Similarity-DT: Kernel Similarity Embedding for Dynamic Texture Synthesis

Shiming Chen[1], Peng Zhang[1], Xinge You[1], Qinmu Peng[1], Xin Liu[2], and Zehong Cao[3],
[1]Huazhong University of Science and Technology
[2]Huaqiao University [3]University of Technology Sydney
{shimingchen,zp_zhg,youxg,pengqinmu}@hust.edu.cn xliu@hqu.edu.cn zehong.cao@utas.edu.au

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. However, there are no DT synthesis methods to consider the similarity prior for representing DT instead, which can explicitly capture the homogeneous and heterogeneous correlation between different frames of DT. In this paper, we propose a novel DT synthesis method (named Similarity-DT), which embeds the similarity prior into the representation of DT. Specifically, we first raise two hypotheses: the content of texture video frames varies over time-to-time, while the more closed frames should be more similar; the transition between frame-to-frame could be modeled as a linear or nonlinear function to capture the similarity correlation. Then, our proposed Similarity-DT integrates kernel learning and extreme learning machine (ELM) into a powerful unified synthesis model to learn kernel similarity embedding to represent the spatial-temporal transition among frame-to-frame of DTs. 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 effectively exhibits the discriminative representation for DTs. Hence our method is capable of preserving long-term temporal continuity of the synthesized DT sequences with excellent sustainability and generalization. We also show that our method effectively generates realistic DT videos with fast speed and low computation, compared with the state-of-the-art approaches.

1. Introduction

Dynamic texture (DT), which exhibits statistical stationarity in the spatial domain and stochastic repetitiveness in the temporal dimension, is one of the dynamic patterns in video sequences and computer vision [9, 33], e.g., flaming fire, rotating windmill. Due to the demands of DT synthesis in video technology applications (e.g., texture recognition [10], video segmentation [3, 24]), synthesizing DTs has gradually become an interesting and cutting-edge problem in computer vision [5, 12, 30, 37, 38, 40]. The goal of DT synthesis is to infer a generating process from a DT example, which then allows producing arbitrarily many new frames of that texture.

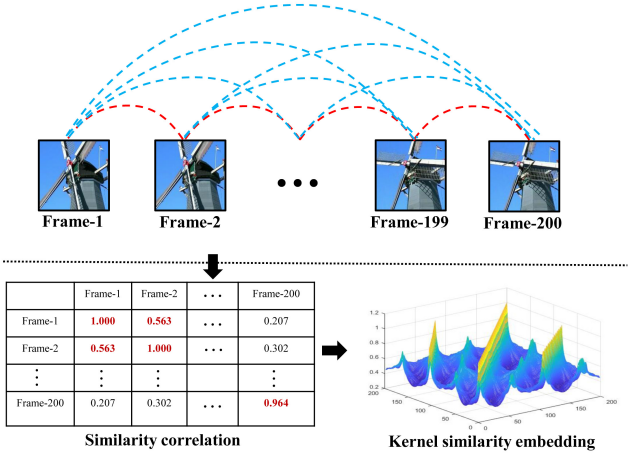


Figure 1. The core idea of our proposed Similarity-DT method. DT exhibits statistical stationarity in the spatial domain and stochastic repetitiveness in the temporal dimension, which means that different frames of DT possess high similarity correlation that can be represented by kernel similarity embedding.

In fact, a similarity correlation between frame-to-frame is an explicit expression of statistical stationarity and stochastic repetitiveness of DT, which is a critical representation that distinguishes DTs from other videos. Similarity representation serves as the learning objective of metric learning for the discriminative model [6, 10, 41], which means that similarity correlation is critical for representation. Some researchers also attempted to mine the potential similarities knowledge of samples to improve the performance of discriminative model [7, 11, 14, 20, 21, 22, 43], which suggests that similarity serves as prior knowledge as important as the class labels and annotation information. Moreover, sim-

ilarity 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 for representing DT, and this is the focus of the present paper.

To make full use of similarity prior, we embed it into the representation of the generative model for DT synthesis. Based on this, we assume that 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 assumptions 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 [27]. 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 exhibited by similarity correlation between frame-to-frame, which can be further elegantly represented by kernel similarity embedding, as demonstrated in Figure 1. Furthermore, extreme learning machine (ELM) as an emergent technology which overcomes some challenges (e.g., slow learning speed, trivial human intervene and poor computational scalability) faced by other computational intelligence techniques, and thus it has recently attracted the attention of more researchers [8, 16, 18, 29]. 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 (named Similarity-DT) to generate high-quality, long-term DT sequences with fast speed and low computation. It integrates kernel learning and ELM into a powerful 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 $\{\boldsymbol{S}_j, j = 1, \cdots, N-1\}$ and response frames $\{\boldsymbol{S}_k, k = 2, \cdots, N\}$. Then, Similarity-DT 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 Similarity-DT.

To summarize, this study makes the following salient contributions:

1) To make full use of similarity prior for representing DTs, we raise two hypotheses, which are essential for the DT model to generate new frames according to current frames using the similarity correlation of different frames.

2) We propose a novel DT synthesis method (Similarity-DT), which integrates kernel learning and extreme learning machine (ELM) into a powerful unified synthesis model to learn kernel similarity embedding to synthesize realistic video sequences with good sustainability.

3) The learned kernel similarity embedding of Similarity-DT elegantly represents the spatial-temporal transition between frame-to-frame of DT. We intuitively analyze and evaluate its availability for DT synthesis.

4) Extensive experiments on two benchmark datasets demonstrate that Similarity-DT shows consistent improvement over the baseline methods with fast generation speed, low computation, and synthesizing high-quality DTs.

2. Related Work

Similarity Prior for Learning. Similarity serves as prior knowledge existing in different samples or self-sample as important as the class labels and 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) [7, 11, 14], content-based image retrieval [20, 21, 22, 43]. For instance, [7, 14] proposed triplet loss that based on similarity and unsimilarity knowledge of different samples for learning discriminative representation. In [11], Fu introduced a simple unsupervised cross domain adaptation approach for person re-ID using self-similarity grouping. In fact, different frames of DT possess a high similarity correlation, which can also be took full advantage for improving the performance of DT synthesis.

Non-neural-network-based DT Synthesis Methods. Dynamic system (DS) modeling methods for DT synthesis [1, 4, 9, 28, 40] are most popular non-neural-network-based methods. DS modeling methods typically learn transition matrix for representing the correlation of different frames of DT, which motivated us to use similarity prior to exhibit the correlation among different frames of DT. In [9], Doretto 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), and then dynamic trajectory was modeled over time. Siddiqi et al. [28] 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) [1]. In [4], Chain introduced a new method (Kernel-DT) for DT synthesis using kernel principal component analysis (KPCA) to learn a nonlinear observation function. In [40], You proposed the kernel principal component regression (KPCR) method to further improve DT synthesis.

Neural-network-based DT Synthesis Methods. Neural network has proven to be an immensely successful discrimi-

native and generative learning machine [6, 26, 31, 39, 41, 42]. In term of DT synthesis, various approaches based on the convolutional neural network (ConvNet) have been proposed [12, 30, 36, 37, 38]. In [30], Tesfaldet proposed a two-stream model for DT synthesis using a set of Gram matrices. In [37, 38], Xie proposed an energy-based spatial-temporal generative ConvNet to model and to synthesize dynamic patterns. [36] presented a dynamic generator model using alternating back-propagation through time algorithm for DT synthesis. In summary, neural-neural-based methods are expert in synthesizing impressive DTs, while they are time-consuming and computationally expensive. Therefore, an extreme learning machine may be the desired successor for DT synthesis with expected generation performance at a surprising learning speed.

3. Method

3.1. Revisiting Extreme Learning Machine

ELM was originally proposed by Huang et al. [18] in 2004, and it serves as an emergent technology that has recently attracted the attention of more researchers[8, 16, 18, 29]. ELM works for generalized single-hidden layer feed-forward networks (SLFNs). Its essence is that the hidden layer of SLFNs needs not be tuned, which means that the feature mapping between the input layer and hidden layer is randomly assigned. 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 for DT synthesis with fast speed.

Before introducing ELM formally, we define some 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 Eq. (1):

$$f_L(\mathbf{x}) = \sum_{j=1}^{L} \beta_j h_j(\mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\beta} \tag{1}$$

where $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_L]^{\mathrm{T}}$ 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 H.

According to Bartlett's theory [2], ELM is to minimize the training errors and the norm of the output weights simultaneously. That is shown in Eq. (2).

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

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

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}(\mathbf{x}_1) \\ \vdots \\ \mathbf{h}(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} h_1(\mathbf{x}_1) & \cdots & h_L(\mathbf{x}_1) \\ \vdots & \vdots & \vdots \\ h_1(\mathbf{x}_N) & \vdots & h_L(\mathbf{x}_N) \end{bmatrix} \tag{3}$$

To solve Eq. (2), the minimal norm least square method is typically used, and the solution is written as Eq. (4).

$$\boldsymbol{\beta} = \boldsymbol{H}^{\dagger}\boldsymbol{Y} \tag{4}$$

where $\boldsymbol{H}^{\dagger}$ is the Moore-Penrose generalized inverse of $\mathbf{H}$, $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_n] \in \mathbb{R}^{m \times n}$. Here we caculate $\boldsymbol{H}^{\dagger}$ using orthogonal projectional method, which can be used in two cases: 1) if $\mathbf{H}^{\mathrm{T}}\mathbf{H}$ is nonsingular, $\mathbf{H}^{\dagger} = \left(\mathbf{H}^{\mathrm{T}}\mathbf{H}\right)^{-1}\mathbf{H}^{\mathrm{T}}$, or 2) if $\mathbf{H}\mathbf{H}^{\mathrm{T}}$ is nonsingular, $\mathbf{H}^{\dagger} = \mathbf{H}^{\mathrm{T}}\left(\mathbf{H}\mathbf{H}^{\mathrm{T}}\right)^{-1}$.

Therefore, Eq. (4) can be rewritten as Eq. (5) or Eq. (6).

$$\boldsymbol{\beta} = \boldsymbol{H}^{T}\left(\boldsymbol{H}\boldsymbol{H}^{T}\right)^{-1}\boldsymbol{Y} \tag{5}$$

$$\boldsymbol{\beta} = \left(\boldsymbol{H}^{T}\boldsymbol{H}\right)^{-1}\boldsymbol{H}^{T}\boldsymbol{Y} \tag{6}$$

Finally, the model of ELM is written as Eq. (7) or Eq. (8).

$$f(\mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\beta} = \mathbf{h}(\mathbf{x})\boldsymbol{H}^{T}\left(\boldsymbol{H}\boldsymbol{H}^{T}\right)^{-1}\boldsymbol{Y} \tag{7}$$

$$f(\mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\beta} = \mathbf{h}(\mathbf{x})\left(\boldsymbol{H}^{T}\boldsymbol{H}\right)^{-1}\boldsymbol{H}^{T}\boldsymbol{Y} \tag{8}$$

Note that, the size of $\mathbf{H}\mathbf{H}^{\mathrm{T}}$ is $N \times N$, and the size of $\mathbf{H}^{\mathrm{T}}\mathbf{H}$ 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. (7) in following section.

3.2. The Proposed Similarity-DT

Presenting from the revisiting in Section 3.1, 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 by the user artificially, which is similar to the selection of transition function of a dynamic system 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, the kernel function embraces an exciting property that it can effectively measure the similarity of different samples, which can elegantly exhibit the similarity correlation between different frames for DT. Therefore, Similarity-DT extends ELM to kernel-ELM with kernel learning (kernel function: $K(\boldsymbol{u}, \boldsymbol{v})$) to learn kernel similarity embedding for representing the spatial-temporal representation of DT.

At first, we define a kernel similarity embedding Ω_{KSE} for Similarity-DT, which is shown in Eq. (9).

$$\boldsymbol{\Omega}_{KSE} = \boldsymbol{H}\boldsymbol{H}^T \tag{9}$$

and

$$\Omega_{KSE_{i,j}} = \mathbf{h}\left(\mathbf{x}_i\right) \cdot \mathbf{h}\left(\mathbf{x}_j\right) = K\left(\mathbf{x}_i, \mathbf{x}_j\right) \tag{10}$$

According to ridge regression theory [15], we can add a regularization factor λ (positive small value) to control the regularization performance of $\|\boldsymbol{\beta}\|$ during optimization, which is unsimilar to [17] that regularizes $\|\mathbf{H}\boldsymbol{\beta}-\mathbf{Y}\|^2$. Then, the optimization object of Similarity-DT is written as:

$$\begin{aligned} &Minimize: L = \frac{1}{2}\lambda\|\boldsymbol{\beta}\|^2 + \frac{1}{2}\sum_{i=1}^{N}\|\boldsymbol{\xi}_i\|^2 \\ &s.t. \quad \mathbf{h}\left(\boldsymbol{x}_i\right)\boldsymbol{\beta} - \mathbf{Y}_i^{\mathrm{T}} = \boldsymbol{\xi}_i^{\mathrm{T}} \end{aligned} \tag{11}$$

where $\boldsymbol{\xi}_i = [\xi_{i,1}, \ldots, \xi_{i,m}]^{\mathrm{T}}$ is the training error vector of the training frame $\mathbf{x}_i$. Thus, a positive diagonal matrix $\lambda\mathbf{I}$ (λ is small) is added to the diagonal axis of $\mathbf{H}\mathbf{H}^{\mathrm{T}}$ for smoothing $\boldsymbol{\Omega}_{KSE}$ during learning output weights $\boldsymbol{\beta}$[1]. Intuitively, if the proper λ is used, the kernel similarity embedding will be more smooth, and thus Similarity-DT will be more stabler and tend to have better generalization. Then, the transition function of Similarity-DT is formulated as Eq. (12) according to Eq. (7).

$$\begin{aligned} f(\mathbf{x}) &= \mathbf{h}(\mathbf{x})\boldsymbol{H}^T\left(\lambda\mathbf{I} + \boldsymbol{H}\boldsymbol{H}^T\right)^{-1}\boldsymbol{Y} \\ &= \begin{bmatrix} K\left(\mathbf{x}, \mathbf{x}_1\right) \\ \vdots \\ K\left(\mathbf{x}, \mathbf{x}_N\right) \end{bmatrix}^T \left(\lambda\mathbf{I} + \boldsymbol{\Omega}_{KSE}\right)^{-1}\boldsymbol{Y} \end{aligned} \tag{12}$$

where $\mathbf{I}$ is identity matrix, $\mathbf{x}$ is 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. (12), we can find that kernel similarity embedding is embedded into the output weight $\boldsymbol{\beta}$, which is the total representation of similarity for DT. Moreover, Similarity-DT needs not to artificially select $\mathbf{h}(\mathbf{x})$ and implicitly maps input data to high dimensional space. To compare Eq. (7) and Eq. (12), it is obvious that kernel-ELM shares similar network structure with ELM and optimizes output weights $\boldsymbol{\beta}$ of ELM using kernel function for learning kernel similarity embedding. Thus, kernel-ELM is more easy to learn a model than ELM and possesses the advantages of ELM.

See Algorithm 1 for a description of the proposed DT synthesis method (Similarity-DT). Specifically, algorithm first divides the input video sequences $\{\boldsymbol{S}_t, t = 1, \cdots, N\}$ (after substracting temporal mean $\overline{S}$) into two sub-sequences:

[1]Proof is attached in Appendix.

Algorithm 1 Similarity-DT

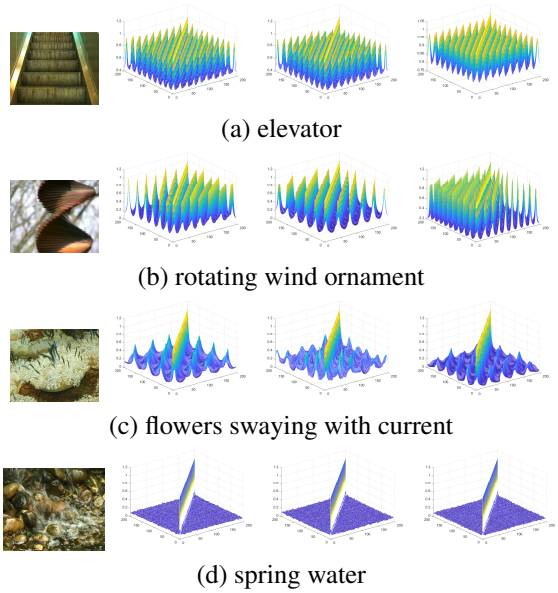
Input:

(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 $\{\tilde{\boldsymbol{S}}_l, l = 1, \cdots, L\}$

Caculate the temporal mean $\overline{S}$ of $\boldsymbol{S}_t$.
Let $\boldsymbol{S}_t \leftarrow \boldsymbol{S}_t - \overline{S}, t = 1, \cdots, N$.
Initialize $\{\tilde{\boldsymbol{S}}_l\}$, for $l = 1, \cdots, L$.
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.
Caculate $\boldsymbol{\Omega}_{KSE}$ and $\left(\lambda\mathbf{I} + \boldsymbol{\Omega}_{KSE}\right)^{-1}\boldsymbol{Y}$ according to Eq. (10).
repeat
  Calculate $\tilde{\boldsymbol{S}}_l = f(\tilde{\boldsymbol{S}}_{l-1})$ by Eq. (12), $l > 1$.
  Let $\tilde{\boldsymbol{S}}_{\boldsymbol{l}} \leftarrow \tilde{\boldsymbol{S}}_l + \overline{S}$.
  Let $l \leftarrow l + 1$
until $l = L$



(a) elevator

(b) rotating wind ornament

(c) flowers swaying with current

(d) spring water

Figure 2. Demonstrating the learned kernel similarity embedding of Similarity-DT elegantly representing the spatial-temporal transition of frame-to-frame of DT. The visualization of kernel similarity embedding in each row are learned from different DT sequences of the same class.

explanatory frames $\{\boldsymbol{S}_j, j = 1, \cdots, N-1\}$ and response frames $\{\boldsymbol{S}_k, k = 2, \cdots, N\}$. Then, the kernel similarity embedding Ω_{KSE} is learned with respect to Eq. (10) and Eq. (9), which is also termed as traning stage. Finally, the endless sequences $\{\tilde{\boldsymbol{S}}_l, l = 1, \cdots, L\}$ (after adding temporal

mean $\overline{S}$) can be generated iteratively with pre-trained model according to Eq. (12).

In fact, the dimensionality D of explanatory frames and response frames is equal, i.e., $n = m = D$. During training, the computational complexity of Similarity-DT 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 Similarity-DT is $\boldsymbol{O}\left(DN\right)$, including N kernel operation and a matrix multiplication.

3.3. Analysis of Kernel Similarity Embedding

DT videos exhibit statistical stationarity in the spatial domain and stochastic repetitiveness in the temporal dimension, which is the critical cue for distinguishing DT videos from other videos and static images [9, 12, 13, 30]. Moreover, this cue can be further described as the similarity correlation between frame-to-frame based on our raised assumptions, as shown in Figure 1. Therefore, what we need to do for DT synthesis is that we should build a DT model to effectively learn the similarity knowledge for expressing the representation of dynamics and texture elements, which are statistically similar and temporally. Here we integrate kernel learning and extreme learning machine into a powerful unified DT synthesis model (Similarity-DT) to learn kernel similarity embedding for achieving this goal.

Similarity-DT represents such features with kernel similarity embedding. See from Eq. (12), we have embedded kernel similarity embedding into the output weight $\boldsymbol{\beta}$ of kernel-ELM. Thus the learned kernel similarity embedding effectively works for representing DT. To intuitively analyze this mechanism, we visualize the learned kernel similarity embedding of some DT sequences (200 frames for each sequence) in the Dyntex dataset after training. See from Figure 2, the learned kernel similarity embeddings of Similarity-DT elegantly represent the spatial-temporal transition of frame-to-frame of DT videos. Specifically, the repetitiveness and stationarity of DT of the elevator, rotating wind ornament, and flowers staying with current, are exhibited by the learned embeddings. As for spring water, although it originally has not obvious repetitiveness influenced by natural factors (e.g., wind), the learned kernel similarity embeddings consistently exhibit the statistical stationarity and similarity for different DT videos of the same class. This demonstrates that the similarity prior among different frames can be effectively expressed by the learned kernel similarity embedding, which is critical for DT synthesis. That is to say, the representation of kernel similarity embedding for DT is discriminative, indicating why our proposed DT synthesis method can generate high-fidelity long-term DT videos.

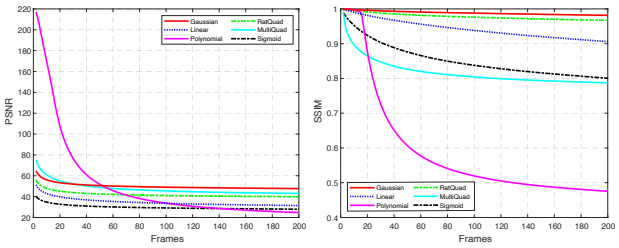


Figure 3. Quantitative comparison of different kernel functions used in our method on Dyntex dataset. we report PSNR (left) and SSIM (right) in terms of the number of generated frames.

4. Experiments

We evaluate our method following standard protocols on two benchmark datasets: Gatech Graphcut Textures[2] [19] and Dyntex[3] [25]. We qualitatively and quantitatively analyze our method and compare it with state-of-the-art dynamic texture synthesis methods. Extensive experiments demonstrate that Similarity-DT synthesizes more long-term, high-quality DT videos, and meanwhile, consistently outperforms the most recent competing algorithms by large margins on Peak Signal-to-Noise Ratio (PSNR) [34], Structural Similarity (SSIM) [35], vision quality (human perception) and time-consuming. We have released our code and posted the synthesized DT videos on the project page https://shiming-chen.github.io/Similarity-page/Similarity.html.

4.1. Implementation Details

In the following experiments, we resize the frame size of all DT videos to 150×100 pixels, which is similar to [40] for facilitating direct comparison. Moreover, we train our model with the first 59 to 200 frames (because the length of the shortest observed sequences is 59) of each DT sequence and synthesize a new one with long-term frame length.

4.2. Experiment 1: Parameters Analysis

Kernel Function $K(u, v)$. As kernel function selection is important for kernel learning [23, 32, 40] and it can directly affect the stability of Similarity-DT. For evaluation, we test several general kernel functions (e.g., Linear kernel, Polynomial kernel, Gaussian kernel, Rational Quadratic kernel, Multiquadric kernel, and Sigmoid kernel) for Similarity-DT on Dyntex dataset. As shown in Figure 3, different kernel functions exhibit various performances. It is obvious that Gaussian kernel and Rational Quadratic kernel outperform other kernels with better PSNR and SSIM scores, which shows that Similarity-DT can synthesize more high-quality DT sequences using these two kernel functions. Generally, Gaussian kernel function shares similar characters with Ra-

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

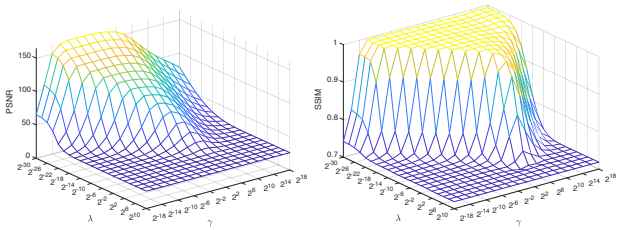


Figure 4. Quantitative comparison of various regularization factors λ and coefficients of Gaussian kernel function γ used in our method on the whole Dyntex dataset. we report PSNR (left) and SSIM (right) in terms of the number of generated frames.

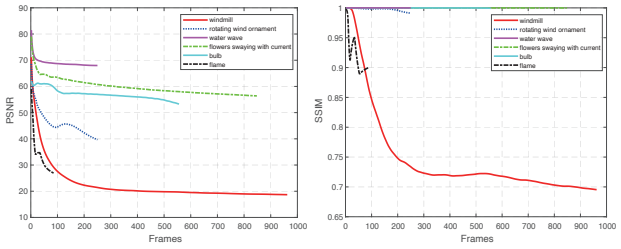


Figure 5. Demonstrate the sustainability of Similarity-DT with quantitative evaluation on 6 DT videos, e.g., bulb, water wave and windmill, etc. we report PSNR (left) and SSIM (right) in terms of the number of generated frames.

tional Quadratic kernel function, and thus Similarity-DT can consistently synthesize realistic DTs with these two kernels. However, the later one is sensitive to its parameter.

Regularization Factor λ. In order to achieve good generalization performance of Similarity-DT, the regularization factor λ, and the coefficient of Gaussian kernel function γ of the models need to be chosen appropriately. We have tried a wide range of λ and γ. See from Figure 4, we can find that the performance of Similarity-DT will be stable in two time periods (period 1:$\lambda < 2^{-14}$, period 2:$\lambda > 2^2$), which shows that Similarity-DT is over-fitting and under-fitting respectively. The regularization of the model is insufficient if regularization factor λ is too small, resulting that DT synthesis model is overly confident to the training sequences (the first 200 frames). Thus model may fail to generate high-quality DT sequences after 200 frames for some DTs that lacks good stochastic repetitiveness in the temporal domain. However, if a too large λ is used, the model is over-regularized, which leads that the stationarity and repetitiveness of DTs are smoothed overly. That is, the weak correlation between different frames is excessively weakened. Figure 4 also shows that, the coefficient of the Gaussian kernel function γ closely interferes with the regularization ability of λ. Therefore, the optimal combination of (λ, γ) of Similarity-DT with the Gaussian kernel is chosen for the following experiments ($\lambda = 2^{-10}, \gamma = 2^8$).

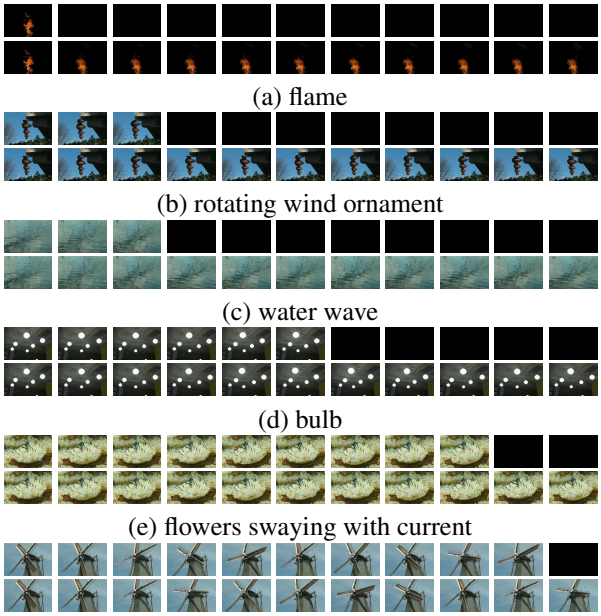


Figure 6. Frames of the synthesized long-term DT sequences generated by Similarity-DT. For each category, the first row displays the 11 frames of the observed sequence (the black frames denote lacking of corresponding frame of the observed sequence), and the second row displays the the corresponding frames of the synthesized sequences. From left to right, the columns are the 2-th, 100-th, 200-th, 300-th, 400-th, 500-th, 600-th, 700-th, 800-th, 900-th, 1000-th frames of the observed and synthesis sequences.

4.3. 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 that mainly refers to no obvious visual decays, divergences, and abrupt jump for the synthesized long-term sequences. Therefore, we intuitively analyse the sustainability of our method using visual quality and quatitative evaluation metrics.

For evaluation, the visual quality comparison over several synthesized DT sequences of different classes is presented in Figure 6. To show the robustness of Similarity-DT, we train the model using the first 200 frames if the length l of observed sequences is longer than 200, and otherwise, the whole frames of observed sequences are used for training. We can see that Similarity-DT not only generates high-fidelity DTs in short-term, but also generates high-quality DTs in long-term even if the observed sequences is short, e.g., flame ($l = 88$), rotating wind ornament ($l = 250$), water wave ($l = 250$), bulb ($l = 556$), etc. Note that Figure 6(a) seemingly shares with similar DT, because these generated frames locate in similar/same cycle of the observed sequence.

Meanwhile, we also show the quantitative evaluation re-

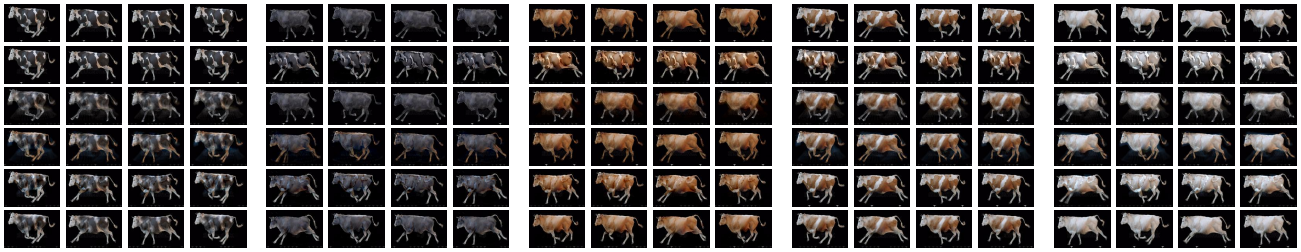


Figure 7. Sythesizing DTs by transfering the trained model of Similarity-DT. For each group, the first row displays the frames of the observed sequences of cow, the other rows display the frames of the synthesized sequences corresponding to the first row with different trained models (from top-bottom: trained on cow1, trained on cow2, trained on cow3, trained on cow4, trained on cow5).

sults to demonstrate the sustainability of Similarity-DT. Here we report SSIM and PSNR in terms of frames (from 1 to 848) of 6 observed sequences because the length of such observed sequences is range from 88 to 848. See from Figure 5, Similarity-DT achieves desired PSNR and SSIM, which show that Similarity-DT synthesized high-fidelity DTs. Although the PSNR and SSIM decreased as the number of generated frames increased for some DT videos (e.g., windmill , frame), they are still large ($PSNR > 18$, $SSIM > 0.69$). Notably, Similarity-DT achieved extensive SSIM index with 1 for whole long-term sequences (e.g., flowers swaying with current, bulb, water wave), which suggests that its generated DTs almost as same as observed sequences. These results prove that our method correctly exhibits the statistical stationarity in the spatial domain and stochastic repetitiveness in the temporal dimension of DT sequences using similarity prior, and thus it can effectively synthesize realistic DTs in the long-term.

4.4. Experiment 3: Generalization Analysis

Good generalization performance is key goal for all learning tasks. Similar to [38], we also specialize our method to learn roughly aligned video sequences of DTs, which are non-stationary in either spatial or temporal domain. In this study, it is different from [38] 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 [38] for evaluation. See from figure 7, the 25 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, and then test the 5 trained models on the 5 observed sequences. Thus, we gain 25 realistic, synthesized sequences.

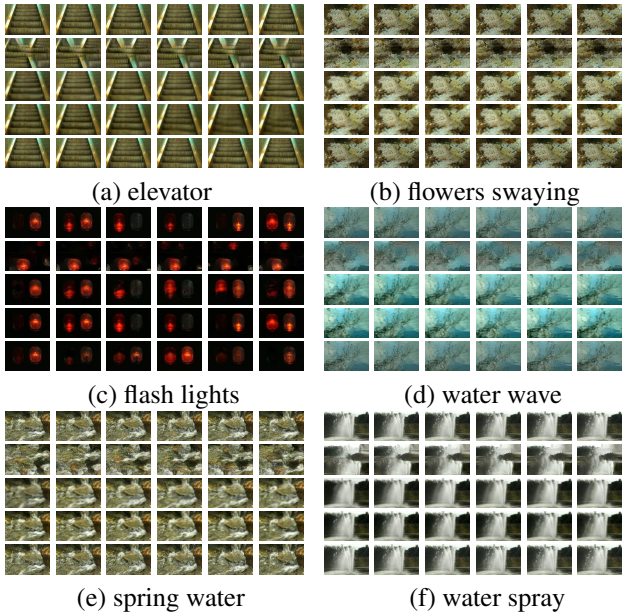


Figure 8. Visual quality comparison between three neural-network-based DT synthesis methods and our method (Similarity-DT) for 6 different DT videos. For each category, the first row displays 6 frames of the observed sequence, and the other rows display the corresponding frames of synthesized sequences generated by different methods (top-to-bottom: TwoStream [30], STGCN [38], DG [36] and Similarity-DT).

The experiment results show that our method can transfer the trained model to generate new sequences for other spatial-temporally aligned DT sequences. In conclusion, our method is effective and efficient for synthesizing realistic appearances and motions of the test cows, which suggests that our method performs excellent generalization performance.

4.5. Experiment 4: Comparison with Baseline Methods

We first compare our method with five non-neural-network-based DT synthesis methods, including FFT-LDS

Table 1. Comparison with different non-neural-network-based DT synthesis methods on PSNR and SSIM. Group 1: evaluating on Dyntex dataset. Group 2: evaluating on Gatech Graphcut Textures dataset. Avg. denotes the average PSNR and SSIM on all the 15 test sequences.

	Ours		FFT-LDS [1]		KPCR [40]		LDS [9]		SLDS [28]		Kernel-DT [4]	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
boiling water	**36.591**	**0.958**	27.570	0.887	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	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	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	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	28.024	0.957	31.350	0.972	31.350	0.972	29.788	0.978
spring water	**64.198**	**1.000**	21.435	0.607	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	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	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	30.740	0.917	29.483	0.889	28.654	0.878	25.565	0.846
water spray in pool	**67.475**	**1.000**	21.076	0.426	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	27.767	0.745	27.394	0.651	27.394	0.651	22.371	0.537
waterfall in mountain	**71.851**	**1.000**	18.509	0.539	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	20.537	0.802	23.840	0.859	23.840	0.859	23.068	0.854
flame	**46.185**	**0.910**	35.046	0.901	15.977	0.730	35.115	0.902	35.115	0.902	33.495	0.887
waterfall	**43.719**	**0.996**	30.406	0.895	31.958	0.937	30.652	0.902	30.652	0.902	26.850	0.868
Avg.	**49.320**	**0.958**	26.293	0.754	24.678	0.751	26.411	0.759	26.322	0.757	24.061	0.729

Table 2. Comparison with baseline methods on time-consuming. The former five are non-neural-network-based DT synthesis methods, and the other three are neural-network-based DT synthesis methods.

	Ours	FFT-LDS [1]	KPCR [40]	LDS [9]	SLDS [28]	Kernel-DT [4]	TwoStream [30]	STGCN [38]	DG [36]
Train. time (Sec.)	0.090	0.928	1.990	0.148	2.475	0.830	-	4188	3904.418
Test time (Sec.)	26.060	5.516	12.214	4.048	3.799	1007.260	8235	7.210	52.292
Generated frames	1200	1200	1200	1200	1200	1200	12	70	120
Using GPU	×	×	×	×	×	×	✓	✓	✓
FPS	46.040	217.560	98.248	296.450	315.900	1.191	0.002	9.709	2.295

[1], KPCR [40], LDS [9], SLDS [28], Kernel-DT [4]. To facilitate direct comparison, we tested all these models using 15 gray DT videos (150×100 pixels) on PSNR and SSIM. See from Table 1, our method attained the best performance over all non-neural-network-based DT methods. Furthermore, the proposed Similarity-DT beated the second best results by a large margins (22.909 for average PSNR and 0.199 for average SSIM on 15 DT videos). Notably, our methods achieved extensive SSIM index with 1 for some DT videos (e.g., bulb, foutain, spring water, etc.). Therefore, one can learn that the Similarity-DT can make full use of the similarity pror for DT synthesis. Indeed, all methods failed to synthesize high-quality DT video for rotating wind ornament, because this video is originally blurry.

Then, our model is compared with three neural-network-based DT synthesis methods, such as TwoStream[4] [30], STGCN[5] [38]and DG[6] [36] on 6 DT videos (e.g., elector, flash lights, and spring water, etc.). For a fair comparison, we display 6 similar index frames of generated sequences. See from Figure 8, the DT sequences generated by TwoStream [30] are divergent, because TwoStream method has a limitation that it cannot synthesize DTs not being spatially homogeneous (e.g., elevator, water spray). As for STGCN [38] and DG [36], 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 time-consuming of different DT synthesis methods, including neural-network-based methods and non-neural-network-based methods. As shown in Table 2, Similarity-DT 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 more high-quality DT videos with fast speed and low computation over 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 dif-

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

ferent frames is a critical prior knowledge for DT synthesis.

5. Conclusion

In this study, we proposed a novel dynamic texture synthesis method (Similarity-DT) that integrates kernel learning and ELM into a powerful unified synthesis model to learn kernel similarity embedding, which elegantly exhibits statistical stationarity in the spatial domain and stochastic repetitiveness in the temporal dimension of DT. Notably, kernel similarity embedding makes use of the exciting property of kernel learning that kernel representation implicitly exhibits the similarity correlation of different frames, which is a critical prior knowledge for synthesizing high-quality DTs. The competitive results on DT videos collected from two benchmark datasets and internet demonstrate the superiority and great potentials of our method for DT synthesis. It also shows clear advantages over all the compared baselines.

References

[1] Bobby Abraham, Octavia I. Camps, and Mario Sznaier. Dynamic texture with fourier descriptors. In *The 4th International Workshop on Texture Analysis and Synthesis*, pages 53–58, 2005. 2, 8

[2] Peter 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*, 44:525–536, 1996. 3

[3] Antoni B Chan and Vasconcelos Nuno. Modeling, clustering, and segmenting video with mixtures of dynamic textures. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 30(5):909–926, 2008. 1

[4] Antoni B. Chan and Nuno Vasconcelos. Classifying video with kernel dynamic textures. *CVPR*, pages 1–6, 2007. 2, 8

[5] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *ICCV*, pages 1511–1520, 2017. 1

[6] Shiming Chen, Yisong Wang, Cheng-Jian Lin, Weiping Ding, and Zehong Cao. Semi-supervised feature learning for improving writer identification. *Information Sciences (INS)*, 482:156–170, 2019. 1, 3

[7] De Cheng, Yihong Gong, Sanping Zhou, Jinjun Wang, and Nanning Zheng. Person re-identification by multi-channel parts-based cnn with improved triplet loss function. In *CVPR*, pages 1335–1344, 2016. 1, 2

[8] Chenwei Deng, Shuigen Wang, Zhen Li, Guang-Bin Huang, and Weisi Lin. Content-insensitive blind image blurriness assessment using weibull statistics and sparse extreme learning machine. *IEEE Transactions on Systems, Man, and Cybernetics: Systems (SMC-A)*, 49:516–527, 2019. 2, 3

[9] Gianfranco Doretto, Alessandro Chiuso, Ying Nian Wu, and Stefano Soatto. Dynamic textures. *International Journal of Computer Vision (IJCV)*, 51(2):91–109, 2003. 1, 2, 5, 8

[10] Christoph Feichtenhofer, Axel Pinz, and Richard P Wildes. Temporal residual networks for dynamic scene recognition. In *CVPR*, pages 4728–4737, 2017. 1

[11] Yang Fu, Yunchao Wei, Guanshuo Wang, Xi Zhou, Honghui Shi, and Thomas S. Huang. Self-similarity grouping: A simple unsupervised cross domain adaptation approach for person re-identification. In *ICCV*, 2019. 1, 2

[12] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In *NeurIPS*, 2015. 1, 3, 5

[13] Bernard Ghanem and Narendra Ahuja. Maximum margin distance learning for dynamic texture recognition. In *ECCV*, pages 223–236, 2010. 5

[14] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017. 1, 2

[15] Arthur E. Hoerl and Robert W. Kennard. Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, 42(1):80–86, 2000. 4

[16] Gao Huang, Shiji Song, Jatinder N. D. Gupta, and Cheng Wu. Semi-supervised and unsupervised extreme learning machines. *IEEE Transactions on Cybernetics (TCYB)*, 44:2405–2417, 2014. 2, 3

[17] Guang-Bin Huang, Hongming Zhou, Xiaojian Ding, and Rui Zhang. Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42:513–529, 2012. 4

[18] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: a new learning scheme of feedforward neural networks. In *IJCNN*, pages 985–990, 2004. 2, 3

[19] Vivek Kwatra, Arno Schödl, Irfan A. Essa, Greg Turk, and Aaron F. Bobick. Graphcut textures: image and video synthesis using graph cuts. *ACM Transactions on Graph (TOG)*, 22:277–286, 2003. 5

[20] Mingbao Lin, Rongrong Ji, Hanying Liu, Xiaoshuai Sun, Yongjian Wu, and Yunsheng Wu. Towards optimal discrete online hashing with balanced similarity. In *AAAI*, 2019. 1, 2

[21] Hong Liu, Rongrong Ji, Yongjian Wu, Feiyue Huang, and Baochang Zhang. Cross-modality binary code learning via fusion similarity hashing. In *CVPR*, pages 6345–6353, 2017. 1, 2

[22] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. Supervised hashing with kernels. In *CVPR*, pages 2074–2081, 2012. 1, 2

[23] Xinwang Liu, Miaomiao Li, Lei Wang, Yong Dou, Jianping Yin, and En Zhu. Multiple kernel k-means with incomplete kernels. *IEEE transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, in press, 2019. 5

[24] Adeel Mumtaz, Weichen Zhang, and Antoni B. Chan. Joint motion segmentation and background estimation in dynamic scenes. In *CVPR*, pages 368–375, 2014. 1

[25] Renaud Péteri, Sándor Fazekas, and Mark J. Huiskes. Dyntex: A comprehensive database of dynamic textures. *Pattern Recognition Letters (PRL)*, 31:1627–1632, 2010. 5

[26] Masaki Saito, Eiichi Matsumoto, and Shunta Saito. Temporal generative adversarial nets with singular value clipping. In *ICCV*, pages 2849–2858, 2017. 3

[27] John Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge: Cambridge University Press, 2004. 2

[28] Sajid M. Siddiqi, Byron Boots, and Geoffrey J. Gordon. A constraint generation approach to learning stable linear dynamical systems. In *NeurIPS*, 2007. 2, 8

[29] Jiexiong Tang, Chenwei Deng, and Guang-Bin Huang. Extreme learning machine for multilayer perceptron. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, 27:809–821, 2016. 2, 3

[30] Matthew Tesfaldet, Marcus A Brubaker, and Konstantinos G Derpanis. Two-stream convolutional networks for dynamic texture synthesis. In *CVPR*, pages 6703–6712, 2018. 1, 3, 5, 7, 8

[31] Ruben Villegas, Jimei Yang, Seunghoon Hong, Xunyu Lin, and Honglak Lee. Decomposing motion and content for natural video sequence prediction. In *ICLR*, 2017. 3

[32] Wei Wang, Hao Wang, Chen Zhang, and Yang Gao. Fredholm multiple kernel learning for semi-supervised domain adaptation. In *AAAI*, pages 2732–2738, 2017. 5

[33] Yizhou Wang and Song-Chun Zhu. A generative method for textured motion: Analysis and synthesis. In *ECC)*, pages 583–598, 2002. 1

[34] Zhou Wang and A. C. Bovik. Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE Signal Processing Magazine*, 26:98–117, 2009. 5

[35] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing (TIP)*, 13(4):600–612, 2004. 5

[36] Jianwen Xie, Ruiqi Gao, Zilong Zheng, Song-Chun Zhu, and Ying Nian Wu. Learning dynamic generator model by alternating back-propagation through time. In *AAAI*, 2019. 3, 7, 8

[37] Jianwen Xie, Song-Chun Zhu, and Ying Nian Wu. Synthesizing dynamic patterns by spatial-temporal generative ConvNet. In *CVPR*, pages 7093–7101, 2017. 1, 3

[38] Jianwen Xie, Song-Chun Zhu, and Ying Nian Wu. Learning energy-based spatial-temporal generative convnets for dynamic patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, in press, 2019. 1, 3, 7, 8

[39] Xianglei Xing, Tian Han, Ruiqi Gao, Song-Chun Zhu, and Ying Nian Wu. Unsupervised disentangling of appearance and geometry by deformable generator network. In *CVPR*, pages 10354–10363, 2019. 3

[40] Xinge You, Weigang Guo, Shujian Yu, Kan Li, José C Príncipe, and Dacheng Tao. Kernel learning for dynamic texture synthesis. *IEEE Transactions on Image Processing (TIP)*, 25(10):4782–4795, 2016. 1, 2, 5, 8

[41] Zhedong Zheng, Xiaodong Yang, Zhiding Yu, Liang Zheng, Yi Yang, and Jan Kautz. Joint discriminative and generative learning for person re-identification. In *CVPR*, 2019. 1, 3

[42] Yipin Zhou and Tamara L. Berg. Learning temporal transformations from time-lapse videos. In *ECCV*, 2016. 3

[43] Fatih Çakir and Stan Sclaroff. Adaptive hashing for fast similarity search. In *ICCV*, pages 1044–1052, 2015. 1, 2

Appendix

In this appendix, Section A provides mathematical details for the derivation of the additional positive diagonal matrix $\lambda\mathbf{I}$ (Eq. (12) in the main paper). Section B displays some frames of the synthesized DTs generated by Similarity-DT using different kernel functions. Section C provides different kernel similarity embeddings that are regularized by different regularization factor λ.

A. Proof of the Additional Positive Diagonal Matrix $\lambda\mathbf{I}$

To improve the stability and generalization performance of Similarity-DT, we can add a regularization factor λ (positive small value) to control the regularization performance of $\|\boldsymbol{\beta}\|$ during optimization. Then, the optimization object of Similarity-DT can be written as Eq. (13).

$$\begin{aligned} &Minimize: L = \frac{1}{2}\lambda\|\boldsymbol{\beta}\|^2 + \frac{1}{2}\sum_{i=1}^{N}\|\boldsymbol{\xi}_i\|^2 \\ &s.t. \quad \mathbf{h}\left(\mathbf{x}_i\right)\boldsymbol{\beta} - \mathbf{Y}_i^{\mathrm{T}} = \boldsymbol{\xi}_i^{\mathrm{T}} \end{aligned} \tag{13}$$

where $i = 1, \ldots, N$ (N is the number of training frames), $\boldsymbol{\xi}_i = [\xi_{i,1}, \ldots, \xi_{i,m}]^{\mathrm{T}}$ is the training error vector of the training sample x_i. Based on the Langrange theorem, training Similarity-DT is equivalent to solve the following optimization object:

$$\begin{aligned} L = \frac{1}{2}\lambda\|\boldsymbol{\beta}\|^2 + \frac{1}{2}\sum_{i=1}^{N}\|\boldsymbol{\xi}_i\|^2 \\ - \sum_{i=1}^{N}\sum_{j=1}^{m}\alpha_{i,j}\left(\mathbf{h}\left(\mathbf{x}_i\right)\boldsymbol{\beta}_j - Y_{i,j} + \xi_{i,j}\right) \end{aligned} \tag{14}$$

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

$$\frac{\partial L}{\partial \boldsymbol{\beta}_j} = 0 \rightarrow \boldsymbol{\beta} = \frac{1}{\lambda}\mathbf{H}^{\mathrm{T}}\boldsymbol{\alpha} \tag{15}$$

$$\frac{\partial L}{\partial \boldsymbol{\xi}_i} = 0 \rightarrow \boldsymbol{\alpha}_i = \boldsymbol{\xi}_i \tag{16}$$

$$\frac{\partial L}{\partial \boldsymbol{\alpha}_i} = 0 \rightarrow \mathbf{h}\left(\mathbf{x}_i\right)\boldsymbol{\beta} - \mathbf{Y}_i^{\mathrm{T}} + \boldsymbol{\xi}_i^{\mathrm{T}} = 0 \tag{17}$$

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

Substituting Eq. (15) and Eq. (16) into Eq. (17), which can be written as:

$$\left(\mathbf{I} + \frac{1}{\lambda}\mathbf{H}\mathbf{H}^{\mathrm{T}}\right)\boldsymbol{\alpha} = \mathbf{Y} \rightarrow \boldsymbol{\alpha} = \left(\left(\mathbf{I} + \frac{1}{\lambda}\mathbf{H}\mathbf{H}^{\mathrm{T}}\right)\right)^{-1}\mathbf{Y} \tag{18}$$

Then, by combining Eq. (15) and Eq. (18), the output weights $\boldsymbol{\beta}$ of the hidden layer can be formulated as:

$$\begin{aligned} \boldsymbol{\beta} &= \frac{1}{\lambda}\mathbf{H}^{\mathrm{T}}\left(\mathbf{I} + \frac{1}{\lambda}\mathbf{H}\mathbf{H}^{\mathrm{T}}\right)^{-1}\mathbf{Y} \\ &= \mathbf{H}^{\mathrm{T}}\left(\lambda\mathbf{I} + \mathbf{H}\mathbf{H}^{\mathrm{T}}\right)^{-1}\mathbf{Y} \end{aligned} \tag{19}$$

Thus, the transition function of Similarity-DT can be formulated as Eq. (20) according to Eq. (1) and Eq. (19).

$$\begin{aligned} f(x) &= \mathbf{h}(\mathbf{x})\boldsymbol{\beta} \\ &= \mathbf{h}(\mathbf{x})\boldsymbol{H}^{T}\left(\lambda\mathbf{I} + \boldsymbol{H}\boldsymbol{H}^{T}\right)^{-1}\boldsymbol{Y} \\ &= \begin{bmatrix} K\left(\mathbf{x}, \mathbf{x}_1\right) \\ \vdots \\ K\left(\mathbf{x}, \mathbf{x}_N\right) \end{bmatrix}^{T}\left(\lambda\mathbf{I} + \boldsymbol{\Omega}_{KSE}\right)^{-1}\boldsymbol{Y} \end{aligned} \tag{20}$$

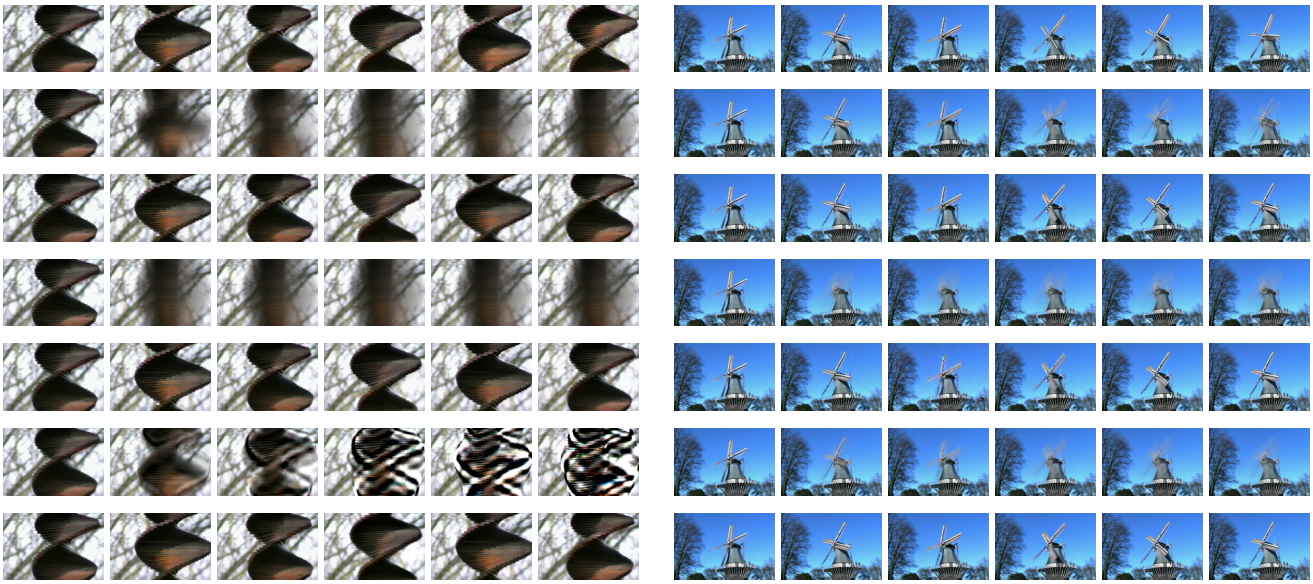
Therefore, a positive diagonal matrix $\lambda\mathbf{I}$ (λ is small) is added to the diagonal axis of $\mathbf{H}\mathbf{H}^{\mathrm{T}}$ for smoothing $\boldsymbol{\Omega}_{KSE}$ during learning the output weights $\boldsymbol{\beta}$.

B. Displaying Some Synthesized DTs Using Different Kernel Functions

In order to intuitively verify the influence of kernel function, we display some frames of two synthesized DT sequences (rotating wind ornament and windmill) generated by Similarity-DT using different kernel functions, as shown in Figure 9.

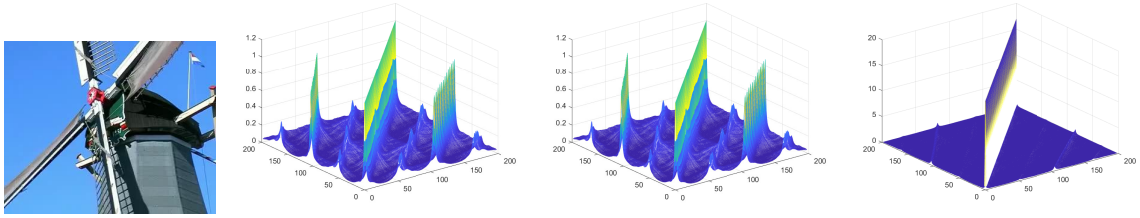
C. Demonstrating the Regularization Ability of Regularization Factor λ

In order to demonstrate the regularization ability of regularization factor λ, we display the corresponding leaned kernel similarity embeddings on two DT videos (e.g., rotating wind ornament and windmill), as shown in Figure 10. It is obvious that if a too large λ is used, the model is over-regularized, which leads that the stationarity and repetitiveness of DTs are smoothed overly. However, if a too small λ is adopted, the model is under-regularized, and thus it will be over-fitting.



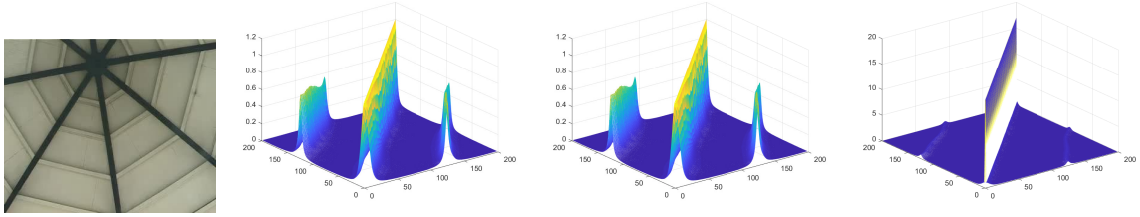
(a) rotating wind ornament (b) windmill

Figure 9. Performance comparisons between different kernel functions used in our proposed Similarity-DT method for the videos “rotating wind ornament" (image (a)) and “windmill" (image (b)). 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 the synthesized sequences generated by Similarity-DT using different kernel functions (top-to-bottom: Linear kernel,Rational Quadratic kernel, Polynomial kernel, Multiquadric kernel, Sigmoid kernel, Gaussian kernel).



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

(a) windmill



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

(b) rotating wind ornament

Figure 10. Demonstrating the regularization ability of regularization factor λ. We display the corresponding leaned kernel similarity embeddings on two samples ((a):windmill; (b):rotating) with different λ values: ($\lambda = 2^{-20}$ shows under-regularization; $\lambda = 2^4$ shows over-regularization).