Contents lists available at ScienceDirect

Signal Processing

journal homepage: www.elsevier.com/locate/sigpro

Dynamic texture modeling and synthesis using multi-kernel Gaussian process dynamic model

Ziqi Zhu [a,b], Xinge You [c,g,*], Shujian Yu [d], Jixin Zou [e], Haiquan Zhao [f]

[a] School of Computer Science and Technology, Wuhan University of Science and Technology, Wuhan, Hubei, China
[b] Hubei Key Lab of Intelligent Information Processing and Real-time Industrial System, Wuhan, Hubei, China
[c] The School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China
[d] The Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, USA
[e] The Institute of Forensic Science, Ministry of Public Security, Beijing, China
[f] School of Electrical Engineering, Southwest Jiaotong University, Chengdu, Sichuan, China
[g] Huazhong University of Science and Technology, Research Institute in ShenZhen, China

ARTICLE INFO

Article history:
Received 26 May 2015
Received in revised form
24 September 2015
Accepted 28 October 2015

Keywords:
Dynamic texture
Synthesis
Gaussian process latent variable model
Multi-kernel learning

ABSTRACT

Dynamic texture (DT) widely exists in various social video media. Therefore, DT modeling and synthesis plays an important role in social media analyzing and processing. In this paper, we propose a Bayesian-based nonlinear dynamic texture modeling method for dynamic texture synthesis. To capture the non-stationary distribution of DT, we utilize the Gaussian process latent variable model for dimensional reduction. Furthermore, we design a multi-kernel dynamic system for the latent dynamic behavior modeling. In our model, we do not make strong assumption on the nonlinear function. Instead, our model automatically constructs a suitable nonlinear kernel for dynamic modeling and therefore is capable of fitting various types of dynamics. We evaluate the effectiveness our methods on the DynTex database and compared with representative DT synthesis method. Experimental results show that our method can achieve synthesis results with higher visual quality.

1. Introduction

In recent years, the amount of social media grows exponentially on Internet. Social media takes many different forms including texts, photos, audios, videos. Among them, video social media holds a high percentage and keeps growing due to its high information-capacity. Dynamic textures (DT) is a sample from a homogeneous stochastic process defined over space and time [1] and widely exists in various video social media. In many video applications, such as video compression [2], film and video rendering [3], and animation [4], it is required to model and synthesize an artificial DT sequence with certain spatial appearance and temporal behavior. Thus, the research of DT synthesis has attracted increasing attentions in recent decades.

Existing DT modeling and synthesis methods can be summarized into three major categories: physics-based methods, sampling based methods and learning based methods.

(1) *Physics based methods* synthesize DT by modeling the physical mechanism of some specific phenomena, such as flames of fire [5], sea wave [6]. The advantage of these methods is the capability of generating impressive synthesis results. However, these methods have high specificity. The model for fire cannot be used for water or smoke. Meanwhile, it is difficult to synthesize natural scenes with

* Correspondence to: The Department of Electronics and Information Engineering, Huazhong University of Science and Technology, Wuhan, Hubei, China
E-mail address: youxg@mail.hust.edu.cn (X. You).

http://dx.doi.org/10.1016/j.sigpro.2015.10.025
0165-1684/© 2015 Elsevier B.V. All rights reserved.

them. Furthermore, their computation cost is extremely expensive.

(2) *Sampling based methods* reassemble the sequences of frames taken from the original video to form a longer DT sequence, ensuring that transitions between consecutive blocks are not noticeable. Meanwhile, some smoothing techniques, such as morphing [7] or diffeomorphic growth [8], are required to diminish discontinuities between blocks. These methods require storage large amount of the original frames (often thousands of video frames) and cannot generate a texture frame unseen.

(3) *Learning based methods* utilize the strong correlations that exist among neighboring pixels in space-time, and learn their parameters from any trainee dynamic texture. Compared with other two types of methods, learning based methods are much more flexible and capable of synthesize a visual approximation of almost any learned DT sequence. Meanwhile, learning based methods only require few memory space for DT synthesize and can achieve high data-compression ratio. Moreover, these methods can also be used for other applications, such as segmentation [9], recognition [10], and editing [11].

Thus, we focus on the learning based method for DT synthesis. In [12–14], the authors use autoregressive-based model for DT synthesis. In these methods, the interaction of pixels within a local causal neighborhood over both space and time are modeled by expressing each pixel in the sequence as a linear combination of its spatial and temporal neighbors. The autoregressive-based model is able to synthesis acceptable DT sequences, but it can not represent non-stationary DT, such as rotations and expansions. To overcome this problem, Doretto et al. proposed the linear dynamic system (LDS) for DT synthesis [15]. In LDS, each video frame is unfolded into a column vector and constitutes a point that follows a trajectory as time evolves. Thus, the key point the LDS is finding an appropriate subspace to describe this trajectory and then identifying the trajectory using methods of dynamical system theory. Later, some LDS-based methods have been proposed to further improve the performance [16–19]. Compared with the autoregressive-based, LDS methods can model unstable DT sequence. However, LDS-based methods achieve good synthesis results only for an oscillatory system [1]. In other situations, they have a tendency toward smoothing the motion and decreasing visual quality over time.

There are two reasons that cause such drawback of LDS-based methods. First, the non-stationary DT contain different data modalities in their appearance distribution which cannot be captured by a linear-dimensionality-reduction scheme. Second, it is difficult for linear dynamic system to model the nonlinear trajectory in the subspace, that governs the evolvement of DT sequence. Motivated by the benefits coming from the multi-view learning [20–25], where performance improvements can be observed when the samples are represented by different feature sets or ”views”, it is necessary to design a nonlinear dynamic system model for DT modeling from the respect of multi-view.

In this paper, we propose a multi-kernel nonlinear dynamic model for DT modeling. Different from existing research, we adopt the Bayesian approach for subspace learning and dynamical system modeling. First, we use the Gaussian process latent variable model (GPLVM) [26]. By adopting nonlinear kernels, such as RBF kernel, GPLVM is capable of capturing the non-stationary distribution of DT sequence through the nonlinear dimensionality reduction process and synthesis the DT frames using mean prediction. Furthermore, to modeling the complex nonlinear trajectory of different DT sequences, we design a multi-kernel nonlinear dynamic system. Compared with other predesigned model [12,15,16], our model is capable of constructing the most suitable nonlinear kernel to fit the given DT data automatically.

The rest of the paper is organized as follows. In Section 2, the related work of learning based method for DT synthesis is briefly introduced. In Section 3, we propose the multi-kernel Gaussian process dynamic model for DT modeling. The learning and synthesis algorithms for multi-kernel Gaussian process dynamic model are given in Section 4. Section 5 presents the experimental results. Conclusions are drawn in Section 6.

2. Related work

In the learning based methods, the DT sequence is regarded as a high-dimensional time series. The modeling of high-dimensional time series usually contains two essential stages, dimensionality reduction and dynamical modeling. Thus, the general model for DT modeling can be expressed as:

$$x_{t+1} = f(x_t, A) + n_{x,t} \tag{1}$$

$$y_t = g(x_t, B) + n_{y,t} \tag{2}$$

where $y_t \in R^D$ denotes the column vector unfolded from the frame at time t of DT sequence. Normally, the dimension D is very large. $x_t \in R^Q$ (with $Q \ll D$) denotes the latent variable from the subspace which govern the dynamic behavior of DT sequence. $g: R^D \rightarrow R^Q$ represents the dimensionality reduction function. $f: R^Q \rightarrow R^Q$ represents the dynamic function. $n_{x,t}$ and $n_{y,t}$ denote the noise. Fig. 1 shows the graph model of DT model represented by (1) and (2).

As for the dimensionality reduction function $g(\cdot)$, most of LDS-based DT modeling methods adopted linear algorithm, such as PCA, for dimensionality reduction [15,17,18]. The linear dimensionality reduction algorithms are straightforward to implement and efficient. However, these algorithms cannot capture complex distribution of most DT sequences. Although there are some nonlinear dimensionality reduction algorithms, they produce either irreversible mapping [27] or several different coordinate spaces [28], which are not suitable for DT modeling and synthesis. Thus, it is desired to find a nonlinear dimensionality reduction algorithm which provides reversible mapping and uniformed subspace.

As for the dynamic function $f(\cdot)$, early LDS methods used linear system for DT modeling [15,29]. However, the dynamic behavior of most DT sequence is not linear and cannot be modeled using linear system. To overcome this

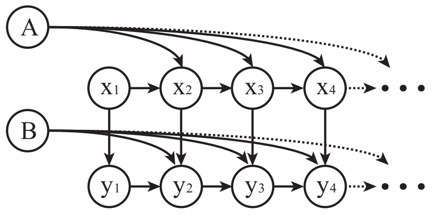

Fig. 1. The graph model of the general dynamic model for DT. *A* and *B* denotes the parameters for function *f* and *g* respectively.

problem, the switching linear system [30] or piecewise linear system [31] are adopted in LDS for nonlinear dynamic modeling. However, these models are designed based on strong assumption of the dynamic behavior and thus are only suitable for specific types of dynamic texture. In this paper, we look forward a more flexible model which is capable of fitting most DT.

3. Multi-kernel Gaussian process dynamic model

To capture the complex distribution of DT sequence, it is desired to design a nonlinear latent mapping function $g(\cdot)$ for dimensionality reduction. Different from existing research, we do not want to make strong assumption on the functional form of $g(\cdot)$. Instead we would like to infer it in a fully Bayesian non-parametric fashion using Gaussian Process [32]. Therefore, we assume that the dynamic texture sequence $\boldsymbol{y}_i$ is a multivariate Gaussian process indexed by $\boldsymbol{x}_i$, and we have:

$$p(\boldsymbol{Y}|\boldsymbol{X},\boldsymbol{\theta}) = \prod_{t=1}^{N} p(y_t|x_t,\boldsymbol{\theta}) = \frac{1}{(2\pi)^{DN/2}|\boldsymbol{K}_Y|^{D/2}} \exp\left(-\frac{1}{2}\mathrm{tr}\left(\boldsymbol{K}_Y^{-1}\boldsymbol{Y}\boldsymbol{Y}^T\right)\right) \tag{3}$$

where $\boldsymbol{Y}=\{y_1,y_2,\dots,y_N\}$ denotes the observed DT sequence and $\boldsymbol{X}=\{x_1,x_2,\dots,x_N\}$ denotes latent variable. $\boldsymbol{K}_Y$ is the kernel matrix which determines the properties of the latent mapping $g(\cdot)$. The elements of kernel matrix are generated by a kernel mapping $(\boldsymbol{K}_Y)_{i,j}=k_Y(x_i,x_j)$. To achieve nonlinear mapping, we use the squared exponential covariance function:

$$k_Y(x_i,x_j) = \theta_1 \exp\left(-\frac{\theta_2}{2}(x_i-x_j)(x_i-x_j)^T\right) + \theta_3\delta_{x_i,x_j} \tag{4}$$

where $\boldsymbol{\theta}=\{\theta_1,\theta_2,\theta_3\}$ denotes hyperparameter for kernel mapping k_Y and δ_{x_i,x_j} denotes the Kronecker delta.

As for the dynamic function $f(\cdot)$, we assume that it can be modeled using first-order Markov model based on Gaussian process. In this model, the kernel function used in Gaussian process determines the properties of the dynamic behavior of the latent variables.

There are many kernel functions successfully used in practice, such as linear kernel function, squared-exponential kernel function, and periodic kernel function. For instance, the linear kernel function gives rise to a linear process, while the squared-exponential kernel function yields a nonlinear, smooth and non-Markovian process [33]. The periodic kernel function is used when the data exhibit strong periodicity [32].

However, as for dynamic texture modeling, the latent dynamic behavior vary greatly among different types of dynamic texture. Thus, it is very difficult to design the most suitable kernel for a dynamic texture empirically. To overcome this problem, we propose a multi-kernel dynamic model for dynamic texture modeling:

$$p(\boldsymbol{X}|\lambda,\boldsymbol{W}) = p(x_1)\prod_{t=2}^{N} p(x_t|x_{t-1},\lambda,\boldsymbol{W}) = p(x_1)\frac{1}{(2\pi)^{Q(N-1)/2}|\boldsymbol{K}_X|^{Q/2}}\exp\left(-\frac{1}{2}\mathrm{tr}\left(\boldsymbol{K}_X^{-1}\boldsymbol{X}_{2:N}\boldsymbol{X}_{2:N}^T\right)\right) \tag{5}$$

where $\boldsymbol{K}_X$ denotes the constructed kernel matrix generated by the multi-kernel function:

$$\boldsymbol{k}_X(x_i,x_j) = \sum_{l=1}^{M} w_l k_l(x_i,x_j) + w_\delta \delta_{x_i,x_j}, \quad i,j=1,2,\dots,N-1 \tag{6}$$

where $k_l, l=1,2,\dots,M$ denotes different types of kernel functions. $\boldsymbol{W}=\{w_l|l=1,2,\dots,M\}$ is the weights for all kernel functions. For all w_l, we assume that $w_l>0$ and $\sum_l w_l=1$. $\lambda=\lambda_l|l=1,\dots,M$ is the hyperparameters for the kernels. Since the parameters for each kernel maybe different, the parameters for each kernel is defined as $\lambda_l=\{(\lambda_l)_1,\dots,(\lambda_l)_{M_l}\}$ and M_l is the number of parameters for kernel k_l.

Finally, to discourage overfitting, we can assume that there are some priors on the hyperparameters as [33]:

$$p(\boldsymbol{\theta}) \propto \prod_i \theta_i^{-1} \quad \text{and} \quad p(\lambda) \propto \prod_{i,j} (\lambda_i)_j^{-1} \tag{7}$$

Based on the latent mapping (3), the latent dynamic model (5) and the prior (7), we can obtain the generative model for dynamic texture:

$$p(\boldsymbol{X},\boldsymbol{Y},\boldsymbol{\theta},\lambda|\boldsymbol{W}) = p(\boldsymbol{Y}|\boldsymbol{X},\boldsymbol{\theta})p(\boldsymbol{X}|\boldsymbol{\theta},\boldsymbol{W})p(\boldsymbol{\theta})p(\lambda) \tag{8}$$

It should be noted that the latent dynamic of MK-GPDM can be easily extended into higher-order Markov model. While using first order Markov model, we assume that the dynamic texture frame is only dependent on the one past frame and ignore the correlation between non-neighboring frame. With higher order Markov model, it is possible for us to predict the future frame with more information.

We take second-order Markov chain as an example. The kernel function of the latent dynamic model is then defined as:

$$\boldsymbol{K}_X([x_i,x_{i-1}],[x_j,x_{j-1}]) = \sum_{l=1}^{M} w_l K_l([x_i,x_{i-1}],[x_j,x_{j-1}]) + w_\delta \delta_{([x_i,x_{i-1}],[x_j,x_{j-1}])} \tag{9}$$

From (9), we can find that the correlation among four frames is considered. However, the number of parameters we need to estimate grows exponentially with the order and the computation cost increases significantly.

4. MK-GPDM learning and prediction

To synthesis new DT sequence based on given training samples, we need to estimate all the parameters $\{\boldsymbol{X},\boldsymbol{\theta},\lambda,\boldsymbol{W}\}$ in model (8) and then predict the value of observing data y_t with given time index t. Thus, the learning algorithm and prediction algorithm will be introduced in this section.

4.1. MK-GPDM learning with MAP estimation

Our objective is to model the dynamic texture video sequence. Thus we need to optimize the latent variable $\boldsymbol{X}$, the hyperparameters $\{\boldsymbol{\theta}, \boldsymbol{\lambda}\}$ and the weights for different kernels $\boldsymbol{W}$ using the given video data $\boldsymbol{Y}$. Here, we design a two-step optimization algorithm to learn MK-GPDM.

Step 1: We estimate the latent variable $\boldsymbol{X}$ and hyperparameters $\{\boldsymbol{\theta}, \boldsymbol{\lambda}\}$ using the maximum a posteriori (MAP) algorithm while the kernel weights $\boldsymbol{W}$ fixed.

According to the Bayesian inference, the posteriori distribution is

$$p(\boldsymbol{X}, \boldsymbol{\theta}, \lambda | \boldsymbol{Y}) \propto p(\boldsymbol{Y} | \boldsymbol{X}, \boldsymbol{\theta}, \lambda) p(\boldsymbol{X}, \boldsymbol{\theta}, \lambda) = p(\boldsymbol{Y} | \boldsymbol{X}, \boldsymbol{\theta}) p(\boldsymbol{X} | \boldsymbol{\theta}) p(\boldsymbol{\theta}) p(\lambda) \tag{10}$$

Since $\boldsymbol{W}$ is fixed in the first step, we consider it as a constant rather than a parameter for the posteriori distribution.

To maximize the posteriori distribution, it is equivalent to minimize the negative logarithm of (10). Thus we have:

$$\mathcal{L} = -\ln p(\boldsymbol{X}, \boldsymbol{\theta}, \lambda | \boldsymbol{Y}) = \frac{D}{2} \ln |\boldsymbol{K}_Y| + \frac{1}{2} \mathrm{tr}\left(\boldsymbol{K}_Y^{-1} \boldsymbol{Y}\boldsymbol{Y}^{-1}\right) + \frac{Q}{2} \ln |\boldsymbol{K}_X| + \frac{1}{2} \mathrm{tr}\left(\boldsymbol{K}_X^{-1} \boldsymbol{X}_{2:N} \boldsymbol{X}_{2:N}^T\right) + \sum_i \theta_i + \sum_{i,j} (\lambda_i)_j + C \tag{11}$$

where C denotes the constant part in $\mathcal{L}$. We minimize $\mathcal{L}$ with respect to $\boldsymbol{X}$, $\boldsymbol{\theta}$, $\boldsymbol{\lambda}$ numerically using the scaled conjugate gradients optimizer.

Step 2: We estimate the kernel weights $\boldsymbol{W}$ while the latent variable $\boldsymbol{X}$ and hyperparameters $\{\boldsymbol{\theta}, \boldsymbol{\lambda}\}$ are fixed. According to (5) and (6), the kernel weights $\boldsymbol{W}$ only related to K_X. Therefore, the optimization problem can be formulated as:

$$\text{minimize } \frac{Q}{2} \ln \left| \boldsymbol{K}_X^{-1} \right| + \frac{1}{2} \mathrm{tr}\left(\boldsymbol{K}_X^{-1} \boldsymbol{X}_{2:N} \boldsymbol{X}_{2:N}^T\right) + \alpha \| \boldsymbol{W} \|_2 \tag{12}$$

with respect to $w_i \ge 0$

$$\text{subject to} \sum_i w_i = 1$$

where K_X^{-1} is defined as (6). α is a predefined positive trade-off parameter between model simplicity and the negative logarithm posteriori distribution. Since it is difficult to obtain the close-form solution, instead we use gradient descent method to optimize (12) with the respect of $\boldsymbol{W}$.

Based on the Step 1 and Step 2 described above, the optimization of model (8) can be summarized by Algorithm 1.

Algorithm 1. Leaning MK-GPDM.

```
1 Input: DT data $\boldsymbol{Y}$
2 Output: The latent variable $\boldsymbol{X}$, the hyperparameters $\{\boldsymbol{\theta}, \boldsymbol{\lambda}\}$ and the
    kernel weights $\boldsymbol{W}$.
3 Initialize X with PCA on Y with Q dimensions. Initialize the $\{\boldsymbol{\theta}, \boldsymbol{\lambda}\}$
    and $\boldsymbol{W}$.
4 For i=1 to I do
5   Fix $\boldsymbol{W}$, optimize (11) with respect to $\{\boldsymbol{X}, \boldsymbol{\theta}, \boldsymbol{\lambda}\}$ using SCG for $J_1$
    iterations.
6   Fix $\{\boldsymbol{X}, \boldsymbol{\theta}, \boldsymbol{\lambda}\}$, optimize (12) with respect to $\boldsymbol{W}$ using gradient
    descent method for $J_2$ iterations.
7   For any $w_i < 0$, $w_i \Leftarrow 0$.
8   Normalize $\boldsymbol{W}$ such that $\sum_i w_i = 1$.
9 End For
```

4.2. MK-GPDM prediction

For many applications of DT synthesis, it is desirable to generate new DT frames efficiently. Therefore we adopt a simple online method, named as mean-prediction, for synthesis new DT frames. In mean-prediction, we generate the latent variable x_t conditioned on x_{t-1} based on the

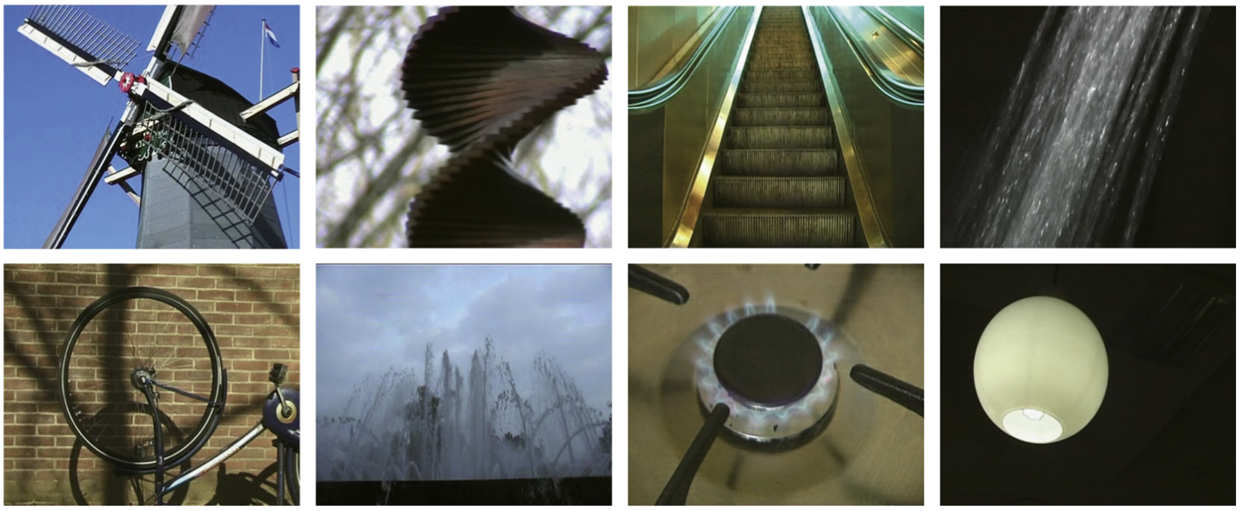
Fig. 2. Sample frames from the DynTex database (left-to-right, top-to-bottom): revolving windmill, rotating whirligig, ascending escalator, warning lamps, revolving bicycle wheel, fountain, controlled fire, swinging pendant lamp.

first-order Markov model using the Gaussian prediction:

$$x_t \sim \mathcal{N}\left(\mu_X(x_{t-1}), \sigma_X^2(x_{t-1})I\right) \tag{13}$$

where

$$\mu_X(x) = \boldsymbol{X}_{2:N}^T \boldsymbol{K}_X^{-1} \mathrm{k}_X(x) \tag{14}$$

$$\sigma_X^2(x) = \boldsymbol{k}_X(x,x) - \mathrm{k}_X(x)^T \boldsymbol{K}_X^{-1} \mathrm{k}_X(x) \tag{15}$$

where $\boldsymbol{k}_X(\cdot,\cdot)$ is the multi-kernel function defined by (6). $\mathrm{k}_X(x)$ is a vector containing $\boldsymbol{k}_X(x, x_i)$ in the i-th entry. In the prediction process, the latent variable x_t is set to be the mean point given by the previous time index as $x_t = \mu_X(x_{t-1})$. Similarly, the new DT sequence is generated by $y_t = \mu_Y(x_t)$.

5. Experiments

In this section, we evaluate the proposed synthesis method on the DynTex database. The some representative learning based DT synthesis methods are evaluated for comparison.

DynTex database [34] is a diverse collection of high quality dynamic texture videos. The latest version contains 650 sequences of dynamic textures, mostly in everyday surroundings. It serves as a standard database for different dynamic texture research areas, such as video texture recognition, video texture segmentation and video texture synthesis. This database contains the so-called golden set of high-quality. Fig. 2 shows some sample frames from the data set.

For evaluation, we test the proposed synthesis method on 4 types of dynamic textures which belong to natural scene. Due to the limitation of computer memory, all testing videos are resized to a resolution of 160×120. In the

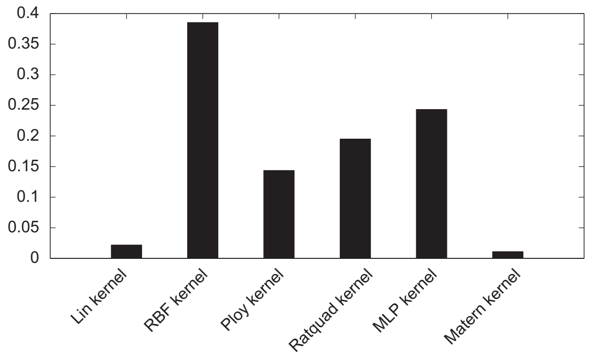

Fig. 4. The kernel weights of MK-GPDM using "64ad410" as training sample.

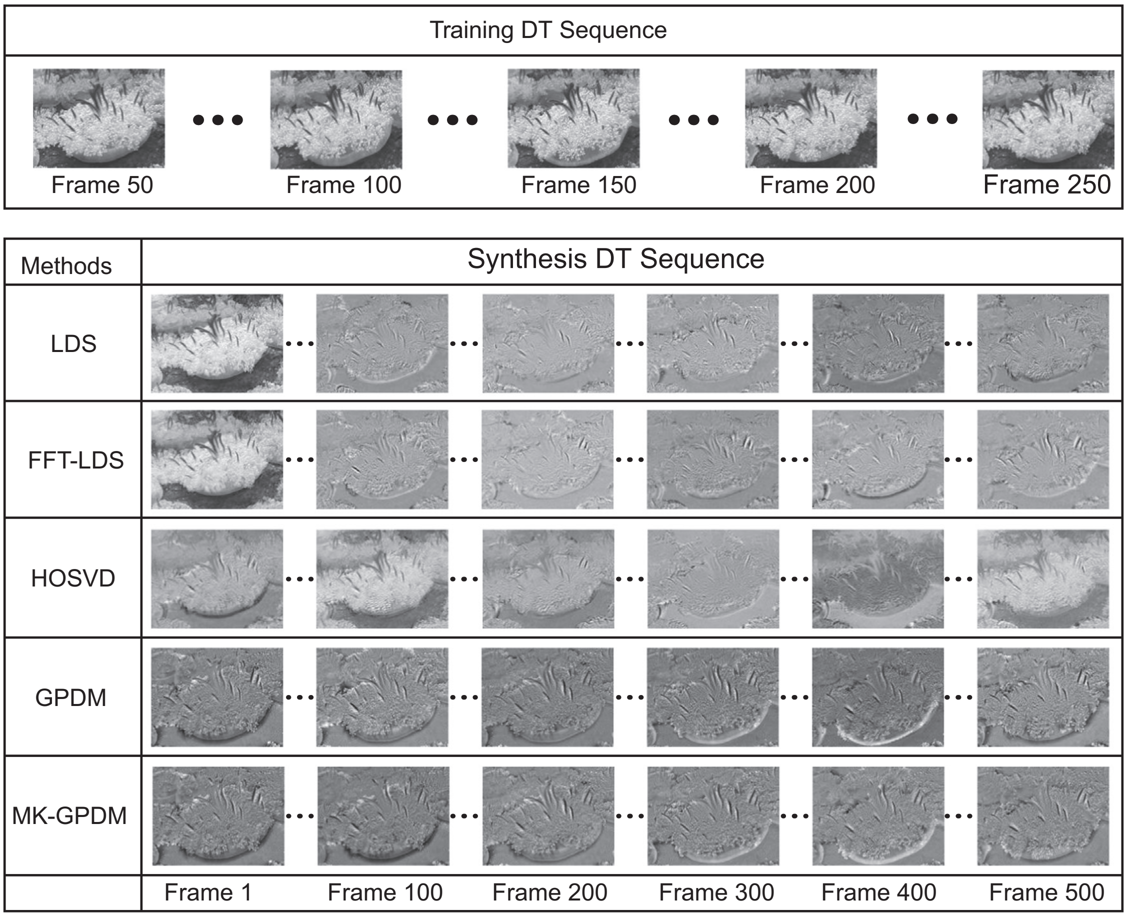

Fig. 3. Synthesis results of sample "64ad410" from dyntex database.

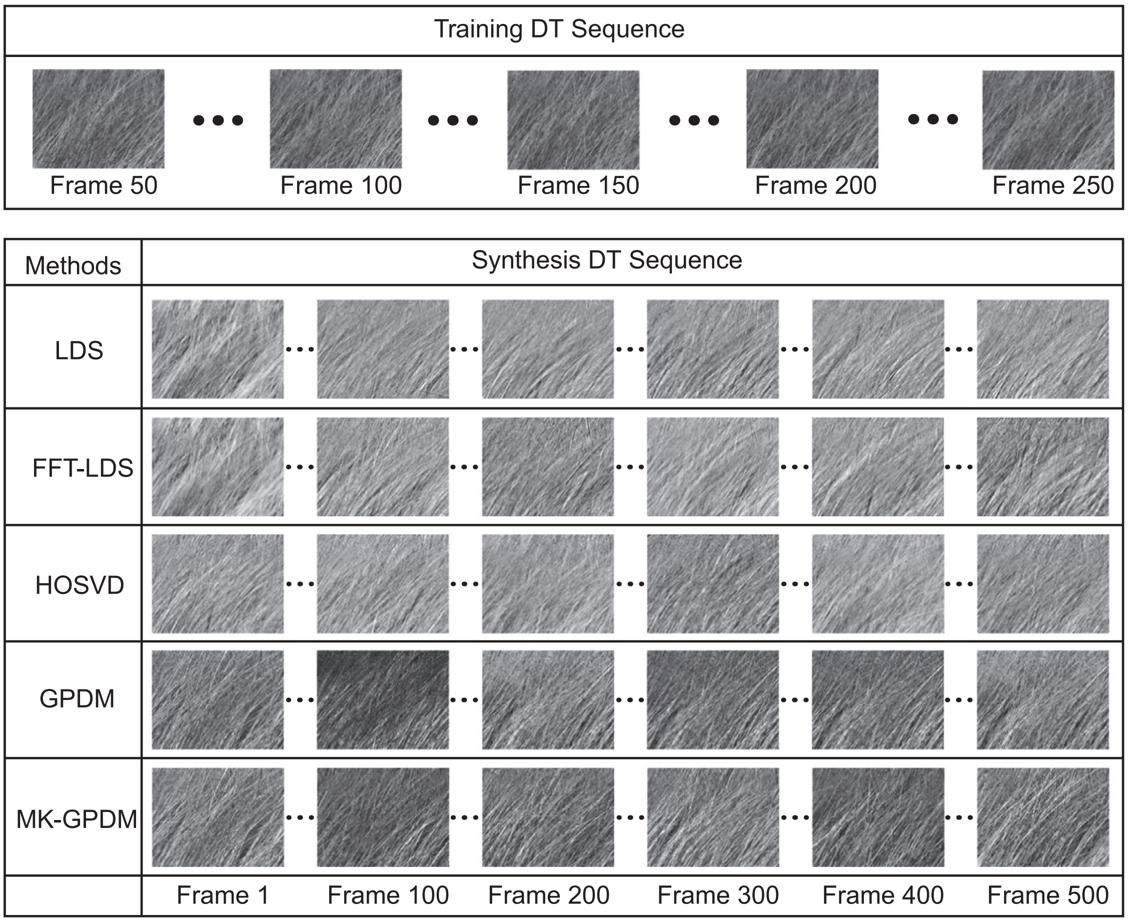

Fig. 5. Synthesis results of sample “54ac210” from dyntex database.

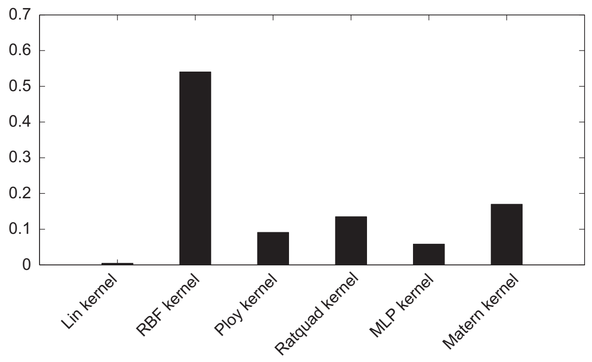

Fig. 6. The kernel weights of MK-GPDM using “54ac210” as training sample.

training stage, we use 250 frames (namely 10 s approximately) as the training DT sample for each DT sequence. In the synthesizing stage, we use the 250th frame of the training sample as the initial frame and generate a DT sequence with 500 frames. All the training samples are converted gray video for convenience of training and synthesizing. We compare our method with the LDS [15], the FFT-LDS [19], High Order-SVD (HOSVD for short) [16] and GPDM [33]. The experimental results are shown Figs. 3–9. To demonstrate the dynamic behavior of synthesized DT sequence, we also generate video sequences, which contains the original DT sequence, the synthesized results of LDS, FFT-LDS, HOSVD, GPDM and our methods, for comparison. All the source code and the synthesized video sequences are available from https://github.com/zhuziqi/Dynamic_Texture_Modeling_using_MKGPDM.

As for the LDS model, we utilize the code provided by Doretto from [15]. As for HOSVD, we utilize the code provided by Costantini from [16]. As for GPDM, we utilize the code provided by Jack from [33]. The dimensionality of the latent subspace is set to be 3 as it is recommend in [33]. As for MK-GPDM, we utilize 6 different types of kernels, including the linear (Lin for short) kernel, the RBF kernel, the polynomial (Ploy for short) kernel, the rational quadratic (Ratquad for short) kernel, the Multilayer Perceptron (MLP for short) kernel and the Matern kernel. The dimensionality of the latent subspace is set to be 20.

Fig. 3 shows the synthesis results of a DT sequence containing actinia. Since the training data is normalized before the training process, the gray-scale of all synthesis results is different from the original DT sequence. In our experiments, we mainly focus on the details of DT frames and smoothness of the synthesized DT sequence. As shown

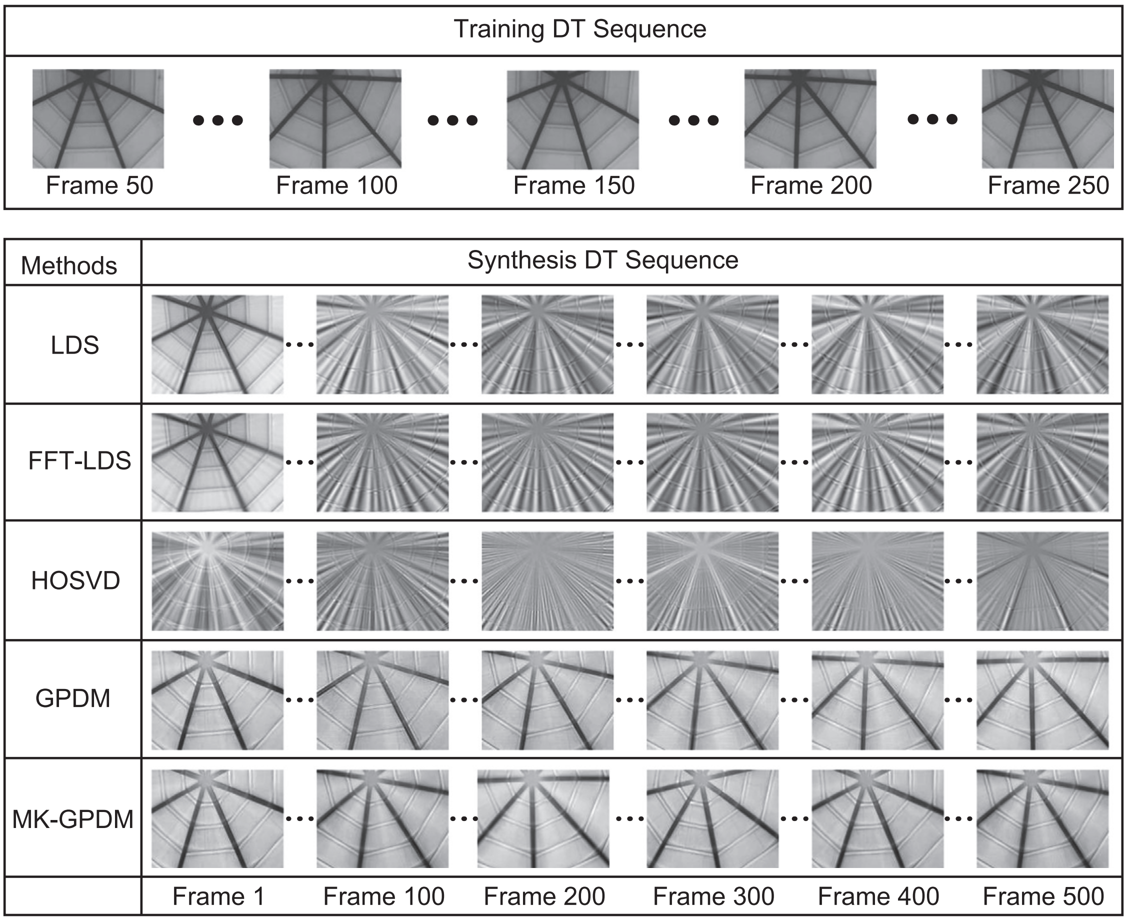

Fig. 7. Synthesis results of sample "649i110" from dyntex database.

in the figure, LDS model is capable of capture the general shape of the actinia, but the details of actinia's tentacle is not clear. The GPDM is capable of capture more image details. However, the global gray-scale is not stable during the synthesis period. Compared with LDS and GPDM, MK-GPDM is capable of capturing most details from training samples and the dynamic behavior is also very smooth. Fig. 4 shows the kernel weights for all 6 different kernels used in MK-GPDM. Among them, the RBF kernel plays the most important role in the multiple dynamic kernel model, but the contribution of the Linear kernel and Matern kernel is quite limited.

Fig. 5 shows the synthesis result using DT sequence "64ad410" as training sample. From the figure and video, we can find that both LDS and GPDM failed to capture the dynamic behavior of the waving grass. However, the synthesis DT sequence of our methods contains crystal details of the grass and waving process is also smooth and natural. The weights of different kernels are shown in Fig. 6. Different from previous sample, Matern kernel plays a secondly important role in MK-GPDM. Similar results can also be found in the synthesis results shown in Figs. 5 and 6.

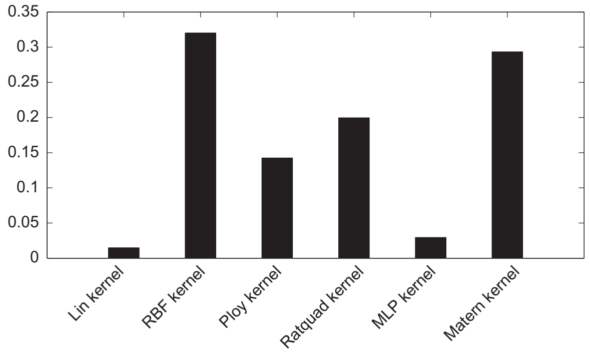

Fig. 8. The kernel weights of MK-GPDM using "649i110" as training sample.

The sample "54ab110" is a DT sequence contains the sea wave. The dynamic behavior of water is much more complex then previous 3 types of DT. As it is shown in Fig. 9, the LDS model can not capture complex dynamic behavior at all. As for GPDM, the quality of synthesized video degenerated quickly. However, by using the proposed MK-GPDM, we obtain a promise synthesis results with smooth transfor-

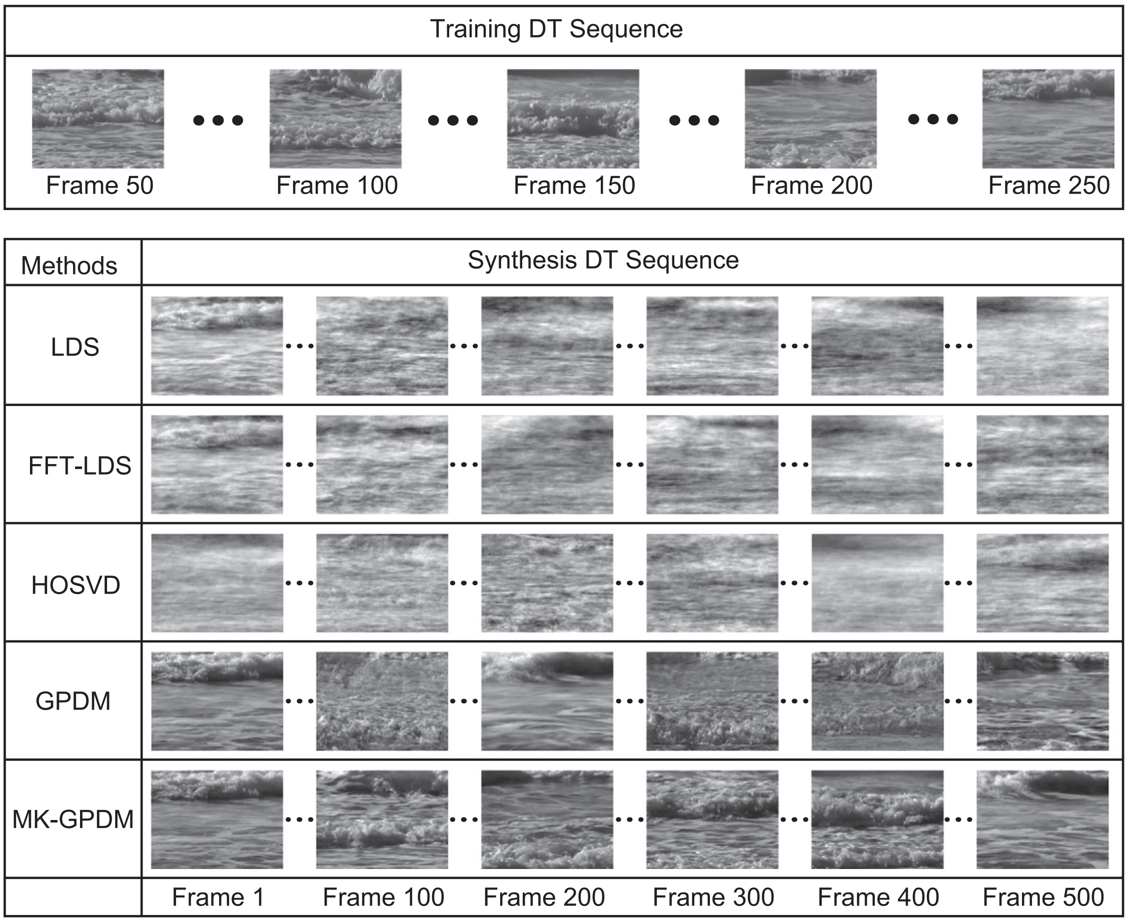

Fig. 9. Synthesis results of sample "54ab110" from dyntex database.

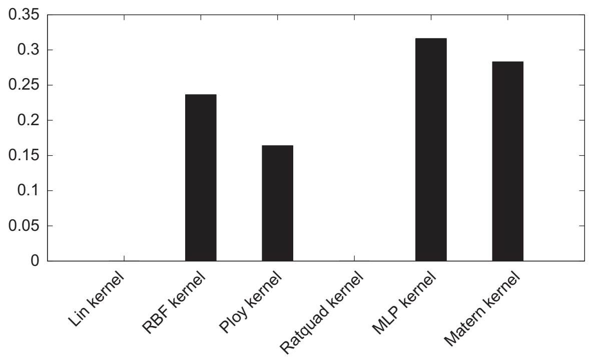

Fig. 10. The kernel weights of MK-GPDM using "54ab110" as training sample.

mation between frames and high quality of DT frames. According to Fig. 10, we can find that the Matern kernel contribute almost as equal as the RBF kernel MK-GPDM.

6. Conclusions

In this paper, we proposed a multi-kernel Gaussian process dynamic model (MK-GPDM) for dynamic texture modeling and synthesis. First, to capture the non-stationary distribution of the DT sequence, we utilize the method of Gaussian process latent variable model for nonlinear dimensional reduction in MK-GPDM. Second, we design a multi-kernel dynamic system for the dynamic behavior modeling of latent variable in subspace. Different from existing methods, our approach is capable of construct the most suitable nonlinear kernel for nonlinear dynamic modeling automatically, and therefore is much more flexible for DT modeling. We evaluate the effectiveness our method on the DynTex database and compared with some representative DT synthesis methods. Experimental results show that our method can achieve higher quality synthesis results.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 61272203, and Grant 61571205, in part by the International Scientific and Technological Cooperation Program of China under Grant 2011DFA12180, in part by the Hubei Province Science and Technology Support Program under Grant 2013BAA120, and in part by the Shenzhen Research Council under Grant JCYJ20140819154343378.

Appendix A. The gradient of $\mathcal{L}$ with the respect to latent variables and hyperparamters:

To estimate the hyperparameters and the latent variable, we need to calculate the gradient of $\mathcal{L}$ with the respect to $\boldsymbol{X}$ and $\{\boldsymbol{\theta}, \lambda\}$ and then use the SCG algorithm. In practice, we use the SCG from the NETLAB matlab toolbox which can be downloaded from http://www.aston.ac.uk/eas/research/groups/ncrg/resources/netlab/downloads/.

The gradient of $\mathcal{L}$ with the respect to $\boldsymbol{X}$ can be found through combining

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{X}_{1:N-1}} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{K}_Y} \cdot \frac{\partial \boldsymbol{K}_Y}{\partial \boldsymbol{X}_{1:N-1}} + \frac{\partial \mathcal{L}}{\partial \boldsymbol{K}_X} \cdot \frac{\partial \boldsymbol{K}_X}{\partial \boldsymbol{X}_{1:N-1}} + \frac{\partial \mathcal{L}}{\partial \boldsymbol{X}_{2:N}} \cdot \frac{\partial \boldsymbol{X}_{2:N}}{\partial \boldsymbol{X}_{1:N-1}} \tag{16}$$

in which

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{K}_Y} = \frac{D}{2} \boldsymbol{K}_Y^{-T} - \frac{1}{2} \left(\boldsymbol{K}_Y^{-1} \boldsymbol{Y} \boldsymbol{Y}^{-1} \boldsymbol{K}_Y^{-1} \right)^T \tag{17}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{K}_X} = \frac{Q}{2} \boldsymbol{K}_X^{-T} - \frac{1}{2} \left(\boldsymbol{K}_X^{-1} \boldsymbol{X}_{2:N} \boldsymbol{X}_{2:N}^{-1} \boldsymbol{K}_X^{-1} \right)^T \tag{18}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{X}_{2:N}} = \frac{1}{2} \left(\boldsymbol{K}_X^{-1} \boldsymbol{X}_{2:N} + \boldsymbol{K}_X^{-T} \boldsymbol{X}_{2:N} \right) \tag{19}$$

with $\frac{\partial \boldsymbol{K}_Y}{\partial x_i}$, $\frac{\partial \boldsymbol{K}_X}{\partial x_i}$ and $\frac{\partial \boldsymbol{X}_{2:N}}{\partial x_i}$ $(i = 1, 2, \dots, N-1)$ via the chain rule.

The gradient of $\mathcal{L}$ with the respect to the hyperparameters $\{\boldsymbol{\theta}, \lambda\}$ can be calculate according to:

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \frac{D}{2} \cdot \mathrm{tr}\left(\boldsymbol{K}_Y^{-1} \frac{\partial \boldsymbol{K}_Y}{\partial \theta_i} \right) - \frac{1}{2} \cdot \left(\boldsymbol{K}_Y^{-1} \boldsymbol{Y} \boldsymbol{Y}^T \boldsymbol{K}_Y^{-1} \right)^T \cdot \frac{\partial \boldsymbol{K}_Y}{\partial \theta_i} + \frac{1}{\theta_i}$$
$$\frac{\partial \mathcal{L}}{\partial \lambda_i} = \frac{Q}{2} \mathrm{tr}\left(\boldsymbol{K}_X^{-1} \frac{\partial \boldsymbol{K}_X}{\partial \lambda_i} \right) - \frac{1}{2} \cdot \left(\boldsymbol{K}_X^{-1} \boldsymbol{X}_{2:N} \boldsymbol{X}_{2:N}^T \boldsymbol{K}_X^{-1} \right)^T \cdot \frac{\partial \boldsymbol{K}_X}{\partial \lambda_i} + \frac{1}{\lambda_i}$$

References

[1] G. Doretto, Dynamic textures: modeling, learning, synthesis, animation, segmentation, and recognition, University of California at Los Angeles, 2005.

[2] R. Costantini, L. Sbaiz, S. Susstrunk, Dynamic texture synthesis: compact models based on luminance-chrominance color representation, in: 2006 IEEE International Conference on Image Processing, IEEE, Atlanta, Georgia, USA, 2006, pp. 2085–2088.

[3] D.G. Aliaga, Visualization of complex models using dynamic texture-based simplification, in: Proceedings of the 7th Conference on Visualization'96, IEEE Computer Society Press, Los Alamitos, California, USA, 1996, pp. 101–106.

[4] P. Lincoln, G. Welch, A. Nashel, A. Ilie, A. State, H. Fuchs, Animatronic shader lamps avatars, in: 8th IEEE International Symposium on Mixed and Augmented Reality, 2009 (ISMAR 2009), IEEE, Orlando, Florida, USA, 2009, pp. 27–33.

[5] J. Stam, E. Fiume, Depicting fire and other gaseous phenomena using diffusion processes, in: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques, ACM, Los Angeles, California, USA, 1995, pp. 129–136.

[6] K. Perlin, An image synthesizer, ACM Siggraph Comput. Graph. 19 (3) (1985) 287–296.

[7] A. Schödl, R. Szeliski, D.H. Salesin, I. Essa, Video textures, in: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, ACM Press/Addison-Wesley Publishing Co., New Orleans, Los Angeles, USA, 2000, pp. 489–498.

[8] Y. Guo, G. Zhao, Z. Zhou, M. Pietikainen, Video texture synthesis with multi-frame LBP-TOP and diffeomorphic growth model, IEEE Trans. Image Process. 22 (10) (2013) 3879–3891.

[9] G. Doretto, D. Cremers, P. Favaro, S. Soatto, Dynamic texture segmentation, in: Proceedings of Ninth IEEE International Conference on Computer Vision, IEEE, Nice, France, 2003, pp. 1236–1242.

[10] P. Saisan, G. Doretto, Y.N. Wu, S. Soatto, Dynamic texture recognition, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001 (CVPR 2001), vol. 2, IEEE, Kauai, Hawaii, USA, 2001, pp. II-58–II-63.

[11] G. Doretto, S. Soatto, Editable dynamic textures, in: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, IEEE, Madison, Wisconsin, USA, 2003, pp. II–137.

[12] M. Szummer, R.W. Picard, Temporal texture modeling, in: Proceedings of International Conference on Image Processing, vol. 3, IEEE, Lausanne, Switzerland, 1996, pp. 823–826.

[13] G. Doretto, E. Jones, S. Soatto, Spatially homogeneous dynamic textures, in: Computer Vision-ECCV 2004, Springer, Prague, Czech Republic, 2004, pp. 591–602.

[14] J. Filip, M. Haindl, D. Chetverikov, Fast synthesis of dynamic colour textures, in: 18th International Conference on Pattern Recognition, 2006 (ICPR 2006), vol. 4, IEEE, Hong Kong, China, 2006, pp. 25–28.

[15] G. Doretto, A. Chiuso, Y.N. Wu, S. Soatto, Dynamic textures, Int. J. Comput. Vis. 51 (2) (2003) 91–109.

[16] R. Costantini, L. Sbaiz, S. Susstrunk, Higher order SVD analysis for dynamic texture synthesis, IEEE Trans. Image Process. 17 (1) (2008) 42–52.

[17] J. Huang, X. Huang, D. Metaxas, Optimization and learning for registration of moving dynamic textures, in: IEEE 11th International Conference on Computer Vision, 2007 (ICCV 2007), IEEE, Rio de Janeiro, Brazil, 2007, pp. 1–8.

[18] Z. Ghahramani, G.E. Hinton, Variational learning for switching state-space models, Neural Comput. 12 (4) (2000) 831–864.

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

[20] C. Xu, D. Tao, C. Xu, Large-margin multi-viewinformation bottleneck, IEEE Trans. Pattern Anal. Mach. Intell. 36 (8) (2014) 1559–1572.

[21] C. Xu, D. Tao, C. Xu, Multi-view intact space learning, IEEE Trans. Pattern Anal. Mach. Intell. 37 (12) (2015) 2531-2544.

[22] Y. Luo, D. Tao, B. Geng, C. Xu, S.J. Maybank, Manifold regularized multitask learning for semi-supervised multilabel image classification, IEEE Trans. Image Process. 22 (2) (2013) 523–536.

[23] Y. Luo, D. Tao, C. Xu, C. Xu, H. Liu, Y. Wen, Multiview vector-valued manifold regularization for multilabel image classification, IEEE Trans. Neural Netw. Learn. Syst. 24 (5) (2013) 709–722.

[24] Y. Luo, T. Liu, D. Tao, C. Xu, Multi-view matrix completion for multi-label image classification, IEEE Trans. Knowl. Data Eng. 27 (11) (2015) 3111-3124.

[25] Y. Luo, D. Tao, R. Kotagiri, C. Xu, Y. Wen, Tensor canonical correlation analysis for multi-view dimension reduction, IEEE Trans. Neural Netw. Learn. Syst. 24 (8) (2015) 2355–2368.

[26] N.D. Lawrence, Gaussian process latent variable models for visualisation of high dimensional data, Adv. Neural Inf. Process. Syst. 16 (3) (2004) 329–336.

[27] J.B. Tenenbaum, V. De Silva, J.C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (5500) (2000) 2319–2323.

[28] Z. Ghahramani, G.E. Hinton, et al., The EM algorithm for mixtures of factor analyzers, Technical Report CRG-TR-96-1, University of Toronto, 1996.

[29] R. Vidal, A. Ravichandran, Optical flow estimation & segmentation of multiple moving dynamic textures, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005 (CVPR 2005), vol. 2, IEEE, San Diego, California, USA, 2005, pp. 516–521.

[30] Y. Li, T. Wang, H.-Y. Shum, Motion texture: a two-level statistical model for character motion synthesis, ACM Trans. Graph. (ToG) 21 (3) (2002) 465–472.

[31] R. Li, T.-P. Tian, S. Sclaroff, Simultaneous learning of nonlinear manifold and dynamical models for high-dimensional time series, In: IEEE 11th International Conference on Computer Vision, 2007 (ICCV 2007), IEEE, Rio de Janeiro, Brazil, 2007, pp. 1–8.

[32] C.E. Rasmussen, Gaussian processes in machine learning, in: Advanced Lectures on Machine Learning, Springer, 2004, pp. 63–71.

[33] J.M. Wang, D.J. Fleet, A. Hertzmann, Gaussian process dynamical models for human motion, IEEE Trans. Pattern Anal. Mach. Intell. 30 (2) (2008) 283–298.

[34] R. Péteri, S. Fazekas, M.J. Huiskes, Dyntex: a comprehensive database of dynamic textures, Pattern Recognit. Lett. 31 (12) (2010) 1627–1632.