Kernelized Similarity Learning and Embedding for Dynamic Texture Synthesis

Shiming Chen, Peng Zhang, Guo-Sen Xie, Qinmu Peng, Zehong Cao, Wei Yuan, and Xinge You, Senior Member, 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 a high similarity correlation that is critical prior knowledge. However, existing methods cannot effectively learn a synthesis model for high-dimensional DT from a small number of training samples. In this article, we propose a novel DT synthesis method, which makes full use of similarity as prior knowledge to address this issue. Our method is based on the proposed kernel similarity embedding, which can not only mitigate the high dimensionality and small sample issues, but also has the advantage of modeling nonlinear feature relationships. Specifically, we first put forward two hypotheses that are essential for the DT model to generate new frames using similarity correlations. Then, we integrate kernel learning and the extreme learning machine into a unified synthesis model to learn kernel similarity embeddings for representing DTs. Extensive experiments on DT videos collected from the Internet and two benchmark datasets, i.e., Gatech Gradient Textures and DynTex, demonstrate that the learned kernel similarity embeddings can provide discriminative representations for DTs. Further, our method can preserve the long-term temporal continuity of the synthesized DT sequences with excellent sustainability and generalization. Meanwhile, it effectively generates realistic DT videos with higher speed and lower computation than the current 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), extreme learning machine (ELM), kernel similarity embedding, similarity prior knowledge.

I. INTRODUCTION AND MOTIVATION

DYNAMIC texture (DT), which exhibits statistical stationarity in the spatial domain and stochastic repetitiveness in the temporal dimension, is one type of dynamic pattern in computer vision [1], e.g., moving vehicles, falling water, flaming fire, and rotating windmill. Due to the demands of dynamic patterns synthesis in video technology applications (e.g., texture recognition [2], video segmentation [3], and super-resolution [4]), synthesizing DTs have gradually become a topic of interest in computer graphics and computer vision [5]–[10]. DT synthesis aims to infer a generation process from a DT example, which then allows an infinitely varying stream of similar looking texture videos to be produced.

In general, DT synthesis methods can be categorized into two groups: 1) non-neural-network-based and 2) neural-network-based methods. The first group methods are popular early approaches for DT synthesis, and can be further classified as physics-based methods [11], [12] and dynamic system (DS) modeling methods [1], [5], [13]–[15]. The second group of methods automatically learn the texture distribution thanks to the effective representation of neural networks [6], [9], [10], [16]–[18]. However, DT is high-dimensional data and with limited samples (DT samples are made up of a single training video with a short sequences length). Non-neural-network-based methods usually seek to reduce the dimensionality of DT for modeling, which may lead to information loss. Further, it is difficult to design a proper dimensionality-reduction algorithm. Meanwhile, neural-network-based methods also 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 model for DT synthesis, which mines and exploits similarity correlations as prior knowledge.

In fact, the similarity correlation between frames is an explicit expression of the statistical stationarity and stochastic repetitiveness of DT. It is critical for distinguishing DTs from other videos. Similarity representations serve as the learning objective of metric learning for discriminative models [2], [19], [20], which indicates the importance of similarity correlations for representation. Some researchers have also attempted to mine the potential similarity knowledge of samples to improve the performance of discriminative models [21]–[25]. This suggests that similarity correlations are as important as class labels and other annotation information when it comes to prior knowledge. Moreover, similarity correlations...
In this work, we propose a novel DT synthesis method to generate high-quality, long-term DT sequences with high speed and low computation. We integrate kernel learning and ELM into a powerfully unified synthesis model to learn kernel similarity embeddings for representing the statistical stationarity in the spatial domain and stochastic repetitiveness in the temporal dimension of DT sequences. Specifically, we preprocess each input DT sequence $S_N$ ($N$ is the length of the sequence), which are divided into two parts: 1) explanatory frames and 2) response frames. Then, our method uses a kernel function to replace the feature mapping function of the hidden layer of the ELM, and thus the kernel similarity embedding is easily learned after training. Finally, the DT sequence is iteratively generated by our trained model.

To summarize, this study makes the following salient contributions.

1) We introduce two important hypotheses that benefit the DT system, allowing it to generate new frames based on current ones, using the similarity correlation of different frames.
2) We propose a novel DT synthesis method, which learns kernel similarity embeddings to synthesize realistic video sequences with good sustainability and generalization.
3) We introduce kernel similarity embedding to mine and exploit similarity as prior knowledge of DT and analyze its availability with intuitive and theoretical insights.
4) We carry out extensive experiments on benchmark datasets to demonstrate that our method provides consistent improvement over the state-of-the-art methods.

The remainder of this article 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 directions 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 $Y = [y_1, \ldots, y_N]^\top \in \mathbb{R}^{N \times m}$ ($N$ is the length of the sequence and $m$ is the dimensionality of the frame) of training set, which can be formulated as follows:

$$y_i = f(y_{i-1}).$$  \hspace{1cm} (1)

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

$$f' = \arg\min_f \frac{1}{2} \sum_{i=2}^{N} (y_i - f(y_{i-1}))^2.$$  \hspace{1cm} (2)

After training, $f'$ can be learnt. Subsequently, given an initial frame $y'_{i-1}$ from an input texture sequence of test set, the endless sequences $\{y'_i\}_{i=2,3,\ldots}$ can be generated iteratively by the following:

$$y'_i = f'(y'_{i-1}).$$  \hspace{1cm} (3)
In the following, we provide a comprehensive review of related work on 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 complicated models. Pegoraro and Parker [11] 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. Higher-order singular value decomposition (HOSVD) analysis for DT synthesis was proposed in [12]. It decomposes the DT into a multidimensional signal without unfolding the video sequences as column vectors, thus allowing dimensionality reduction to be performed in the spatial, temporal, and chromatic domain. In summary, although physics-based methods can generate impressive DTs, they are highly application specific with weak generalization.

2) Dynamic System Modeling Methods: DS modeling methods for DT synthesis are the most popular non-neural-network-based methods. DS modeling methods typically learn transition functions to represent the correlation between different frames of DTs using linear or nonlinear dimensionality-reduction algorithms. However, a proper dimensionality-reduction algorithm is difficult to design, which may lead to the information loss of DT. This is the main limitation of DS modeling methods. Doretto et al. [1] proposed a pioneering DS method for DT synthesis using a simple linear DS (LDS) to project the input video frames into lower-dimensional space by SVD. Siddiqi et al. [15] proposed a stable-LDS (SLDS)-based method to incrementally add constraints to a relaxed system solution and to improve stability. To better adapt the standard LDS-based method to devices with limited memory and computational power, Abraham et al. [14] proposed a new DT synthesis with Fourier descriptors (FFT-LDS), which requires far fewer parameters compared to standard LDS approaches. Chain and Vasconcelos [13] introduced a new method (Kernel-DT) for DT synthesis using kernel principal component analysis (KPCA) to learn a nonlinear observation function. However, the aforementioned DS modeling method faces the fundamental problem in that 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 et al. [5].

B. Neural-Network-Based Methods

Neural networks have proven to be immensely successful discriminative and generative learning machines [19], [34]–[36]. In terms of DT synthesis, various approaches based on ConvNet have been proposed [6], [9], [10], [16], [17]. Gatys et al. [16] introduced a new model for 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. [9] proposed a two-stream model for DT synthesis. This method represents the texture’s appearance and dynamic nature using a set of Gram matrices. Tulyakov et al. [18] presented the motion and content decomposed generative adversarial network (MoCoGAN) for video generation. MoCoGAN is good at generating DTs when learning from a large amount of training data. Xie et al. [6], [17] proposed an energy-based spatial-temporal generative ConvNet to model and synthesize dynamic patterns. This model is beneficial for generating realistic dynamic patterns when the input sequences are incomplete with either occluded pixels or missing frames. Xie et al. [10] presented a dynamic generator model using an 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 train a discriminative network or an inference network. Nevertheless, neural-network-based methods fail to exhibit powerful representations because DTs are high-dimensional data and with limited samples. Meanwhile, these methods are time-consuming and computationally expensive due to their large number of parameters. Therefore, an ELM may be the desired successor for DT synthesis, with good generalization performance and a high learning speed expected.

III. PROPOSED DT SYNTHESIS METHOD

Similarity, which exists in different samples or self-samples, is as important prior knowledge as the class labels and other annotation information. Thus, some researchers have recently considered making use of the similarity knowledge of samples to improve the performance of the discriminative model for various tasks, i.e., person reidentification (re-ID) [21]–[23] and content-based image retrieval [24], [25]. DT exhibits statistical stationarity in the spatial domain and stochastic repetitiveness in the temporal dimension, indicating that different frames of DT possess high similarity correlations. These similarity correlations can also be viewed as critical prior knowledge, which may mitigate the high dimensionality and small sample issues for DT synthesis. Therefore, we propose a novel DT synthesis method. Our model synthesizes desirable DTs at a high speed using kernel similarity embedding, based on ELM and kernel learning.

In this section, we first revisit ELM to better understand the proposed method. We then illustrate our method, which uses ELM-based kernel similarity embedding. Meanwhile, we introduce the additive regularization factor for smoothing the kernel similarity embedding, which means our method is stabler with 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 the Extreme Learning Machine

ELM was initially proposed by Huang et al. [37], serving as an emergent technology that has recently attracted much attention [27]–[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 several challenges (e.g., slow learning speed, trivial human intervention, 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, in this article, we take full advantage of ELM to expedite the generation speed for DT synthesis, with good generalization.

Before introducing ELM formally, we define some notations. Consider a dataset $D = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, where $x_i \in \mathbb{R}^n$, $y_i \in \mathbb{R}^m$, $i = 1, \ldots, N$. The ELM with $T$ hidden nodes can be formulated as follows:

$$f(x) = \sum_{t=1}^{T} \beta_t h_t(x) = \mathbf{h}(x)\mathbf{\beta}$$

(4)

where $\mathbf{\beta} = [\beta_1, \ldots, \beta_T]^{-}\in \mathbb{R}^{T\times m}$ ($T$ and $m$ are the number of nodes of the hidden layer and output layer, respectively) is the output weights between the hidden layer and the output layer, $\mathbf{h}(x) = [h_1(x), \ldots, h_T(x)]$ is the output vector of the hidden layer with respect to the input $x$. Intuitively, $\mathbf{h}(x)$ is a feature mapping, which maps the input $x$ from $n$-dimensional input space to the $T$-dimensional hidden layer feature space (ELM feature space) $\mathbf{H}$.

According to Bartlett’s theory [38], the smaller the norm of weights, the better the generalization performance of networks, and the smaller the training errors. That is, minimizing the norm of the output weights actually involves maximizing the distance between the separating margins of different domains in the feature space. Therefore, different from traditional intelligent learning algorithms, ELM aims to minimize the training errors and the norm of the output weights simultaneously. This is shown in the following:

$$\text{Minimize} : \mathcal{L} = \|\mathbf{H}\mathbf{\beta} - \mathbf{Y}\|^2 + \|\mathbf{\beta}\|$$

(5)

where $\mathbf{H}$ is the output matrix of the hidden layer is shown in the following:

$$\mathbf{H} = \begin{bmatrix} h(x_1) \\ \vdots \\ h(x_N) \end{bmatrix} = \begin{bmatrix} h_1(x_1) & \cdots & h_T(x_1) \\ \vdots & \vdots & \vdots \\ h_1(x_N) & \cdots & h_T(x_N) \end{bmatrix}$$

(6)

To solve (5), the minimal norm least square method is typically used, and the solution is written as follows:

$$\mathbf{\beta} = \mathbf{H}^\dagger \mathbf{Y}$$

(7)

where $\mathbf{H}^\dagger$ is the Moore–Penrose generalized inverse of $\mathbf{H}$, $\mathbf{Y} = [y_1, \ldots, y_N]^\dagger \in \mathbb{R}^{N\times m}$. Different methods can be used to calculate the Moore–Penrose generalized inverse of $\mathbf{H}$, e.g., orthogonalization, orthogonal projection, or singular value decomposition. Here we use the orthogonal projection method, which has two forms: 1) if $\mathbf{H}^\dagger \mathbf{H}$ is nonsingular, $\mathbf{H}^\dagger = (\mathbf{H}^\dagger \mathbf{H})^{-1} \mathbf{H}^\dagger$ or 2) if $\mathbf{H} \mathbf{H}^\dagger$ is nonsingular, $\mathbf{H}^\dagger = (\mathbf{H} \mathbf{H}^\dagger)^{-1} \mathbf{H}$. Therefore, (7) can be rewritten as follows:

$$\mathbf{\beta} = \mathbf{H}^\dagger (\mathbf{H} \mathbf{H}^\dagger)^{-1} \mathbf{Y}$$

(8)

Finally, the ELM can be written as follows:

$$f(x) = \mathbf{h}(x)\mathbf{\beta} = \mathbf{h}(x) (\mathbf{H} \mathbf{H}^\dagger)^{-1} \mathbf{Y}$$

(9)

$$f(x) = \mathbf{h}(x)\mathbf{\beta} = \mathbf{h}(x) (\mathbf{H}^\dagger \mathbf{H})^{-1} \mathbf{H}^\dagger \mathbf{Y}.$$
optimization objective of our method can be formulated as follows:

$$L = \frac{1}{2} \lambda \|\beta\|^2 + \frac{1}{2} \sum_{i=1}^{N} \|\xi_i\|^2$$

subject to

$$h(x_i)\beta - y_i^T = \xi_i^T$$  \hspace{1cm} (14)

where $i = 1, \ldots, N$ ($N$ is the number of training frames) and $\xi_i = [\xi_{i,1}, \ldots, \xi_{i,m}]^T$ is the training error vector of the training sample $x_i$. According to the Lagrange theorem, training our method is equivalent to solving the following optimization object:

$$L = \frac{1}{2} \lambda \|\beta\|^2 + \frac{1}{2} \sum_{i=1}^{N} \|\xi_i\|^2$$

$$- \sum_{i=1}^{N} m \sum_{j=1}^{m} \alpha_{ij}(h(x_i)\beta_j - y_{ij} + \xi_{ij})$$  \hspace{1cm} (15)

where $\beta_j$ is the vector of the weights that links the hidden layer to the $j$th output node and $\beta = [\beta_1, \ldots, \beta_m]$, and $\alpha_{ij}$ is the Lagrange multiplier corresponding to the $j$th output of the $i$th training sample. Then, we have the following KKT corresponding optimality conditions:

$$\frac{\partial L}{\partial \beta_j} = 0 \rightarrow \lambda \beta_j = \sum_{i=1}^{N} \alpha_{ij}h(x_i)^T \rightarrow \beta = \frac{1}{\lambda}H^T\alpha$$  \hspace{1cm} (16)

$$\frac{\partial L}{\partial \xi_i} = 0 \rightarrow \alpha_i = \xi_i$$  \hspace{1cm} (17)

$$\frac{\partial L}{\partial \alpha_i} = 0 \rightarrow h(x_i)\beta - y_i^T + \xi_i^T = 0$$  \hspace{1cm} (18)

where $\alpha_i = [\alpha_{i,1}, \ldots, \alpha_{i,m}]^T$ and $\alpha = [\alpha_1, \ldots, \alpha_N]^T$.

We then substitute (16) and (17) into (18), which can be written as follows:

$$\left(1 + \frac{1}{\lambda}HH^T\right)\alpha = Y \rightarrow \alpha = \left(1 + \frac{1}{\lambda}HH^T\right)^{-1}Y$$  \hspace{1cm} (19)

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

$$\beta = \frac{1}{\lambda}H^T\left(I + \frac{1}{\lambda}HH^T\right)^{-1}Y$$

$$\beta = H^T\left(\lambda I + HH^T\right)^{-1}Y.$$

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

$$f(x) = h(x)^T\left(\lambda I + HH^T\right)^{-1}Y.$$  \hspace{1cm} (21)

By combining (1) and (21), the transition function of our method can be rewritten as follows:

$$f(x) = \begin{bmatrix} K(x, x_1) \\ \vdots \\ K(x, x_N) \end{bmatrix}^T (\lambda I + \Omega_{KSM})^{-1}Y.$$  \hspace{1cm} (22)

Here, $x$ is the test frame, $[x_1, \ldots, x_N]$ is with respect to the elements of the explanatory frames, and $Y$ is with respect to the response frames. From (22), we find that the proposed model is only related to the kernel function, input data $x$ and the number of training samples. The kernel similarity embedding is not related to the number of outputs nodes. Thus, our method does not need to artificially select $h(x)$ and implicitly maps the input data to a high-dimensional space. Furthermore, the feature mapping $h(x)$ and dimensionality of the feature space (nodes of the hidden layer) are unknown to the user; instead, the corresponding kernel $K(u, v)$ is given. Moreover, (22) intuitively shows that the kernel similarity matrix is incorporated into the kernel similarity embedding, which will effectively use the similarity as prior information for representing DTs.

See Algorithm 1 for a description of the proposed DT synthesis method. Specifically, the algorithm first divides the input video sequence $S_{1:N}$ (after subtracting the temporal mean $\bar{S}$) into two subsequences: 1) explanatory frames $[S_j, j = 1, \ldots, N - 1]$ and 2) response frames $[S_k, k = 2, \ldots, N]$. Then, the kernel similarity matrix $\Omega_{KSM}$ is learned with respect to (12) and (13), where it is incorporated into kernel similarity
as shown in Fig. 3. The kernel similarity matrices of each row frames for each sequence) in the Dyntex dataset after training, learned kernel similarity matrices of some DT sequences (200 ding. To intuitively analyze this mechanism, we visualize the matrix, which is incorporated into the kernel similarity embedded DT synthesis model to learn kernel similarity embeddings for achieving this goal.

Moreover, this cue can be further elaborated as the similarity in the spatial domain and stochastic repetitiveness in the temporal dimension, which is the key cue for distinguishing DT videos of the same class. (a) Elevator. (b) Rotating wind ornament. (c) Flowers swaying with current. (d) Waterfall. (e) Water wave. (f) Spring water.

Our method represents such features with a kernel similarity matrix, which is incorporated into the 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 Fig. 3. The kernel similarity matrices of each row in Fig. 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 exist in everyday surroundings. The different DT sequences of the same class are acquired with different views or different periods. As can be seen from Fig. 3, the kernel similarity matrices produced by our method elegantly represent the similarity correlations of DT videos. Specifically, the repetitiveness and stationarity of the DT of the elevator, rotating wind ornament, and flowers swaying with current are clearly exhibited by the learned matrices. As for the waterfall, water wave, and spring water, although these objects do not originally have obvious repetitiveness influenced by natural factors, the kernel similarity matrices consistently exhibit the statistical stationarity and similarity for different DT videos of the same class. To this end, kernel similarity embedding effectively mines and exploits the similarity as prior knowledge for representing DT using kernel similarity matrices, overcoming the challenge of high-dimensionality and small sample issues. That is, the feature representations produced by our method are discriminative, enabling the proposed method to generate high-fidelity, and long-term DT videos. Additionally, these results also show that our hypotheses are not limited to frames at several time points in the same time interval more similar.

2) Theoretical Insight: Kernel similarity embedding for DT synthesis can be viewed as the kernel embedding of the conditional distribution for regression problems, where the feature

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], [9], [16]. Moreover, this cue can be further elaborated as the similarity correlation between frames, as shown in Fig. 1. Therefore, what we need to do for DT synthesis is to build a DT model for representing the features of the dynamic 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 embeddings for achieving this goal.

Our method represents such features with a kernel similarity matrix, which is incorporated into the 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 Fig. 3. The kernel similarity matrices of each row in Fig. 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 exist in everyday surroundings. The different DT sequences

Algorithm 1 Kernel Similarity Embedding for DT Synthesis

Input:
(1) Training video sequences \{S_l, l = 1, \ldots, N\}
(2) Number of synthesized image sequences \(L\)
(3) Kernel function \(K(u, v)\)

Output:
(1) Synthesized image sequences \{\tilde{S}_l, l = 1, \ldots, L\}

1: Calculate the temporal mean \(\overline{S}\) of \(S_l\).
2: Let \(S_l \leftarrow S_l - \overline{S}, l = 1, \ldots, N\).
3: Initialize \{\tilde{S}_l\}, for \(l = 1, \ldots, L\).
4: Define explanatory frames \{\(S_j, j = 1, \ldots, N - 1\)\} and response frames \{\(S_k, k = 2, \ldots, N\)\} using training video sequences.
5: Calculate \(\Omega_{KSM}\) and \((\lambda I + \Omega_{KSM})^{-1}Y\) according to Eq. (12) and Eq. (13).
6: repeat
7: Calculate \(\tilde{S}_l = f(\tilde{S}_{l-1})\) by Eq. (22), \(l > 1\).
8: Let \(\tilde{S}_l \leftarrow \tilde{S}_l + \overline{S}\).
9: Let \(l \leftarrow l + 1\)
10: until \(l = L\)
vector $\Phi = [\phi(y_1), \ldots, \phi(y_N)]^\top$ ($\Phi$ is a mapping function) in the reproducing kernel Hilbert space (RKHS) is substituted by $Y = [y_1, \ldots, y_N]^\top$ in the original data domain. The key idea is to map conditional distributions into infinite-dimensional feature spaces using kernels, such that all the statistical features of arbitrary distributions and high-dimensional data can ultimately be captured [40], [41]. Its formulation is as follows:

$$\hat{Y}_{x} = \sum_{i=1}^{N} \phi(y_i)W_i(x) = W(x)\Phi$$

$$= K(x)(G + \lambda I)^{-1}\Phi$$

where $K(x) = [k(x, x_1), \ldots, k(x, x_N)]$, $G$ is the Gram matrix for samples from variable $X$, and $W(x) = [W_1, \ldots, W_N]^\top$ is a nonuniform weight vector determined by the value $x$ of the conditioning variable. Indeed, this nonuniform weight vector reflects the effects of conditioning on the embedding. As for the kernel similarity embedding, it can be rewritten as (24) according to (22):

$$f(x) = \begin{bmatrix}
K(x, x_1) \\
\vdots \\
K(x, x_N)
\end{bmatrix}^\top
(\lambda I + \Omega_{KSM})^{-1}Y$$

$$= W(x)Y.$$

It is obvious that the synthesized frames are conditioned by training sample $x$. When comparing (23) and (24), we observe that the kernel similarity embedding for DT is similar to the kernel embedding for the conditional distribution. The difference is that the kernel similarity embedding for DT synthesis predicts the future frames $y_i$ in the original data domain, while the kernel embedding for the conditional distribution predicts the feature $\phi(y_i)$ in RKHS. To this end, kernel similarity embedding possesses the properties of kernel embedding for conditional distributions. Thus, it will effectively 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 provide the implementation details and parameter settings. Furthermore, we intuitively analyze the sustainability and generalization of our method. Finally, we demonstrate, by visual evaluation, time consumption, and quantitative evaluation, that our method is superior to nine baseline methods, including both non-neural-network-based and neural-network-based DT synthesis methods.

A. Implementation Details

For the following experiments, the DT videos were collected from the Internet and two benchmark datasets, i.e., Gatech Graphcut Textures$^2$ [42] and Dytex$^3$ [43]. These two benchmark datasets are publicly available and have been widely used in recent publications [1], [5], [6], [9], [12]–[17]. We resize the frame size of all DT videos to 150 x 100 pixels, following to [5] for direct comparison. Moreover, we train our model using the first 59 to 200 frames because the length of the shortest observed DT sequence is 59. We then synthesize a new frame with the long-term frames for quantitative evaluation. All the experiments presented in this article are conducted in MATLAB 2018b under Windows 10 with a 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) [44] and structural similarity (SSIM) [45]. These are common quantitative evaluation metrics in static image generation [7], [46]–[48], DT synthesis [6], [17], [49], and other future frame prediction problems [18], [50]. Formally, PSNR can be written as follows:

$$PSNR = \frac{1}{L - 1} \sum_{i=2}^{L} 10 \log_{10} \frac{255^2}{MSE(S_l - \hat{S}_l)}$$

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

SSIM was originally designed for image quality assessment, and later used to provide a perceptual judgment on similarity between videos. It can be formulated as follows:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{\mu_x^2 + \mu_y^2 + C_1(\sigma_x^2 + \sigma_y^2 + C_2)}$$

where $x$ and $y$ are the frames of the observed sequence ($S$) and generated sequence ($\hat{S}$), respectively; $\mu_x$ and $\mu_y$ are the local means; $\sigma_x$ and $\sigma_y$ are the standard deviations; $\sigma_{xy}$ is the cross-covariance for frame $x$ and $y$; and $C_1$ and $C_2$ are smoothing factors. However, the SSIM in (26) is used for evaluating the similarity between two frames. To evaluate the whole video sequence, the mean of SSIM is used, as shown in the following:

$$SSIM(S, \hat{S}) = \frac{1}{L - 1} \sum_{i=2}^{L} SSIM(S_l, \hat{S}_l)$$

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

B. Experiment 1 (Hyper-Parameters Selection)

In principle, there are two hyper-parameters influencing the performance of our proposed method: 1) the kernel function $K(u, v)$ [in (22)] and the regularization factor $\lambda$ of the kernel similarity embedding [in (22)]. As the kernel function selection is important for kernel learning [5], [51]–[54] and it can directly affect the stability of our method, we comprehensively test the effects on the overall performance in order to obtain the optimal solution. Moreover, since the regularization factor $\lambda$ and kernel size $\gamma$ interfere with the learning stability for the kernel similarity embedding and impair the generalization

\[http://www.cc.gatech.edu/cpl/projects/graphcuttextures\]
\[http://projects.cwi.nl/dytex/database.html\]
To achieve good generalization performance, the regularization factor $\lambda$ and the kernel size $\gamma$ of the model need to be chosen appropriately. We investigate a wide range of $\lambda$ and $\gamma$. Specifically, we use 21 different values of $\lambda = \{2^{-30}, 2^{-28}, \ldots, 2^{6}, 2^{10}\}$ and 19 different values of $\gamma = \{2^{-18}, 2^{-16}, \ldots, 2^{16}, 2^{18}\}$ for evaluation on the Dyntex dataset, resulting in a total of 399 pairs of $(\lambda, \gamma)$, as shown in Fig. 6. We find that the performance of our method becomes stable after two-time periods (period 1: $\lambda < 10^{-14}$ and period 2: $\lambda > 10^{2}$), which indicates that our method is over-fitting and under-fitting, respectively. Note that when $\lambda$ is small, the PSNR cannot be stable [see Fig. 6(a)] because $PSNR \propto +\infty$ when some generated frames are extremely similar to observed frames. The regularization of the model is insufficient if the regularization factor $\lambda$ is too small, which results in the DT synthesis model being overly confident for the training frames (the first 200 frames) and failing to generate high-quality DT frames after 200 frames. However, if too large $\lambda$ is used, the model is over-regularized, which leads to the stationarity and repetitiveness of the DT being overly smooth. That is, the weak correlations between different frames decrease excessively (see Fig. 7). Fig. 6 further shows that the kernel size $\gamma$ also closely interferes with the regularization ability of $\lambda$. We can observe that the PSNR and SSIM are not sensitive to $\gamma$.
Fig. 7. Demonstrating the regularization ability of regularization factor $\lambda$. We display the corresponding kernel similarity matrices on two samples with different $\lambda$ values: ($\lambda = 2^{-20}$ shows under-regularization; $\lambda = 2^4$ shows over-regularization). (a) Windmill. (b) Rotating.

when $\lambda$ is small, but a relatively large $\gamma$ seems to yield better results. Therefore, the optimal combination of $(\lambda, \gamma)$ 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 a synthesis method with good sustainability performance, i.e., no obvious visual decay, divergence, or abrupt jumps for generated long-term sequences. Therefore, we intuitively analyze 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 different classes is presented in Fig. 8. To demonstrate the robustness of our method, we train the model using the first 200 frames if the length $l$ of observed sequences is longer than 200, and otherwise, all frames of observed sequences are used. We observe that our method not only generates high-fidelity DTs in the short term, but also generates high-quality DTs in the 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 ($l = 848$), and windmill ($l = 962$). Note that Fig. 8(a) seemingly shares with a similar DT (from 100th to 1000th), because these generated frames are located in a similar/same cycle. Furthermore, our method still keeps synthesizing realistic DTs (including the details) in the long term, e.g., the flag in the “windmill” sample exhibits its dynamics in the long-term generated sequences.

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 the six observed sequences used in the former visual quality evaluation. As can be seen from Fig. 9, our method achieves desirable mean PSNR and SSIM, which shows that it synthesizes high-fidelity DTs. Although the mean PSNR and SSIM decrease as the number of generated frames increases for some DT videos (e.g., windmill and frame), they are still large ($PSNR > 18$ dB and $SSIM > 0.69$). Notably, our method achieves a high SSIM index of 1 for whole long-term sequences just using 200 frames for training (e.g., flowers swaying with current, bulb, and water wave), which suggests that it generates videos that are nearly the same as the observed sequences. These results prove that our method accurately exhibits the statistical stationarity in the spatial domain and the stochastic repetitiveness in the temporal dimension of DT sequences using kernel similarity embedding, and can thus synthesize realistic DTs in the long term.

D. Experiment 3 (Generalization Analysis)

Good generalization performance is a key goal for all learning tasks. Similar to [17], we also specialize in our method to learn roughly aligned DT videos, which are nonstationary in either the spatial or temporal domain. This study differs...
from [17], however, which trains a model using all roughly aligned video sequences for one example (e.g., five training sequences for the running cow). Our method instead trains a model using only one video sequence for each example, which confirms 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 used in [17] for evaluation. As can be seen from Fig. 10, the three results of modeling and synthesizing DTs from roughly aligned video sequences are displayed. Specifically, we first train 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 obtain 33 realistic, synthesized sequences.

The experimental 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 for the test animals, which suggests that it has high generalization ability. However, the transferred model cannot synthesize consistent motions for some cows (e.g., cow1→cow2), because these samples are not initially aligned well.

E. Experiment 4 (Comparisons to State-of-the-Art)

In this section, we compare our method with nine state-of-the-art methods for DT synthesis, including non-neural-network-based methods (LDS [1], FFT-LDS [14], HOSVD [12], SLDS [15], Kernel-DT [13], and KPCR [5]) and neural-network-based methods (TwoStream4 [9], STGCN5 [17], and DG6 [10]). To better verify and validate the performance of our method, we simultaneously leverage quantitative evaluation metrics (SSIM and PSNR), time consumption and vision quality. Specifically, we first compare our method with six non-neural-network-based methods, including FFT-LDS [14], HOSVD [12], KPCR [5], LDS [1], SLDS [15], and Kernel-DT [13]. To facilitate direct comparison, we test all models with 150×100 pixels using 17 gray DT videos, in terms of PSNR and SSIM. As can be seen from Table I, our method attains the best performance on most of DT sequences, except for the videos of flashing lights and beach. This is because these two videos lack the good spatial-temporal characteristics 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 a significant SSIM index of 1 for some DT videos (e.g., bulb, fountain, spring water, etc.). Therefore, our method clearly makes full use of the similarity as prior knowledge for DT synthesis using kernel similarity embedding. However, all methods fail to synthesize high-quality DT videos for the rotating wind ornament, because this sample is originally blurry.

Next, our model is compared with three neural-network-based methods, including TwoStream [9] and STGCN [17] and DG [10] on six DT videos (e.g., elector, flashlights, and beach). For a fair comparison, we display six same index frames of generated sequences. As shown in Fig. 11, the DT sequences generated by TwoStream are divergent because this method cannot effectively synthesize DTs that are not spatially homogeneous (e.g., elevator and water spray). As for STGCN and DG, the DT sequences generated by these appear blurred because these two methods rely on more training data.

4https://ryersonvisionlab.github.io/two-stream-projpage/
5http://www.stat.ucla.edu/jxie/STGConvNet/STGConvNet.html
6http://www.stat.ucla.edu/jxie/DynamicGenerator/DynamicGenerator.html
TABLE I

Comparison With Non-Neural-Network-Based DT Synthesis Methods in Terms of PSNR (dB) and SSIM

|                  | Ours  | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM |
|------------------|-------|------|------|------|------|------|------|------|------|------|------|
| boiling water    | 36.591| 0.958| 27.570| 0.987| 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| 34.340| 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| 29.279| 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| 27.657| 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| 26.635| 0.773| 26.489| 0.772| 24.474| 0.748|

Fig. 11. Visual quality comparison between three neural-network-based methods and our method for six different DT videos. For each category, the first row displays six frames of the observed sequence, and the other rows display the corresponding frames of the sequences generated by different methods (from top to bottom: TwoStream [9], STGCN [17], DG [10], and our method). (a) Elevator. (b) Flowers swaying. (c) Flash lights. (d) Water wave. (e) Spring water. (f) Water spray.

Intuitively, our method generates high-fidelity DT sequences, including realistic details of DTs.

Finally, we report the time consumption of different DT synthesis methods, including neural-network-based and non-neural-network-based methods. As shown in Table II, our method non-neural-network-based methods (except for Kernel-DT) achieve real-time (25 fps) generation with 46.040 fps, while neural-network-based methods fail. Moreover, the neural-network-based methods are time consuming and computationally expensive to train. In summary, our method synthesizes high-quality DT videos with high speed and low computation, making it superior to the state-of-the-art DT methods. This can be attributed to its 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 critical prior knowledge for DT synthesis.

V. DISCUSSION

In this study, we propose a novel DT synthesis method to address the high-dimensionality and small sample issues of DT synthesis. Specifically, our method leverages a kernel similarity matrix to mine and capture the similarity as prior knowledge of DT, which is incorporated into the kernel similarity embedding. Then, high-fidelity DTs are synthesized iteratively by the learned model. Notably, our method is dissimilar to the existing kernel-based DT synthesis methods [5], [13], which use the kernel function to learn a nonlinear observation function for dimensionality reduction. The experimental results on several benchmarks show that the similarity correlation is critical prior knowledge for representing DTs and the kernel similarity embedding effectively solves the aforementioned issues. Thus, our method can achieve promising results for DT synthesis.

To evaluate our method, we intuitively and theoretically analyzed the effectiveness of the kernel similarity embedding for DT synthesis (Section III-C). Then, we evaluated the influence of the kernel function, the regularization factor $\lambda$, and the kernel size $\gamma$ (Section IV-B). Meanwhile, we intuitively validated the sustainability and generalization of our method using vision quality and quantitative evaluation metrics (Sections IV-C and IV-D). Finally, we compared our method to nine state-of-the-art methods (including neural-network-based and non-neural-network-based methods) (Section IV-E).

Although our method achieves significant results, it is limited in that its sustainability is impacted if the DTs lack...
and the model parameters could be jointly learned. Accordingly, the optimal combination coefficients combined as a novel similarity matrix with a set of combination to learn multiple kernel features, which could then be com-

To achieve this goal, we argue that multiple kernel learning
repetitiveness in the temporal dimension can be well learned.
The statistical stationarity in the spatial domain and stochastic repetitiveness in the temporal dimension may be a potential choice to overcome this limitation. Because scalable kernel similarity embedding can learn an optimal similarity representation for various data, and thus the statistical stationarity in the spatial domain and stochastic repetitiveness in the temporal dimension can be well learned. To achieve this goal, we argue that multiple kernel learning (MKL) [52]–[57] may be a good solution for constructing the scalable kernel similarity matrix by learning multiple kernel features. Specifically, we could employ a group of base kernels to learn multiple kernel features, which could then be combined as a novel similarity matrix with a set of combination coefficients. Accordingly, the optimal combination coefficients and the model parameters could be jointly learned.

VI. CONCLUSION

In this article, we have proposed a novel DT synthesis method that integrates kernel learning and the ELM into a powerfully unified synthesis method to learn kernel similarity embeddings for representing DTs. Notably, kernel similarity embedding not only effectively addresses the high-dimensionality and small sample issues by using similarity as prior knowledge, but also has the advantage of modeling nonlinear feature representation relationships for DT. The competitive results on DT videos collected from two benchmark datasets and the Internet demonstrate the superiority and great potential 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 an MKL model to learn a scalable kernel similarity embedding for DT synthesis, which will enable the similarity representations of DT to effectively be controlled. Furthermore, we will also incorporate multi-view methods into DT synthesis, as some DT sequences were acquired with moving cameras and different views (e.g., some DT videos in Dyntex).

TABLE II

|                 | Ours | FFT-LDS [12] | HO-SVD [12] | KPCR [5] | LDL [1] | SLDS [15] | Kernel-DT [13] | TwoStream [9] | STOCN [17] | DO [10] |
|-----------------|------|--------------|-------------|----------|---------|-----------|----------------|---------------|------------|---------|
| Train. time (Sec.) | 0.090 | 0.928 | 1.399 | 1.990 | 0.148 | 2.475 | 0.830 | - | 4.188 | 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        | x    | x    | x    | x    | x    | x    | x | ✓ | ✓ | ✓ |
| FPS             | 46.040 | 217.560 | 243.790 | 98.248 | 296.450 | 315.900 | 1.191 | 0.002 | 9.709 | 2.295 |

Fig. 12. Some generated frames of DT sequences displaying the limitations of our method. For each category, the first row displays the 11 frames of the observed sequence (black frame indicates that the corresponding frame of the observed sequence is lacking), and the second row shows the corresponding frames of videos synthesized by our method. From left to right, the columns are the 2nd, 100th, 200th, 300th, 400th, 500th, 600th, 700th, 800th, 900th, and 1000th frames of the observed and synthesized sequences. (a) Waterfall. (b) Spring water. (c) Flag.

stochastic repetitiveness in the temporal dimension (e.g., waterfall and spring water). As can be seen from Fig. 12, our method falls into visual decay after 200 frames (the length of the training frames), and thus fails to synthesize high-fidelity DT in the long term. In the future, scalable kernel similarity embedding may be a potential choice to overcome this limitation.

REFERENCES

[1] G. Doretto, A. Chiuso, Y. N. Wu, and S. Soatto, “Dynamic textures,” Int. J. Comput. Vis., vol. 51, no. 2, pp. 91–109, 2003.
[2] C. Feichtenhofer, A. Pinz, and R. P. Wildes, “Temporal residual networks for dynamic scene recognition,” in Proc. CVPR, 2017, pp. 4728–4737.
[3] J. Mei, B. Gao, D. Xu, W. Yao, X. Zhao, and H. Zhao, “Semantic segmentation of 3D LiDAR data in dynamic scene using semi-supervised learning,” IEEE Trans. Intell. Transp. Syst., vol. 21, no. 6, pp. 2496–2509, Jun. 2020.
[4] C.-C. Hsu, L.-W. Kang, and C.-W. Lin, “Temporally coherent supersolution of textured video via dynamic texture synthesis,” IEEE Trans. Image Process., vol. 24, pp. 919–931, 2015.
[5] X. You, W. Guo, S. Yu, K. Li, J. C. Principe, and D. Tao, “Kernel learning for dynamic texture synthesis,” IEEE Trans. Image Process., vol. 25, pp. 4782–4795, 2016.
[6] J. Xie, S.-C. Zhu, and Y. N. Wu, “Synthesizing dynamic patterns by spatial-temporal generative ConvNet,” in Proc. CVPR, 2017, pp. 7093–7101.
[7] Q. Chen and V. Koltun, “Photographic image synthesis with cascaded refinement networks,” in Proc. ICCV, 2017, pp. 1511–1520.
[8] J. Xie, Y. Lu, R. Gao, S.-C. Zhu, and Y. Wu, “Cooperative training of descriptor and generator networks,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 42, no. 1, pp. 27–45, Jan. 2020.
[9] M. Tesfaldet, M. A. Brubaker, and K. G. Derpanis, “Two-stream convolutional networks for dynamic texture synthesis,” in Proc. CVPR, 2018, pp. 6703–6712.
[10] 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.
[11] V. Pogoraro and S. G. Parker, “Physically-based realistic fire rendering,” in Proc. NPH, 2006, pp. 51–59.
[12] R. Costantini, L. Sbaiz, and S. Süssbrunk, “Higher order SVD analysis for dynamic texture synthesis,” IEEE Trans. Image Process., vol. 17, pp. 45–52, 2008.
[13] A. B. Chan and N. Vasconcelos, “Classifying video with kernel dynamic textures,” in Proc. CVPR, 2007, pp. 1–6.
[14] B. Ahlbrahim, O. I. Camps, and M. Sznier, “Dynamic texture with fourier descriptors,” in Proc. 4th Int. Workshop Texture Anal., 2005, pp. 53–58.
[15] 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.
[16] L. A. Gatys, A. S. Ecker, and M. Bethge, “Texture synthesis using convolutional neural networks,” in Proc. NeurIPS, 2015, pp. 262–270.
[17] J. Xie, S.-C. Zhu, and Y. Wu, “Learning energy-based spatial-temporal generative ConvNets for dynamic patterns,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 43, no. 2, pp. 516–531, Feb. 2021.
Shiming Chen is currently pursuing the Ph.D. degree in information and communication engineering with the Huazhong University of Sciences and Technology, Wuhan, China.

His research results have expounded in more than ten publications at prestigious journals and prominent conferences, such as IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, NeurIPS, CVPR, ICCV, AAAI, and IJCAI. His current research interests span computer vision and machine learning with a series of topics, such as generative modeling and learning, and zero-shot learning.

Mr. Chen serves as the Reviewer for prestigious journals, such as IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, IEEE TRANSACTIONS ON IMAGE PROCESSING, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS; IEEE TRANSACTIONS ON CYBERNETICS, IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS, AND INFORMATION FUSION.
Peng Zhang received the Ph.D. degree in information and communication engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2020. His research interests include computer vision, pattern recognition, and machine learning.

Guo-Sen Xie received the Ph.D. degree in computer science from the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2016. He was with the Inception Institute of Artificial Intelligence, Abu Dhabi, UAE. His research results have expounded in more than 20 publications at prestigious journals and prominent conferences, such as IEEE TRANSACTIONS ON IMAGE PROCESSING, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, CVPR, and ECCV. His research interests include computer vision and machine learning.

Qinmu Peng received the Ph.D. degree in computer science from Hong Kong Baptist University, Hong Kong, in 2015. He is currently an Assistant Professor with the School of Electronics Information and Communications, Huazhong University of Science and Technology, Wuhan, China. His research results have expounded in more than 20 publications at prestigious journals and prominent conferences, such as Proceedings of the National Academy of Sciences of the United States of America, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS, IEEE TRANSACTIONS ON HUMAN-MACHINE SYSTEMS, IEEE TRANSACTIONS ON MULTIMEDIA, and IJCAI. His current research interests include multimedia analysis, computer vision, and medical image analysis.

Zehong Cao received the Ph.D. degree in information technology from the University of Technology Sydney, Ultimo, NSW, Australia, in 2017. He is an Associate Professor with the University of South Australia, Adelaide, SA, Australia, and an Adjunct Fellow with the School of Computer Science, University of Technology Sydney. He has authored more than 50 papers published in top-tier International Conferences, such as AAMAS and AAAI, and IEEE and ACM Transactions Series, with five ESI highly cited papers. His current research interests include computer vision, machine learning, computational intelligence, and bio-signal processing.

Dr. Cao is the Leading Guest Editor of IEEE TRANSACTIONS ON FUZZY SYSTEMS and IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS in 2020, and the Associate Editor for Neurocomputing since 2019 and Scientific Data since 2019.

Wei Yuan received the B.S. degree in electronic engineering from Wuhan University, Wuhan, China, in 1999, and the Ph.D. degree in electronic engineering from the University of Science and Technology of China, Hefei, China, in 2006. He is currently a Professor with the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan. His research results have expounded in more than 20 publications at prestigious journals and prominent conferences, such as IEEE TRANSACTIONS ON SIGNAL PROCESSING, IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, IEEE TRANSACTIONS ON CYBERNETICS, IEEE TRANSACTIONS ON SMART GRID, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS, and WWW. His current research interests include machine learning, computer vision, and information security.

Xinge You (Senior Member, IEEE) received the B.S. and M.S. degrees in mathematics from Hubei University, Wuhan, China, in 1990 and 2000, respectively, and the Ph.D. degree from the Department of Computer Science, Hong Kong Baptist University, Hong Kong, in 2004. He is currently a Professor with the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan. His research results have expounded in more than 60 publications at prestigious journals and prominent conferences, such as IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, IEEE TRANSACTIONS ON IMAGE PROCESSING, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, IEEE TRANSACTIONS ON CYBERNETICS, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, CVPR, ECCV, and IJCAI. His current research interests include image processing, wavelet analysis and its applications, pattern recognition, machine learning, and computer vision.

Prof. You served/serves as an Associate Editor of the IEEE TRANSACTIONS ON CYBERNETICS and IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS.