of frame) of training set, which can be formulated as:

$$y_t = f\left(y_{t-1}\right) \tag{1}$$

A straightforward way to learn f is to solve the following objective function:

$$f' = \underset{f}{\operatorname{argmin}} \frac{1}{2} \sum_{t=2}^{N} \left(y_t - f\left(y_{t-1}\right)\right)^2 \tag{2}$$

After training, f' can be learnt. Subsequently, given an initial frame y'_{t-1} from an input texture sequence of test set, the endless sequences $\{y'_t\}_{t=2,3,\ldots}$ can be generated iteratively by:

$$y'_t = f'\left(y'_{t-1}\right) \tag{3}$$

In the following, we provide a comprehensive review of related work of DT synthesis based on various methods.

A. Non-Neural-Network-Based Methods

1) Physics-Based Methods: Physics-based methods describe DT by simulating its physical mechanism using complicate models. In [13], Pegoraro and Parker presented a new method for the physically-based rendering of frames from detailed simulations of frame dynamics, which accounts for their unique characteristics. This method can synthesize highly realistic renderings of various types of frames. Nealen et al. [14] proposed a physically based deformable model for DT synthesis and made a connection to the simulation of various DT, e.g., fluids, gases, and melting objects. Higher-order SVD (HOSVD) analysis for DT synthesis is proposed in [15]. It decomposes the DT as a multi-dimensional signal without unfolding the video sequences on column vectors, and thus it allows us to perform dimensionality-reduction in the spatial, temporal, and chromatic domain. In summary, although physics-based methods can generate an impressive DT, they are highly application-specific with weak generalization.

2) Dynamic System Modeling Methods: Dynamic system (DS) modeling methods for DT synthesis are the most popular non-neural-network-based method. DS modeling methods typically learn transition function for representing the correlation of different frames of DTs using linear or nonlinear dimensionality-reduction algorithms. A proper dimensionality-reduction algorithm is hard to design, and it may account for information loss of DT. These are the main limitations of DS modeling methods. In [1], Doretto et al. proposed pioneering DS method for DT synthesis using a simple linear dynamic system (LDS) to project the input video frames into lower dimensional space by singular value decomposition (SVD). Siddiqi et al. [20] proposed a stable-LDS (SLDS) based method to add constraints to a relaxed system solution incrementally and to improve stability. To better adapt the standard LDS-based method to memory- and computational power-limited devices, Abraham proposed new DT synthesis with Fourier descriptors (FFT-LDS) [19], which requires far fewer parameters for DT synthesis compared to standard LDS approaches. In [18], Chain and Vasconcelos introduced a new method (Kernel-DT) for DT synthesis using kernel principal component analysis (KPCA) to learn a nonlinear observation function. There is an essential problem existing in the aforementioned DS modeling methods, in which the column vector dimension of the unfolded frame is often too large compared to the number of given texture frames. To address this problem, the kernel principal component regression (KPCR) method was proposed for DT synthesis by You [8].

B. Neural-Network-Based Methods

The neural network has proven to be an immensely successful discriminative and generative learning machine [23], [38]–[40]. In term of DT synthesis, various approaches based on the ConvNet have been proposed [7], [9], [11], [12], [22]. In [7], Gatys et al. introduced a new model of DT synthesis based on the feature spaces of convolutional neural networks that represent texture using the correlations between feature maps in several layers. Motivated by the works on style transfer and enabled by the two-stream model, Tesfaldet et al. proposed a two-stream model for DT synthesis [11]. This method represents the textures appearance and dynamic of DT using a set of Gram matrices. [21] presented the motion and content decomposed generative adversarial networks (MoCoGAN) for video generation. MoCoGAN is good at generating DTs when learning from a large number of training data. In [9], [12], Xie er al. proposed an energy-based spatial-temporal generative ConvNet to model and to synthesize dynamic patterns. This model is beneficial for generating realistic dynamic patterns when the incomplete input sequences with either occluded pixels or missing frames. [22] presented a dynamic generator model using alternating back-propagation through time algorithm for DT synthesis. This model is efficient in terms of computational cost and model parameter size because it does not need to recruit a discriminative network or an inference network. In summary, neural-neural-based methods fail to exhibit their powerful representation because DT is high-dimensional data and lack of enough samples. Meanwhile, they are time-consuming and computationally expensive for learning their large number of parameters. Therefore, an extreme learning machine may be the desired successor for DT synthesis with expected generalization performance at a surprising learning speed.

III. The Proposed DT Synthesis Method

Similarity is a key prior knowledge existing in different samples or self-sample as important as the class labels and other annotation information. Thus, some researchers recently consider to make use of the similarity knowledge of samples to improve the performance of the discriminative model in various tasks, i.e., person re-identification (re-ID) [25]–[27], content-based image retrieval [28]–[31]. DT exhibits statistical stationarity in the spatial domain and stochastic repetitiveness in the temporal dimension, indicating that different frames of DT possess a high similarity correlation. These similarity correlations can also be viewed as the critical prior knowledge, which may mitigate the high-dimensionality and small sample issues for DT synthesis. Therefore, we propose a novel DT synthesis method. It is recommended to synthesize desirable DTs at fast synthesis speed using kernel similarity embedding, which bases on ELM and kernel learning.

In this section, we first revisit ELM for conveniently understanding the proposed method. We then illustrate our method that uses ELM based kernel similarity embedding. Meanwhile, we introduce the additive regularization factor to smooth kernel similarity embedding, and thus our method will be stabler and tend to have better generalization. Finally, we intuitively and theoretically analyze the mechanism of how and why our method can generate realistic, long-term DT videos.

A. Revisiting Extreme Learning Machine

ELM has initially been proposed by Huang et al. [33], and it serves as an emergent technology that has recently attracted much attention [33]–[37]. ELM works for generalized single-hidden layer feedforward networks (SLFNs). Its essence is that the hidden layer of SLFNs need not be tuned, which means that the feature mapping between the input layer and hidden layer is randomly assigned. With better generalization performance, ELM overcomes some challenges (e.g., slow learning speed, trivial human intervene and poor computational scalability) faced by other computational intelligence techniques. Moreover, the parameters of the hidden layer are randomly initialized during training, and then the weights of the output layer are learned. Therefore, we take full advantage of ELM to expedite the generated speed for DT synthesis with good generalization.

Before introducing ELM formally, we define the notations. Given a dataset $T = \{(\mathbf{x}_1, \mathbf{y}_1), \cdots, (\mathbf{x}_N, \mathbf{y}_N)\}$, where $\mathbf{x}_i \in R^n$, $\mathbf{y}_i \in R^m$, $i = 1, ..., N$. The model of ELM with L hidden nodes can be formulated as:

$$f_L(x) = \sum_{i=1}^{L} \beta_i h_i(\mathbf{x}) = \mathbf{h}(x)\boldsymbol{\beta} \tag{4}$$

where $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_L]^\top$ is the vectors of output weights between the hidden layer and the output layer, $\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}), \ldots, h_L(\mathbf{x})]$ is the output vector of the hidden layer with respect to the input $\mathbf{x}$, and L is the number of nodes of hidden layer. Intuitively, $\mathbf{h}(\mathbf{x})$ is a feature mapping, it maps the input $\mathbf{x}$ from n-dimensional input space to the L-dimensional hidden layer feature space (ELM feature space) $\boldsymbol{H}$.

According to Bartlett's theory [41], the smaller the norm of weights are, the better generalization performance networks tend to achieve, while networks reaches smaller training error. See work [42], minimizing the norm of the output weights is actually to maximize the distance of the separating margins of different domains in the feature space. Therefore, different from traditional intelligent learning algorithms, ELM is to minimize the training errors and the norm of the output weights simultaneously. That is shown in Eq. (5).

$$Minimize : \|\boldsymbol{H}\boldsymbol{\beta} - \boldsymbol{Y}\|^2 \quad and \quad \|\boldsymbol{\beta}\| \tag{5}$$

where $\boldsymbol{H}$ is output matrix of the hidden layer, shown in Eq. (6).

$$\boldsymbol{H} = \begin{bmatrix} \mathbf{h}(\mathbf{x}_1) \\ \vdots \\ \mathbf{h}(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} h_1(\mathbf{x}_1) & \cdots & h_L(\mathbf{x}_1) \\ \vdots & \vdots & \vdots \\ h_1(\mathbf{x}_N) & \vdots & h_L(\mathbf{x}_N) \end{bmatrix} \tag{6}$$

To solve Eq. (5), the minimal norm least square method is typically used, and the solution is written as Eq. (7).

$$\boldsymbol{\beta} = \boldsymbol{H}^\dagger \boldsymbol{Y} \tag{7}$$

where $\boldsymbol{H}^\dagger$ is the Moore-Penrose generalized inverse of $\boldsymbol{H}$, $\boldsymbol{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_n] \in \mathbb{R}^{m\times n}$. There are different methods can be used to calculate the Moore-Penrose generalized inverse of $\boldsymbol{H}$, e.g., orthogonalization method, orthogonal projection method, and singular value decomposition. Here we use the orthogonal projectional method, which can be used in two cases: 1) if $\boldsymbol{H}^\top\boldsymbol{H}$ is nonsingular, $\boldsymbol{H}^\dagger = \left(\boldsymbol{H}^\top\boldsymbol{H}\right)^{-1}\boldsymbol{H}^\top$, or 2) if $\boldsymbol{H}\boldsymbol{H}^\top$ is nonsingular, $\boldsymbol{H}^\dagger = \boldsymbol{H}^\dagger\left(\boldsymbol{H}\boldsymbol{H}^\top\right)^{-1}$.

Therefore, Eq. (7) can be rewritten as Eq. (8) or Eq. (9).

$$\boldsymbol{\beta} = \boldsymbol{H}^\top\left(\boldsymbol{H}\boldsymbol{H}^\top\right)^{-1}\boldsymbol{Y} \tag{8}$$

$$\boldsymbol{\beta} = \left(\boldsymbol{H}^\top\boldsymbol{H}\right)^{-1}\boldsymbol{H}^\top\boldsymbol{Y} \tag{9}$$

Finally, the model of ELM can be written as Eq. (10) or Eq. (11).

$$f(x) = \mathbf{h}(x)\boldsymbol{\beta} = \mathbf{h}(x)\boldsymbol{H}^\top\left(\boldsymbol{H}\boldsymbol{H}^\top\right)^{-1}\boldsymbol{Y} \tag{10}$$

$$f(x) = \mathbf{h}(x)\boldsymbol{\beta} = \mathbf{h}(x)\left(\boldsymbol{H}^\top\boldsymbol{H}\right)^{-1}\boldsymbol{H}^\top\boldsymbol{Y} \tag{11}$$

Note that, the size of $\boldsymbol{H}\boldsymbol{H}^\top$ is $N \times N$, the size of $\boldsymbol{H}^\top\boldsymbol{H}$ is $L \times L$. Indeed, $N < L$ in the field of DT synthesis. From practical point of view, we get the solution of ELM based on Eq. (10) in following section.

B. Kernel Similarity Embedding for DT Synthesis

Presenting from the revisiting in Section III-A, we know that feature mapping $\mathbf{h}(\mathbf{x})$ is crucial for ELM. However, $\mathbf{h}(\mathbf{x})$ is known to the user and selected artificially, which is similar to the selection of dimensionality-reduction function of physics-based methods and DS modeling methods for DT synthesis. Moreover, the nodes L of the hidden layer of ELM are typically more than the dimensions of input data. It means that feature mapping function $\mathbf{h}(\mathbf{x})$ explicitly maps samples to high dimensional space, which is equivalent to the original idea of the kernel function. Furthermore, kernel function possesses an exciting property that can effectively measure the similarity of different samples, which can elegantly exhibit the similarity correlation between different frames for DT. Therefore, we extend ELM to kernel-ELM by kernel learning (kernel function: $K(\boldsymbol{u}, \boldsymbol{v})$) for DT synthesis.

The architectural overview of our method is shown in Figure 2. Before training, all input DT sequences substrate their temporal mean, $\boldsymbol{S}_t \leftarrow \boldsymbol{S}_t - \overline{S}$, and the input video sequence is divided into two sub-sequences: explanatory frames and response frames. During training, we use kernel-ELM to learn the kernel similarity matrix for representing the statistical stationarity in the spatial domain and the stochastic repetitiveness in the temporal dimension of DT. The kernel similarity matrix will be further embedded into kernel similarity embedding for representation. During testing, the high-fidelity long-term DT sequence is synthesized iteratively using kernel similarity embedding.

At first, we define a kernel similarity matrix Ω_{KSM}:

$$\boldsymbol{\Omega}_{KSM} = \boldsymbol{H}\boldsymbol{H}^\top \tag{12}$$

and

$$\Omega_{KSM_{i,j}} = \mathbf{h}(\mathbf{x}_i) \cdot \mathbf{h}(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j) \tag{13}$$

In fact, kernel-ELM shares a similar network structure with ELM and optimizes output weights $\boldsymbol{\beta}$ of ELM using kernel function $K(u; v)$ for learning kernel similarity embedding. Therefore, kernel-ELM is easier to learn a model than ELM and keeps the merits of ELM. According to ridge regression theory [43], we can add a regularization factor λ (positive small value) to control the regularization performance of $\|\boldsymbol{\beta}\|$ during optimization, which is dissimilar to [34] that regularizes the term $\|\boldsymbol{H}\boldsymbol{\beta} - \boldsymbol{Y}\|^2$. If a proper λ is used, the kernel similarity matrix will be more smooth, and thus our method will be stabler

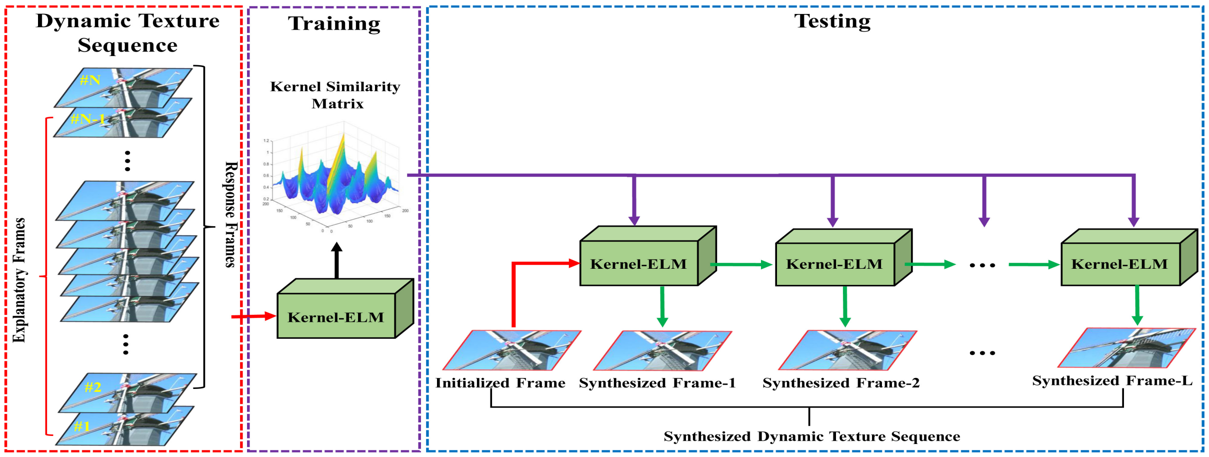


Fig. 2: The architectural overview of our proposed DT synthesis method.

and tend to have better generalization performance. Then, the optimization objective of our method can be formulated as:

$$\begin{aligned} &Minimize: L = \frac{1}{2}\lambda\|\boldsymbol{\beta}\|^2 + \frac{1}{2}\sum_{i=1}^{N}\|\boldsymbol{\xi}_i\|^2 \\ &s.t. \quad \mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta} - \mathbf{Y}_i^\top = \boldsymbol{\xi}_i^\top \end{aligned} \tag{14}$$

where $i = 1, \dots, N$ (N is the number of training frames), $\boldsymbol{\xi}_i = [\xi_{i,1}, \dots, \xi_{i,m}]^\top$ is the training error vector of the training sample x_i. According to the Lagrange theorem, training our method is equivalent to solve the following optimization object:

$$\begin{aligned} L = &\frac{1}{2}\lambda\|\boldsymbol{\beta}\|^2 + \frac{1}{2}\sum_{i=1}^{N}\|\boldsymbol{\xi}_i\|^2 \\ &- \sum_{i=1}^{N}\sum_{j=1}^{m}\alpha_{i,j}\left(\mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta}_j - Y_{i,j} + \xi_{i,j}\right) \end{aligned} \tag{15}$$

where $\boldsymbol{\beta}_j$ is the vector of the weights that links hidden layer to the jth output node of output layer and $\boldsymbol{\beta} = [\boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}_m]$, $\alpha_{i,j}$ is Lagrange multiplier corresponding to the jth output of ith training sample. Then, we have the following KKT corresponding optimality conditions:

$$\frac{\partial L}{\partial \boldsymbol{\beta}_j} = 0 \to \lambda\boldsymbol{\beta}_j = \sum_{i=1}^{N}\alpha_{i,j}\mathbf{h}(\mathbf{x}_i)^\top \to \boldsymbol{\beta} = \frac{1}{\lambda}\boldsymbol{H}^\top\boldsymbol{\alpha} \tag{16}$$

$$\frac{\partial L}{\partial \boldsymbol{\xi}_i} = 0 \to \boldsymbol{\alpha}_i = \boldsymbol{\xi}_i \tag{17}$$

$$\frac{\partial L}{\partial \boldsymbol{\alpha}_i} = 0 \to \mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta} - \mathbf{Y}_i^\top + \boldsymbol{\xi}_i^\top = 0 \tag{18}$$

where $\boldsymbol{\alpha}_i = [\alpha_{i,1}, \dots, \alpha_{i,m}]^\top$ and $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_N]^\top$.

Substituting Eq. (16) and Eq. (17) into Eq. (18), which can be written as:

$$\left(\mathbf{I} + \frac{1}{\lambda}\boldsymbol{H}\boldsymbol{H}^\top\right)\boldsymbol{\alpha} = \boldsymbol{Y} \to \boldsymbol{\alpha} = \left(\mathbf{I} + \frac{1}{\lambda}\boldsymbol{H}\boldsymbol{H}^\top\right)^{-1}\boldsymbol{Y} \tag{19}$$

where $\mathbf{I}$ is identity matrix. Then, by combining Eq. (16) and Eq. (19), the output weights $\boldsymbol{\beta}$ of the hidden layer can be formulated as:

$$\begin{aligned} \boldsymbol{\beta} &= \frac{1}{\lambda}\boldsymbol{H}^\top\left(\mathbf{I} + \frac{1}{\lambda}\boldsymbol{H}\boldsymbol{H}^\top\right)^{-1}\boldsymbol{Y} \\ &= \boldsymbol{H}^\top\left(\lambda\mathbf{I} + \boldsymbol{H}\boldsymbol{H}^\top\right)^{-1}\boldsymbol{Y} \end{aligned} \tag{20}$$

Thus, the transition function of our method (output function of kernel-ELM) can be formulated as Eq. (21)) according to Eq. (4) and Eq. (20).

$$f(x) = \mathbf{h}(\mathbf{x})\boldsymbol{H}^\top\left(\lambda\mathbf{I} + \boldsymbol{H}\boldsymbol{H}^\top\right)^{-1}\boldsymbol{Y} \tag{21}$$

By Combing Eq. (1) and Eq. (21), the transition function of our method can be rewritten as Eq. (22).

$$f(x) = \begin{bmatrix} K(\mathbf{x}, \mathbf{x}_1) \\ \vdots \\ K(\mathbf{x}, \mathbf{x}_N) \end{bmatrix}^\top \left(\lambda\mathbf{I} + \boldsymbol{\Omega}_{KSM}\right)^{-1}\boldsymbol{Y} \tag{22}$$

$\mathbf{x}$ is the test frame, $[\mathbf{x}_1, ..., \mathbf{x}_N]$ is respected to the element of explanatory frames, and $\boldsymbol{Y}$ is respected to response frames. See from Eq. (22), we can find that the proposed model is only relevant to kernel function, input data $\mathbf{x}_i$ and the number of training samples. The kernel similarity embedding is not related to the number of outputs nodes. Thus our method needs not to select artificially $\mathbf{h}(\mathbf{x})$ and implicitly maps input data to high dimensional space. Furthermore, feature mapping $\mathbf{h}(\mathbf{x})$ and the dimensionality of the feature space (nodes of hidden layer) is unknown to users; instead, its corresponding kernel $K(\boldsymbol{u}, \boldsymbol{v})$ is given to users. Morever, the Eq. (22) intuitively shows that kernel similarity matrix is embedded into kernel similarity embedding, which will effectively use the similarity prior information for representing DTs.

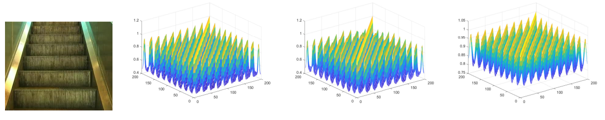
See Algorithm 1 for a description of the proposed DT synthesis method. Specifically, algorithm first divides the input video sequence $\boldsymbol{S}_{1:N}$ (after subtracting temporal mean $\overline{S}$) into two sub-sequences: explanatory frames $\{\boldsymbol{S}_j, j = 1, \cdots, N-1\}$ and response frames $\{\boldsymbol{S}_k, k = 2, \cdots, N\}$. Then, the kernel similarity matrix Ω_{KSM} is learned with respect to Eq. (13) and Eq. (12), in which it will be embedded into kernel similarity

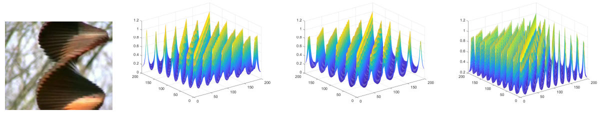
Algorithm 1 Kernel Similarity Embedding for DT Synthesis

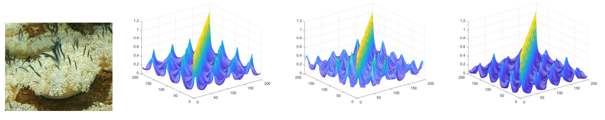
```
Input:
    (1) Training video sequences $\{\boldsymbol{S}_t, t = 1, \cdots, N\}$
    (2) Number of synthesized image sequences $L$
    (3) Kernel function $K(\boldsymbol{u}, \boldsymbol{v})$
Output:
    (1) Synthesized image sequences $\{\tilde{\boldsymbol{S}}_l, l = 1, \cdots, L\}$

 1: Caculate the temporal mean $\overline{S}$ of $\boldsymbol{S}_t$.
 2: Let $\boldsymbol{S}_t \leftarrow \boldsymbol{S}_t - \overline{S}$, $t = 1, \cdots, N$.
 3: Initialize $\{\tilde{\boldsymbol{S}}_l\}$, for $l = 1, \cdots, L$.
 4: Define explanatory frames $\{\boldsymbol{S}_j, j = 1, \cdots, N-1\}$ and
    response frames $\{\boldsymbol{S}_k, k = 2, \cdots, N\}$ using training video
    sequences.
 5: Calculate $\boldsymbol{\Omega}_{KSM}$ and $\left(\lambda\mathbf{I} + \boldsymbol{\Omega}_{KSM}\right)^{-1}\boldsymbol{Y}$ according to
    Eq. (12) and Eq. (13).
 6: repeat
 7:     Calculate $\tilde{\boldsymbol{S}}_l = f(\tilde{\boldsymbol{S}}_{l-1})$ by Eq. (22), $l > 1$.
 8:     Let $\tilde{\boldsymbol{S}}_l \leftarrow \tilde{\boldsymbol{S}}_l + \overline{S}$.
 9:     Let $l \leftarrow l + 1$
10: until $l = L$
```

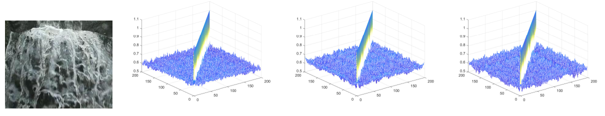
embedding for representing DT. Finally, the endless sequences $\{s'_t\}_{t=2,3,\ldots}$ (after adding temporal mean $\overline{S}$) can be generated iteratively with pre-trained model according to Eq. (22).

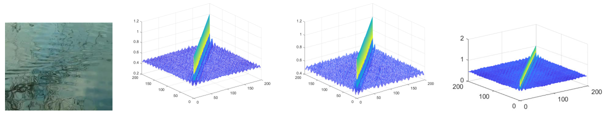
In fact, the dimensionality D of explanatory frames and response frames is equal, and thus $n = m = D$. During training, the computational complexity of our method is $\boldsymbol{O}\left(D^2N^2\right)$, including $N \times N$ kernel operation, an inverse operation, and matrix multiplication. During testing, the computational complexity of our method is $\boldsymbol{O}\left(DN\right)$, including N kernel operation and matrix multiplication.


(a) Elevator


(b) Rotating wind ornament


(c) Flowers swaying with current


(d) Waterfall


(e) Water wave

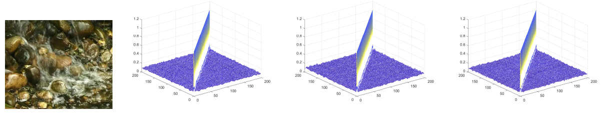

(f) Spring water

Fig. 3: Visualization of kernel similarity matrices of DTs are displayed. For each row, the first image is the frame of a DT video, and the other three are the kernel similarity matrices learned from different DT sequences of the same class.

C. Analysis of Kernel Similarity Embedding

1) Intuitive Insight: DT videos exhibit statistical stationarity in the spatial domain and stochastic repetitiveness in the temporal dimension, which is the key cue for distinguishing DT videos from other videos and static images [1], [7], [11], [44]. Moreover, this cue can be further elaborated as the similarity correlation between frame-to-frame, as shown in Figure 1. Therefore, what we need to do for DT synthesis is that we should build a DT model for presenting the features of dynamics and texture elements, which are statistically similar and temporally stationary. Here we integrate kernel learning and ELM into a powerfully unified DT synthesis model to learn kernel similarity embedding for achieving this goal.

Our method represents such features with a kernel similarity matrix, which is embedded into kernel similarity embedding. To intuitively analyze this mechanism, we visualize the learned kernel similarity matrices of some DT sequences (200 frames for each sequence) in the Dyntex dataset after training, as shown in Figure 3. The kernel similarity matrices of each row in Figure 3 are learned from different DT sequences of the same class, e.g., elevator, waterfall, rotating wind ornament, flowers swaying with current, water wave and spring water, which are existing mostly in everyday surroundings. The different DT sequences of the same class are acquired with different views or different time periods. See from Figure 3, kernel similarity matrices of our method elegantly represent the similarity correlation of DT videos. Specifically, the repetitiveness and stationarity of DT of the elevator, rotating wind ornament and flowers staying with current, are clearly exhibited by the learned matrices. As for waterfall, water wave, and spring water, although these objects originally have not obvious repetitiveness influenced by natural factors, kernel similarity matrices consistently exhibit the statistical stationarity and similarity for different DT videos of the same class. To this end, kernel similarity embedding well mines and exploits the similarity prior knowledge for representing DT using kernel similarity matrix, which will well overcome the challenge of high-dimensionality and small sample issues. That is, the representation features of our method are discriminative, which makes the proposed method can generate high-fidelity, long-term DT videos.

2) Theoretical Insight: In fact, kernel similarity embedding for DT synthesis can be view as the kernel embedding of conditional distribution for regression problem, where the feature vector $\Phi = \left[\phi\left(\mathbf{y}_1\right), \ldots, \phi\left(\mathbf{y}_N\right)\right]^\top$ (Φ is mapping function) in reproducing kernel Hilbert space (RKHS) is substituted by $Y = [\mathbf{y}_1, \ldots, \mathbf{y}_N]^\top$ in original data domain. The key idea of kernel embedding of conditional distribution is to map conditional distributions into infinite-dimensional feature spaces using kernels, such that we can ultimately capture all the statistical features of arbitrary distributions and high-dimensional data [45], [46]. Its formulation is shown as:

$$\begin{aligned}\widehat{\mu}_{\mathbf{Y}|\mathbf{x}} &= \sum_{i=1}^{N} \phi\left(\mathbf{y}_i\right) W_i(\mathbf{x}) = \boldsymbol{W}(\mathbf{x})\Phi \\ &= K_{:\mathbf{x}}(G+\lambda I)^{-1}\Phi\end{aligned} \tag{23}$$

where $K_{:\mathbf{x}} = [k(\mathbf{x}, \mathbf{x}_1), \ldots, k(\mathbf{x}, \mathbf{x}_N)]$, G is the Gram matrix for samples from variable $\boldsymbol{X}$, $\boldsymbol{W}(x) = [W_1, \ldots, W_N]^\top$ is non-uniform weight vector determined by the value $\mathbf{x}$ of the conditioning variable. Indeed, this non-uniform weight vector reflect the effects of conditioning on the embedding. As for kernel similarity embedding, it can be rewritten as Eq. (24) according to Eq. (22).

$$\begin{aligned}f(x) &= \begin{bmatrix} K(\mathbf{x}, \mathbf{x}_1) \\ \vdots \\ K(\mathbf{x}, \mathbf{x}_N) \end{bmatrix}^\top \left(\lambda \mathbf{I} + \boldsymbol{\Omega}_{KSM}\right)^{-1} \boldsymbol{Y} \\ &= \boldsymbol{W}(\mathbf{x})\boldsymbol{Y}\end{aligned} \tag{24}$$

It is obvious that the synthesized frames are conditioned by training sample $\mathbf{x}$. To compare Eq. (23) and Eq. (24), we observed that the kernel similarity embedding for DT is similar to the kernel embedding for conditional distribution. The difference is that kernel similarity embedding for DT synthesis predicts the future frames $\mathbf{y}_i$ in the original data domain, while kernel embedding for conditional distribution predicts the feature $\phi(\mathbf{y}_i)$ in RKHS. To this end, kernel similarity embedding possesses the properties of kernel embedding for conditional distribution. Thus it will well represent the statistical features (e.g., similarity correlation representation) by modeling nonlinear feature relationships for DT.

IV. Experiments and Evaluation

In the following sections, we illustrate the implementation details and parameter setting. Furthermore, we intuitively analyze the sustainability and generalization of our method. Finally, we demonstrate, by visual evaluation, time-consuming and the quantitative evaluation metric, that our method is superior to 9 baseline methods, including non-neural-network-based DT synthesis methods and neural-network-based DT synthesis methods.

A. Implementation Details

In the following experiments, the DT videos were collected from internet and two benchmark datasets, i.e., Gatech Graphcut Textures[1] [17] and Dyntex[2] [47]. These two benchmark datasets are publicly available and have been widely used in recent publications [1], [7]–[9], [11], [12], [15], [18]–[20]. We resize the frame size of all DT videos to 150×100 pixels, which

[1]http://www.cc.gatech.edu/cpl/projects/graphcuttextures
[2]http://projects.cwi.nl/dyntex/database.html

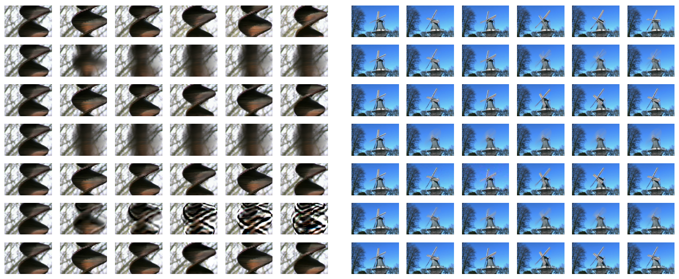


Fig. 4: Performance comparisons between different kernel functions used in our method for the videos "rotating wind ornament" (left) and "windmill" (right). For each category, the first row displays 6 frames of the observed sequence, and the other rows display the corresponding frames (left-to-right: 1-th, 100-th, 150-th, 210-th, 230-th, 250-th) of synthesized sequences generated by our method using different kernel functions (top-to-bottom: Linear kernel, Rational Quadratic kernel, Polynomial kernel, Multiquadric kernel, Sigmoid kernel, Gaussian kernel).

is similar to [8] for facilitating direct comparison. Moreover, we train our model using the first 59 to 200 frames because the length of the shortest observed sequences is 59 of each DT sequence. We synthesize a new one with the long-term frame to observed DT videos for quantitative evaluation. All the experiments presented in this paper are conducted in MATLAB 2018b under a windows 10 with 64 bit operating system.

In addition, we use two metrics to quantitatively evaluate the performance of the proposed method, including Peak Signal-Noise Ratio (PSNR) [48] and Structural SIMilarity (SSIM) [49]. They are common quantitative evaluation metrics in static image generation [10], [50], [51], DT synthesis [8], [9], [12] and other future frame prediction problem [21], [52]. Formally, PSNR can be written as Eq. (25).

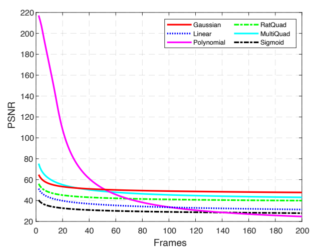
$$PSNR = \frac{1}{L-1}\sum_{t=2}^{L} 10\log_{10}\frac{255^2}{MSE\left(\widehat{S}_t - S_t\right)} \tag{25}$$

where L is the length of observed sequence, S_t $(t = 2, 3, \cdots, L)$ and $\widehat{S}_t$ $(t = 2, 3, \cdots, L)$ are the observed video frames and generated video frames, respectively. Intuitively, PNSR is presented with the prediction error between observed sequence and generated sequence. The higher the PSNR is, the better high-fidelity DT video is generated.

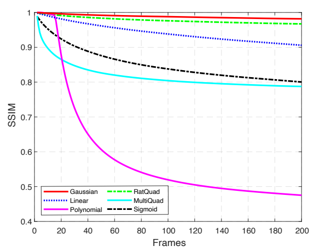
SSIM was originally designed for image quality assessment, and later is used for providing a perceptual judgment on similarity between videos. It can be formulated as Eq. (26).

$$\mathrm{SSIM}(\mathbf{x}, \mathbf{y}) = \frac{\left(2\mu_x\mu_y + C_1\right)\left(2\sigma_{xy} + C_2\right)}{\left(\mu_x^2 + \mu_y^2 + C_1\right)\left(\sigma_x^2 + \sigma_y^2 + C_2\right)} \tag{26}$$

where x and y are the frame of observed sequence ($\boldsymbol{S}$) and generated sequence ($\widehat{\boldsymbol{S}}$) respectively; μ_x and μ_y are the local means; σ_x and σ_y are the standard deviations; and the σ_{xy} is the cross-covariance for frame x and y; C_1 and C_2 are smooth factors. However, SSIM in Eq. (26) is used for evaluating the

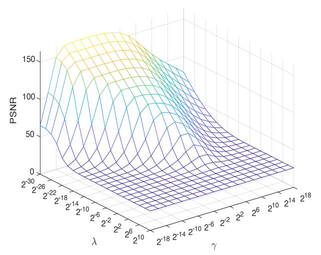


(a) PSNR

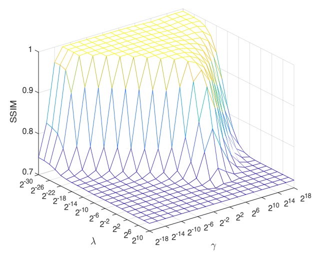


(b) SSIM

Fig. 5: Quantitative comparison of different kernel functions (Gaussian: Gaussian kernel; Linear: Linear kernel; Polynomial: Polynomial kernel; RatQuad: Rational Quadratic kernel; MultiQuad: Multiquadric kernel; Sigmoid: Sigmoid kernel).

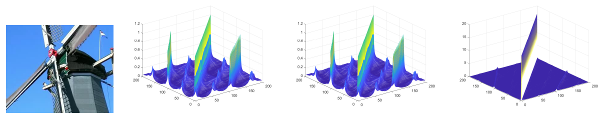


(a) PSNR



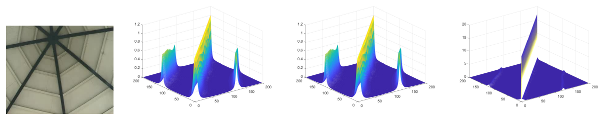
(b) SSIM

Fig. 6: Quantitative comparison of various regularization factors λ and kernel size γ used in our method on the whole Dyntex.



frame of sample original $\lambda = 2^{-20}$, $\gamma = 10^8$ $\lambda = 2^4$, $\gamma = 2^8$

(a) Windmill



frame of sample original $\lambda = 2^{-20}$, $\gamma = 2^8$ $\lambda = 2^4$, $\gamma = 2^8$

(b) Rotating

Fig. 7: Demonstrating the regularization ability of regularization factor λ. We display the corresponding kernel similarity matrices on two samples with different λ values: ($\lambda = 2^{-20}$ shows under-regularization; $\lambda = 2^4$ shows over-regularization).

similarity between two frames. For evaluating whole video sequence, the mean of SSIM is used, as shown in Eq. (27).

$$\text{SSIM}(\boldsymbol{S}, \widehat{\boldsymbol{S}}) = \frac{1}{L-1} \sum_{t=2}^{L} \text{SSIM}\left(S_t, \widehat{S}_t\right) \tag{27}$$

Obviously, SSIM ranges from -1 to 1 with a larger score indicating greater similarity. A larger SSIM indicates a better synthesis quality due to higher perceptual similarity between the synthesized and observed sequences. Therefore, SSIM used in this paper according to Eq. (27).

B. *Experiment 1: Hyper-Parameters Selection*

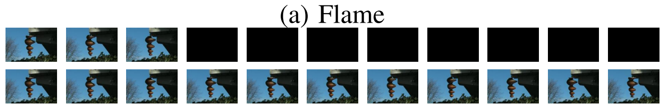
In principle, there are two hyper-parameters influencing the performance of our proposed method: the kernel function $K(u,v)$ (in Eq. (22)) and the regularization factor λ of kernel similarity embedding (in Eq. (22)). As the kernel function selection is important for kernel learning [8], [53], [54] and it can directly affect the stability of our method, we comprehensively test whole effects on the overall performance for obtaining the optimal solution at the beginning. Moreover, due to the regularization factor λ and kernel size γ interfere with the stability of leaning for kernel similarity embedding and impairs the generalization performance of our model, we also deal with the optimal selections of λ and γ.

1) Kernel Function $K(u,v)$: In fact, most DT sequences lie in nonlinear manifolds containing different data modalities in their appearance distribution, structure dimension, and stochastic repetitiveness, which are difficult to describe using low-dimensional latent variables with linear observation functions. Furthermore, the kernel function effectively represents the similarity correlation between different frames with Euclidean distance. Therefore, a kernel function is critical for kernel similarity embedding to make use of similarity prior knowledge in this work. Here we take several generic kernel functions (e.g., Linear kernel, Polynomial kernel, Gaussian kernel, Rational Quadratic kernel, Multiquadric kernel, and Sigmoid kernel) for testing and select the optimal one for our model.

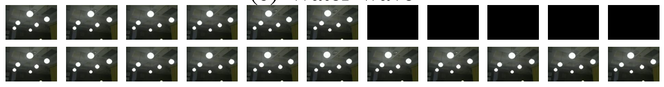
For evaluation, we test different kernel functions for our method on the Dyntex dataset. As shown in Figure 5, the different kernel function exhibit various performance. Gaussian and Rational Quadratic kernel functions outperforms other kernel functions with better PSNR and SSIM scores, which show that our method can synthesize more high-quality DT sequences using these two kernel functions. Furthermore, the Gaussian kernel function still achieves better performance after 200 frames of synthesized DT videos, which do not exist in the training frames. This shows that the Gaussian kernel function may be fit for our method with better generalization performance. Meanwhile, we display different frames of two synthesized DT sequences (rotating wind ornament and windmill) with different kernel functions, as shown in Figure 4. Intuitively, the frames of synthesized DTs using Gaussian kernel, Multiquadric kernel, and Rational Quadratic kernel are realistic, while other kernels are failed (especially after 200 frames). In summary, the Gaussian kernel achieved better performance of DT quality and sustainability. Therefore, we integrate the Gaussian kernel function ($K(u,v) = exp(-\gamma\|u-v\|^2)$) with ELM into a powerfully unified DT synthesis system to learn kernel similarity embedding for representing the spatial-temporal transition of DT videos in the later experiments.

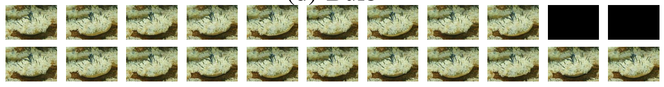
2) Regularization Factor λ, Kernel Size γ: According to ridge regression theory [60], we add a positive value $\lambda\mathbf{I}$ ($\mathbf{I}$ is identity matrix) to the diagonal axis of kernel similarity matrix (Eq. 22) for learning a more stable and better generalization performance of DT synthesis model. Furthermore, it is known that the performance of SVM is sensitive to the combination of the regularization factor and kernel size (λ, γ) [42]. Therefore,

(a) Flame


(b) Rotating wind ornament

(c) Water wave


(d) Bulb


(e) Flowers swaying with current

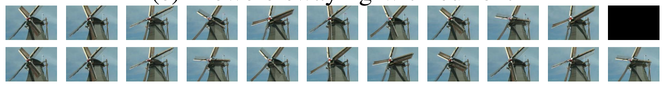
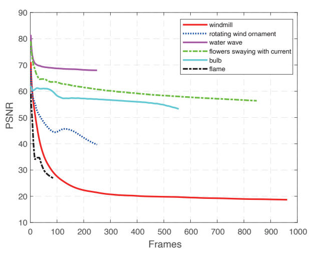
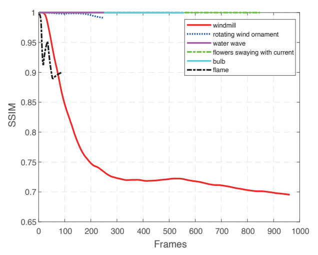

(f) Windmill

Fig. 8: Synthesizing long-term DT sequences using our method. For each category, the first row displays the 11 frames of the observed sequence (black frame denotes lacking the corresponding frame of observed sequence), and the second row displays the corresponding frames of synthesized videos. From left to right, the columns are the 2-nd, 100-th, 200-th, 300-th, 400-th, 500-th, 600-th, 700-th, 800-th, 900-th, 1000-th frames of observed sequences and synthesis sequences.



(a) PSNR



(b) SSIM

Fig. 9: Demonstrate the sustainability of our method with quantitative evaluation on 6 DT videos.

we also simultaneously analyze the influence of these two parameters (λ, γ) for our method.

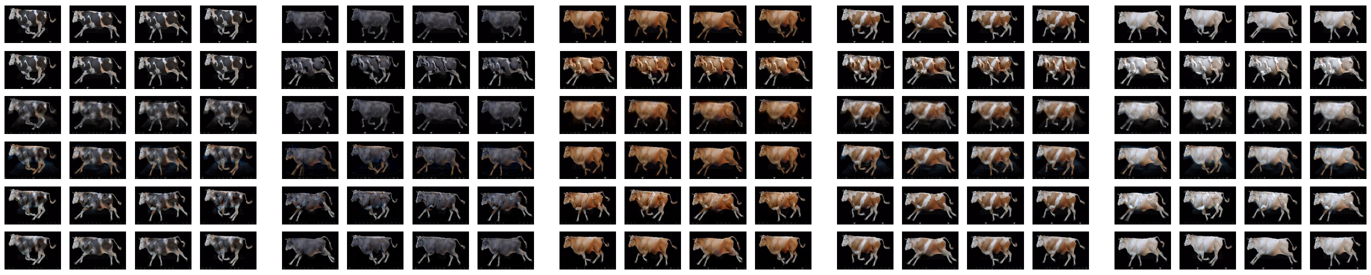
To achieve good generalization performance of our method, the regularization factor λ, and the kernel size γ of the model need to be chosen appropriately. We have tried a wide range of λ and γ. Specifically, we have used 21 different value of $\lambda = \{2^{-30}, 2^{-28}, \cdots, 2^{8}, 2^{10}\})$ and 19 different values of $\gamma = \{2^{-18}, 2^{-16}, \cdots, 2^{16}, 2^{18}\})$ for evaluation on Dyntex dataset, resulting in a total of 399 pairs of (λ, γ), as shown in Figure 6. See from Figure 6, we can find that the performance of our method will be stable in two time periods (period 1:$\lambda < 10^{-14}$, period 2:$\lambda > 10^{2}$), which show that our method is over-fitting and under-fitting respectively. Note that when λ is small, the PSNR can not be stable (see Figure 6(a)) because $PSNR \propto +\infty$ when some generated frames are extremely similar to observed frames. The regularization of the model is insufficient if regularization factor λ is too small, which results that the DT synthesis model is overly confident to the training frames (the first 200 frames) and fails to generate high-quality DT frames after 200 frames. However, if a too large λ is used, the model is over-regularized, which leads that the stationarity and repetitiveness of DT are smoothed overly. That is, the weak correlation between different frames is excessively decreased (see Figure 7). Figure 6 also show that the kernel size γ also closely interferes with the regularization ability of λ. We can observe that the PSNR and SSIM are not sensitive to γ while λ is small, but relatively large γ seems better. Therefore, the optimal combination of (λ, γ) of our method with Gaussian kernel is chosen for later experiments ($\lambda = 10^{-10}, \gamma = 10^{8}$).

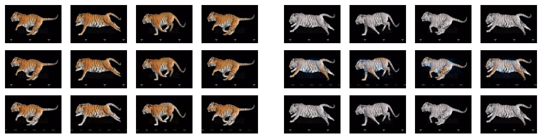
C. Experiment 2: Sustainability Analysis

DT synthesis aims to generate high-quality, long-term DT sequences, which requires that we should design a synthesis method with good sustainability performance that mainly refers to no obvious visual decays, divergences, and abrupt jump for generated long-term sequences. Therefore, we intuitively analysis the sustainability of our method using visual quality and quantitative evaluation metrics on two datasets.

For evaluation, the visual quality comparison over several synthesized DT sequences of the different classes is presented in Figure 8. To show the robustness of our method, we train the model using the first 200 frames if the length of observed sequences is longer than 200, and otherwise, the whole frames of observed sequences are used for training. We can observe that our method not only generates high-fidelity DTs in short-term, but also generates high-quality DTs in long-term even if the observed sequences are short, e.g. flame ($l = 88$), rotating wind ornament ($l = 250$), water wave ($l = 250$), bulb ($l = 556$), flowers swaying with current ($l = 848$), windmill ($l = 962$). Note that Figure 8(a) seemingly shares with similar DT (from 100-th to 1000-th), because these generated frames locate in similar/same cycle. Furthermore, our method still keeps synthesizing the realistic DT (including the details) in the long-term, e.g., the flag in sample "windmill" exhibits its dynamic in long-term generated sequences.

Indeed, we also show the quantitative evaluation results to demonstrate the sustainability of our method. Here we report the mean SSIM and PSNR in terms of frames (from 1 to 848) of 6 observed sequences that used in the former visual quality evaluation. See from Figure 9, our method achieves desired mean PSNR and SSIM, which show that it synthesized high-fidelity DTs. Although the mean PSNR and SSIM decrease as the number of generated frames increasing for some DT videos (e.g., windmill, frame), they are still huge ($PSNR > 18$ dB, $SSIM > 0.69$). Notably, our method achieved extensive SSIM index with 1 for whole long-term sequences just using 200 frames for training (e.g., flowers swaying with current, bulb, water wave), which suggests that its generated videos almost as same as the observed sequences. These results prove that our method accurately exhibits the statistical stationarity in the


(a) Running cows


(b) Running tigers

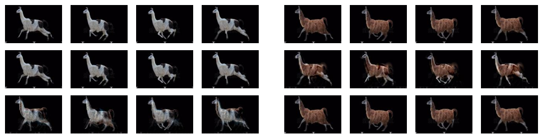

(c) Running llamas

Fig. 10: Sythesizing DTs by transferring the trained model using our method. For the running cows, the first row displays the 20 frames of 5 observed sequences, the other rows display the frames of the synthesized sequences corresponding to the first row with different trained model (from top to bottom: trained on cow1, trained on cow2, trained on cow3, trained on cow4, trained on cow5). For the running tiger and running llamas, they are similar to the running cows, but they just use 2 observed sequences for training and testing.

spatial domain and the stochastic repetitiveness in the temporal dimension of DT sequences using kernel similarity embedding, and thus it can synthesize realistic DTs in long-term.

D. Experiment 3: Generalization Analysis

Good generalization performance is a key goal for all learning tasks. Similar to [12], we also specialize in our method to learn roughly aligned DT videos, which are non-stationary in either the spatial or temporal domain. In this study, it is different from [12] by training a model using all roughly aligned with video sequences for one example (e.g., 5 training sequences for the running cow). Our method trains a model just using one video sequence for one example, which may effectively verify the generalization performance of our method.

Spatially aligned with the sense for each time step, the target objects in different videos possess the same locations, shapes, and poses, while it is the same as temporally aligned with the starting and ending times of the actions in different videos. We take the DT videos that were used in [12] for evaluation. See from figure 10, the 3 results of modeling and synthesizing DTs from roughly aligned video sequences are displayed. Specifically, we firstly trained a model on each sequence of the running cows/tigers/llamas, and then test the 5/2/2 trained models on the 5/2/2 observed sequences. Thus, we gain 33 realistic, synthesized sequences.

The experiment results show that our method can transfer the trained model to generate new sequences for other spatial-temporally aligned DT sequences. In summary, our method is effective and efficient for synthesizing realistic appearances and motions of the test animals, which suggests that our method performs excellent generalization performance. Indeed, the part of transferred model could not synthesize consistent motions for some cows (e.g., cow1$\rightarrow$cow2), because these samples are not aligned initially well.

E. Experiment 4: Comparisons to State-of-the-Arts

In this section, we compare our method with 9 state-of-the-art methods of DT synthesis, including non-neural-network-based methods(LDS [1], FFT-LDS [19], HOSVD [15], SLDS [20], Kernel-DT [18] and KPCR [8]) and neural-network-based methods (TwoStream[3] [11], STGCN[4] [12] and DG[5] [22]). To better verify and validate the performance of our method, we simultaneously leverage the quantitative evaluation metric (SSIM, PSNR), time-consuming and vision quality.

Specifically, we first compare our method with 6 non-neural-network-based methods, including FFT-LDS [19], HOSVD [15], KPCR [8], LDS [1], SLDS [20], Kernel-DT [18]. To facilitate direct comparison, we tested all these models with 150$\times$100 pixels using 17 gray DT videos on PSNR and SSIM. See from Table I, our method attained the best performance on most of DT sequences, except for the videos of flashing lights and beach. That is because these two videos lack good spatial-temporal characteristic of DT. Notably, the proposed method beats the second-best results by a large margin (19.017 dB for average PSNR and 0.153 for average SSIM on 17 DT videos). Our method also achieves an significant SSIM index with 1 for some DT videos (e.g., bulb, fountain, spring water, etc.). Therefore, one can learn that our method can make full use of the similarity pror knowledge for DT synthesis using kernel similarity embedding. Indeed, all methods fail to synthesize high-quality DT video for rotating wind ornament, because this sample is originally blurry.

Then, our model is compared with 3 neural-network-based methods, such as TwoStream [11] and STGCN [12] and DG [22] on 6 DT videos (e.g., elector , flashlights and beach, etc). For a fair comparison, we display 6 same index frames of

[3]https://ryersonvisionlab.github.io/two-stream-projpage/
[4]http://www.stat.ucla.edu/ jxie/STGConvNet/STGConvNet.html
[5]http://www.stat.ucla.edu/jxie/DynamicGenerator/DynamicGenerator.html

TABLE I: COMPARISON WITH NON-NEURAL-NETWORK-BASED DT SYNTHESIS METHODS ON PSNR (dB) AND SSIM.

	Ours		FFT-LDS [19]		HOSVD [15]		KPCR [8]		LDS [1]		SLDS [20]		Kernel-DT [18]	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
boiling water	**36.591**	**0.958**	27.570	0.887	27.617	0.891	24.726	0.840	27.604	0.891	27.604	0.891	26.114	0.870
elevator	**46.149**	**0.996**	34.307	0.949	34.289	0.946	31.029	0.913	34.420	0.952	34.384	0.951	30.109	0.893
rotating wind ornament	**15.882**	0.564	13.387	0.500	12.644	0.473	15.038	**0.569**	13.387	0.500	13.387	0.500	12.131	0.459
flower in current	**47.609**	**0.999**	30.372	0.922	31.778	0.949	37.069	0.988	31.392	0.946	30.922	0.937	27.297	0.891
bulb	**49.445**	**1.000**	31.229	0.958	31.236	0.972	28.024	0.957	31.350	0.972	31.350	0.972	29.788	0.978
flashing lights	17.874	0.748	26.714	0.875	26.712	0.876	22.242	0.796	**26.724**	**0.878**	**26.724**	**0.878**	25.668	0.788
spring water	**64.198**	**1.000**	21.435	0.607	21.363	0.606	21.271	0.641	21.453	0.610	21.453	0.610	21.211	0.643
washing machine	**33.399**	**0.960**	30.875	0.931	30.865	0.933	28.630	0.905	30.913	0.934	30.913	0.934	26.391	0.902
fountain	**68.641**	**1.000**	19.567	0.401	19.554	0.401	18.394	0.357	19.569	0.402	19.569	0.402	18.745	0.382
water spray	**43.215**	**0.994**	29.387	0.880	29.683	0.890	30.740	0.917	29.483	0.889	28.654	0.878	25.565	0.846
water spray in a pool	**67.475**	**1.000**	21.076	0.426	21.036	0.423	20.603	0.433	21.079	0.427	21.079	0.427	19.483	0.394
water wave	**51.840**	**0.999**	27.385	0.650	27.311	0.646	27.767	0.745	27.394	0.651	27.394	0.651	22.371	0.537
waterfall	**44.708**	**0.998**	39.649	0.991	43.240	0.998	45.337	0.999	41.310	0.996	41.310	0.996	27.073	0.955
waterfall in a mountain	**71.851**	**1.000**	18.509	0.539	18.507	0.540	18.408	0.534	18.513	0.540	18.513	0.540	18.303	0.535
flag	**53.605**	**1.000**	23.839	0.858	23.797	0.859	20.537	0.802	23.840	0.859	23.840	0.859	23.068	0.854
flame	**46.185**	**0.910**	27.567	0.874	27.463	0.867	26.974	0.904	27.558	0.877	26.435	0.857	33.495	0.887
beach	22.736	0.719	26.684	0.838	31.017	0.899	**33.148**	**0.942**	26.779	0.846	26.779	0.846	29.251	0.905
mean	**45.965**	**0.932**	26.444	0.770	26.948	0.775	26.467	0.779	26.633	0.775	26.489	0.772	24.474	0.748

TABLE II: COMPARISON WITH STATE-OF-THE-ART DT SYNTHESIS METHODS ON TIME-CONSUMING.

	Ours	FFT-LDS [19]	HOSVD [15]	KPCR [8]	LDS [1]	SLDS [20]	Kernel-DT [18]	TwoStream [11]	STGCN [12]	DG [22]
Train. time (Sec.)	0.090	0.928	1.399	1.990	0.148	2.475	0.830	-	4188	3904.418
Test time (Sec.)	26.060	5.516	4.922	12.214	4.048	3.799	1007.260	8235	7.210	52.292
Generated frames	1200	1200	1200	1200	1200	1200	1200	12	70	120
Using GPU	×	×	×	×	×	×	×	✓	✓	✓
FPS	46.040	217.560	243.790	98.248	296.450	315.900	1.191	0.002	9.709	2.295

generated sequences. See from Figure 11, the DT sequences generated by TwoStream are divergent because the TwoStream method has a limitation that it cannot well synthesize DTs not being spatially homogeneous (e.g., elevator, water spray). As for STGCN and DG, the DT sequences generated by them appear blurred because these two methods lie on more training data. Intuitively, our method generated high-fidelity DT sequences, including realistic details of DTs.

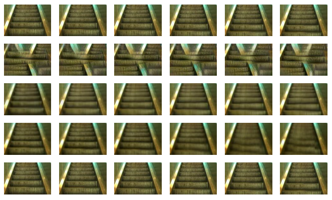
Finally, we report the time-consuming of different DT synthesis methods including neural-network-based methods and non-neural-network-based methods. As shown in Table II, our method can satisfy the real-time (25 fps) generation with 46.040 fps as well as non-neural-network-based methods (except for Kernel-DT), while the neural-network-based methods are failed. Moreover, the neural-network-based methods are time-consuming and computationally expensive for training. In summary, our method powerfully synthesizes high-quality DT videos with fast speed and low computation superior to the state-of-the-art DT methods, which benefits from the discriminative representation of kernel similarity embedding for exhibiting the spatial-temporal transition of DTs. It directly shows that the similarity correlation of different frames is a critical prior knowledge for DT synthesis.

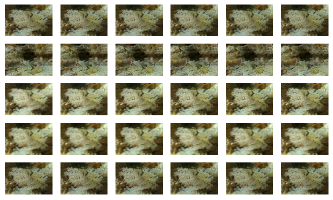
V. DICUSSION

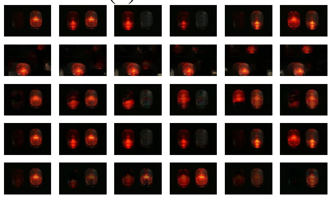
In this study, we propose a novel DT synthesis method to address the high-dimensionality and small sample issues for DT synthesis. Specifically, our method leverages a kernel similarity matrix to mine and capture the similarity prior knowledge of DT, which is embedded into kernel similarity embedding. Then, high-fidelity DTs are synthesized iteratively by learned model. Notably, our method is dissimilar to the existing kernel-based DT synthesis methods [8], [18], [55], which use kernel function to learn a nonlinear observation function for dimensionality-reduction. The experimental results on well-known benchmark datasets show that the similarity correlation is a critical prior knowledge for representing DT and the kernel similarity embedding effectively solves the aforementioned issues. Thus our method can achieve promising results of DT synthesis.

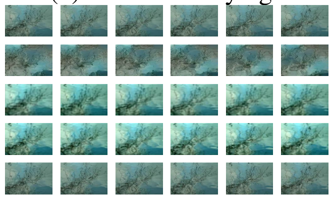
For evaluating our method, we intuitively and theoretically analyzed the effectiveness of kernel similarity embedding for DT synthesis (Section III-C). Then, we evaluated the influence of the selection of kernel function, the regularization factor λ and the kernel size γ (Section IV-B). Meanwhile, we intuitively validated the sustainability and generalization of our method using vision quality and quantitative evaluation metrics (Section IV-C and Section IV-D). Eventually, we took our method to compare with 9 state-of-the-art methods (including neural-network-based methods and non-neural-network-based methods) (Section IV-E).

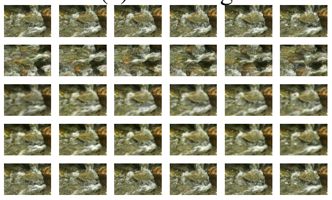
Although our method has achieved promising results, it has a limitation that the sustainability is impacted if DTs lack of stochastic repetitiveness in the temporal dimension (e.g., waterfall, spring water). See from Figure 12, our method falls in visual decay after 200 frames, and thus it fails to synthesize high-fidelity DTs for the long-term. In the future, the scalable kernel similarity embedding may be a potential choice to overcome this limitation. Because scalable kernel similarity embedding could adjust the similarity representation, and thus the statistical stationarity in the spatial domain and stochastic repetitiveness in the temporal dimension can be artifically controled.


(a) Elevator


(b) Flowers swaying


(c) Flash lights


(d) Water wave


(e) Spring water

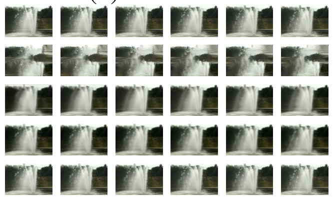
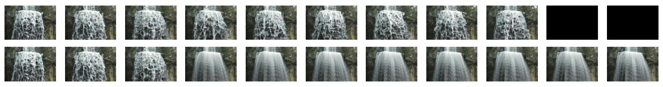

(f) Water spray

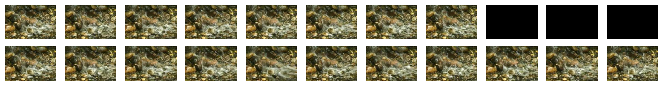
Fig. 11: Visual quality comparison between three neural-network-based methods and our method for 6 different DT videos. For each category, the first row displays 6 frames of the observed sequence, and the other rows display the corresponding frames of synthesized sequences generated by different methods (from top to bottom: TwoStream [11], STGCN [12], DG [22] and our method).

VI. Conclusion

In this paper, we proposed a novel DT synthesis method that integrates kernel learning and extreme learning machine into a powerfully unified synthesis method to learn kernel similarity embedding for representing DT. Notably, kernel similarity embedding not only effectively address the high-dimensionality and small sample issues using similarity prior knowledge, but also has the advantage of modeling nonlinear representation feature relationship for DT. The competitive results on DT videos collected from two benchmark datasets and the internet demonstrate the superiority and great potentials of our method for DT synthesis. It also shows obvious advantages over all the compared state-of-the-art approaches.

In the future, we will design a more effective learning model to learn scalable kernel similarity embedding for DT synthesis, because scalable kernel similarity embedding will effectively control the similarity representations of DT. Furthermore, we will also adopt multi-view methods into DT synthesis as some DT sequences were acquired with the moving camera and different views (e.g., some DT videos in Dyntex).


(a) Waterfall


(b) Spring water

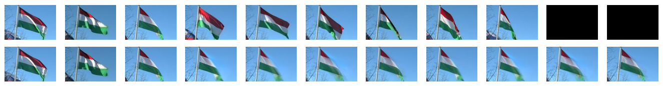

(c) Flag

Fig. 12: Displaying some generated frames of DT sequences with limitation of our method. For each category, the first row displays the 11 frames of the observed sequence (black frame denotes lacking the corresponding frame of observed sequence), and the second row displays the corresponding frames of synthesized videos by our method. From left to right, the columns are the 2-nd, 100-th, 200-th, 300-th, 400-th, 500-th, 600-th, 700-th, 800-th, 900-th, 1000-th frames of observed sequences and synthesis sequences.

References

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

[2] Y. Wang and S.-C. Zhu, "A generative method for textured motion: Analysis and synthesis," in *Proc. ECCV*, 2002, pp. 583–598.

[3] C. Feichtenhofer, A. Pinz, and R. P. Wildes, "Temporal residual networks for dynamic scene recognition," in *Proc. CVPR*, 2017, pp. 4728–4737.

[4] A. B. Chan and V. Nuno, "Modeling, clustering, and segmenting video with mixtures of dynamic textures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 5, pp. 909–926, 2008.

[5] A. Mumtaz, W. Zhang, and A. B. Chan, "Joint motion segmentation and background estimation in dynamic scenes," in *Proc. CVPR*, 2014, pp. 368–375.

[6] C.-C. Hsu, L.-W. Kang, and C.-W. Lin, "Temporally coherent superresolution of textured video via dynamic texture synthesis," *IEEE Transactions on Image Processing*, vol. 24, pp. 919–931, 2015.

[7] L. A. Gatys, A. S. Ecker, and M. Bethge, "Texture synthesis using convolutional neural networks," in *Proc. NeurIPS*, 2015, pp. 262–270.

[8] X. You, W. Guo, S. Yu, K. Li, J. C. Príncipe, and D. Tao, "Kernel learning for dynamic texture synthesis," *IEEE Transactions on Image Processing*, vol. 25, no. 10, pp. 4782–4795, 2016.

[9] J. Xie, S.-C. Zhu, and Y. N. Wu, "Synthesizing dynamic patterns by spatial-temporal generative ConvNet," in *Proc. CVPR*, 2017, pp. 7093–7101.

[10] Q. Chen and V. Koltun, "Photographic image synthesis with cascaded refinement networks," in *Proc. ICCV*, 2017, pp. 1511–1520.

[11] M. Tesfaldet, M. A. Brubaker, and K. G. Derpanis, "Two-stream convolutional networks for dynamic texture synthesis," in *Proc. CVPR*, 2018, pp. 6703–6712.

[12] J. Xie, S.-C. Zhu, and Y. N. Wu, "Learning energy-based spatial-temporal generative convnets for dynamic patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. [Online]. Available: http://doi.org/10.1109/TPAMI.2019.2934852

[13] V. Pegoraro and S. G. Parker, "Physically-based realistic fire rendering," in *Proc. NPH*, 2006, pp. 51–59.

[14] A. Nealen, M. Müller, R. Keiser, E. Boxerman, and M. Carlson, "Physically based deformable models in computer graphics," *Computer Graphics Forum*, vol. 25, no. 4, pp. 809–836, 2006.

[15] R. Costantini, L. Sbaiz, and S. Süsstrunk, "Higher order svd analysis for dynamic texture synthesis," *IEEE Transactions on Image Processing*, vol. 17, pp. 42–52, 2008.

[16] A. Schödl, R. Szeliski, D. Salesin, and I. A. Essa, "Video textures," in *Proc. SIGGRAPH*, 2000, pp. 489–498.

[17] V. Kwatra, A. Schödl, I. A. Essa, G. Turk, and A. F. Bobick, "Graphcut textures: image and video synthesis using graph cuts," *ACM Transactions on Graph*, vol. 22, pp. 277–286, 2003.

[18] A. B. Chan and N. Vasconcelos, "Classifying video with kernel dynamic textures," 2007, pp. 1–6.

[19] B. Abraham, O. I. Camps, and M. Sznaier, "Dynamic texture with fourier descriptors," in *Proc. the 4th International Workshop on Texture Analysis*, 2005, pp. 53–58.

[20] S. M. Siddiqi, B. Boots, and G. J. Gordon, "A constraint generation approach to learning stable linear dynamical systems," in *Proc. NeurIPS*, 2007, pp. 1329–1336.

[21] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz, "MoCoGAN: Decomposing motion and content for video generation," in *Proc. CVPR*, 2018, pp. 1526–1535.

[22] J. Xie, R. Gao, Z. Zheng, S.-C. Zhu, and Y. N. Wu, "Learning dynamic generator model by alternating back-propagation through time," in *Proc. AAAI*, 2019, pp. 5498–5507.

[23] S. Chen, Y. Wang, C.-J. Lin, W. Ding, and Z. Cao, "Semi-supervised feature learning for improving writer identification," *Information Sciences*, vol. 482, pp. 156–170, 2019.

[24] Z. Zheng, X. Yang, Z. Yu, L. Zheng, Y. Yang, and J. Kautz, "Joint discriminative and generative learning for person re-identification," in *Proc. CVPR*, 2019, pp. 2138–2147.

[25] D. Cheng, Y. Gong, S. Zhou, J. Wang, and N. Zheng, "Person re-identification by multi-channel parts-based cnn with improved triplet loss function," in *Proc. CVPR*, 2016, pp. 1335–1344.

[26] Y. Fu, Y. Wei, G. Wang, X. Zhou, H. Shi, and T. S. Huang, "Self-similarity grouping: A simple unsupervised cross domain adaptation approach for person re-identification," in *Proc. ICCV*, 2019, pp. 6112–6121.

[27] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," *arXiv preprint arXiv:1703.07737*, 2017.

[28] M. Lin, R. Ji, H. Liu, X. Sun, Y. Wu, and Y. Wu, "Towards optimal discrete online hashing with balanced similarity," in *Proc. AAAI*, 2019, pp. 8722–8729.

[29] H. Liu, R. Ji, Y. Wu, F. Huang, and B. Zhang, "Cross-modality binary code learning via fusion similarity hashing," in *Proc. CVPR*, 2017, pp. 6345–6353.

[30] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *Proc. CVPR*, 2012, pp. 2074–2081.

[31] F. Çakir and S. Sclaroff, "Adaptive hashing for fast similarity search," in *Proc. ICCV*, 2015, pp. 1044–1052.

[32] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge: Cambridge University Press, 2004.

[33] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *Proc. IJCNN*, 2004, pp. 985–990.

[34] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, pp. 513–529, 2012.

[35] J. Tang, C. Deng, and G.-B. Huang, "Extreme learning machine for multilayer perceptron," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, pp. 809–821, 2016.

[36] L. L. C. Kasun, Y. Yang, G.-B. Huang, and Z. Zhang, "Dimension reduction with extreme learning machine," *IEEE Transactions on Image Processing*, vol. 25, pp. 3906–3918, 2016.

[37] C. Deng, S. Wang, Z. Li, G.-B. Huang, and W. Lin, "Content-insensitive blind image blurriness assessment using weibull statistics and sparse extreme learning machine," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, pp. 516–527, 2019.

[38] M. Saito, E. Matsumoto, and S. Saito, "Temporal generative adversarial nets with singular value clipping," 2017, pp. 2849–2858.

[39] X. Xing, T. Han, R. Gao, S.-C. Zhu, and Y. N. Wu, "Unsupervised disentangling of appearance and geometry by deformable generator network," in *Proc. CVPR*, 2019, pp. 10 354–10 363.

[40] Y. Zhou and T. L. Berg, "Learning temporal transformations from time-lapse videos," in *Proc. ECCV*, 2016, pp. 262–277.

[41] P. L. Bartlett, "The sample complexity of pattern classification with neural networks: The size of the weights is more important than the size of the network," *IEEE Transactions on Information Theory*, vol. 44, pp. 525–536, 1996.

[42] J. A. K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Processing Letters*, vol. 9, pp. 293–300, 1999.

[43] A. E. Hoerl and R. W. Kennard, "Ridge regression: biased estimation for nonorthogonal problems," *Technometrics*, vol. 42, no. 1, pp. 80–86, 2000.

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

[45] L. Song, J. Huang, A. J. Smola, and K. Fukumizu, "Hilbert space embeddings of conditional distributions with applications to dynamical systems," in *Proc. ICML*, 2009, pp. 961–968.

[46] L. Song, K. Fukumizu, and A. Gretton, "Kernel embeddings of conditional distributions: A unified kernel framework for nonparametric inference in graphical models," *IEEE Signal Processing Magazine*, vol. 30, no. 4, pp. 98–111, 2013.

[47] R. Péteri, S. Fazekas, and M. J. Huiskes, "Dyntex: A comprehensive database of dynamic textures," *Pattern Recognition Letters*, vol. 31, pp. 1627–1632, 2010.

[48] Z. Wang and A. C. Bovik, "Mean squared error: Love it or leave it? a new look at signal fidelity measures," *IEEE Signal Processing Magazine*, vol. 26, pp. 98–117, 2009.

[49] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[50] D. Liu, Z. Wang, B. Wen, J. Yang, W. Han, and T. S. Huang, "Robust single image super-resolution via deep networks with sparse prior," *IEEE Transactions on Image Processing*, vol. 25, pp. 3194–3207, 2016.

[51] X. Liu, D. Zhai, R. Chen, X. Ji, D. Zhao, and W. Gao, "Depth super-resolution via joint color-guided internal and external regularizations," *IEEE Transactions on Image Processing*, vol. 28, pp. 1636–1645, 2019.

[52] W. Xiong, W. Luo, L. Ma, W. Liu, and J. Luo, "Learning to generate time-lapse videos using multi-stage dynamic generative adversarial networks," in *Proc. CVPR*, 2018, pp. 2364–2373.

[53] X. Liu, X. Zhu, M. Li, L. Wang, C. Tang, J. Yin, D. Shen, H. Wang, and W. Gao, "Late fusion incomplete multi-view clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, pp. 2410–2423, 2018.

[54] X. Liu, M. Li, L. Wang, Y. Dou, J. Yin, and E. Zhu, "Multiple kernel k-means with incomplete kernels," *IEEE transactions on Pattern Analysis and Machine Intelligence*, 2019. [Online]. Available: http://doi.org/10.1109/TPAMI.2019.2892416

[55] Z. Zhu, X. You, S. Yu, J. Zou, and H. Zhao, "Dynamic texture modeling and synthesis using multi-kernel gaussian process dynamic model," *Signal Processing*, vol. 124, pp. 63–71, 2016.

Shiming Chen is currently a full-time Ph.D student in the School of Electronic Information and Communitaions, Huahzong University of Sciences and Technology, China. His current research interests include image/video synthesis, computer vision and machine learning.

Peng Zhang is currently pursuing the Ph.D degree in the School of Electronic Information and Communications at Huazhong University of Science and Technology, China. His research interests include computer vision, pattern recognition, and machine learning.

Xinge You is currently a Professor in School of Electronics Information and Communications, Huazhong University of Science and Technology. He received the Ph.D. degree in Department of Computer Science, Hong Kong Baptist University, in 2004. His research results have expounded in 150+ publications at prestigious journals and prominent conferences, such as IEEE T-PAMI, T-IP, T-NNLS, T-CYB, T-CSVT, IJCAI, ECCV. His current research interests include pattern recognition, machine earning, and computer vision.

Xin Liu is currently an Associate Professor with the Department of Computer Science and Technology, Huaqiao University, China. He received the Ph.D. degree in computer science from Hong Kong Baptist University, Hong Kong, in 2013. His research results have expounded in 30+ publications at prestigious journals and prominent conferences, such as IEEE T-PAMI, T-NNLS, T-IFS, PR, CVIU, ICASSP, ICME. His current research interests include multimedia analysis, computer vision, and machine learning.

Zehong Cao is a Lecturer with the Discipline of Information and Communication Technology, School of Technology, Environments and Design, University of Tasmania, Australia. He received the Ph.D. degree in information technology from the University of Technology Sydney, Australia, in 2017. He had an ESI highly cited paper in 2019 and a string of successful over 30 publications among the most respected journals, including Nature Scientific Data, IEEE T-FS, T-NNLS, T-CYB, T-SMCA, etc. His current esearch interests include computer vision, machine learning, and bio-signal processing.

Dacheng Tao (F'15) is a Professor of computer science and an ARC Laureate Fellow with the School of Information Technologies and the Inaugural Director of the UBTECH Sydney Artificial Intelligence Centre, at the University of Sydney. He mainly applies statistics and mathematics to artificial intelligence and data science. His research results have expounded in one monograph and 300+ publications at prestigious journals and prominent conferences, such as IEEE T-PAMI, T-IP, IJCV, JMLR, NeurIPS, ICML, CVPR, ICCV, ECCV; and the 2017 IEEE Signal Processing Society Best Paper Award. He is a fellow of the Australian Academy of Science, AAAS, IEEE, IAPR, OSA, and SPIE.