TEMPORALLY MULTIPLE DYNAMIC TEXTURES SYNTHESIS USING PIECEWISE LINEAR DYNAMIC SYSTEMS

Xing Yan, Hong Chang, Xilin Chen

Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS),
Institute of Computing Technology, CAS, Beijing, 100190, China
{xing.yan, hong.chang, xilin.chen}@vipl.ict.ac.cn

ABSTRACT

Real-world nonlinear dynamic textures (DTs) usually consist of temporally multiple linear DTs which cannot be correctly modeled by previous works. In this paper, we propose piecewise linear dynamic systems (PLDS) to model temporally multiple DTs. PLDS simultaneously decides the temporal segmentation, models each DT segment with an LDS and the whole DT by switching between the LDS'. Experimental results verify that PLDS can capture the stochastic and dynamic nature of temporally multiple DTs and it synthesizes nonlinear DTs without decay or divergence. An EM-like algorithm iterating between sequence division and LDS' fitting is adopted to learn the model parameters.

Index Terms— Temporally Multiple Dynamic Textures, Piecewise Linear Dynamic Systems, Temporal Segmentation

1. INTRODUCTION

Dynamic textures (DTs) are video sequences with spatially repetitive and temporally varying visual patterns that exhibit certain stationary properties over time, such as flames, moving escalators, fountains, waterfall, vegetation blowing in the wind, and so on. DT analysis and synthesis is an important research topic in image/video processing and computer vision. During the past decades, various methods have been proposed for this purpose. A typical parametric model for DT synthesis was proposed by Doretto et al. [1], where a DT is modeled as a Linear Dynamic System (LDS) by estimating system states using Principal Component Analysis (PCA) and describing their trajectory as time evolves. The LDS-based model has achieved success in various vision tasks such as synthesis, editing [2], segmentation [3] and video coding [4][5].

However, the visual quality of the synthesized DTs using basic LDS is often unsatisfactory. To this end, many LDS variants have been proposed. Yuan et al. [6] analyze the stability of the basic LDS and present closed-loop LDS (CLDS) to deal with the non-stable case which may generate longer DTs without any visible discontinuities. Chan and Vasconcelos [7] propose to learn a nonlinear observation function using kernel-PCA. The resulting kernel DT is capable of modeling a wider range of video motion such as chaotic motion or camera motion. Nonlinear models have been proposed to model nonlinear DTs, which use nonparametric methods to generate future observations. Masiero and Chiuso [8] generate samples from the estimated state distribution. In [9], observed data are embedded into a higher dimensional phase space and warped to a strange attractor, followed by predicting future observations using information from the strange attractor. Another class of methods model nonlinear DTs with multiple LDS', which indeed enhance the representation ability of original LDS. Early works include switching-based linear models [10][11] which select the system parameters or switch between different LDS outputs at each time step. Then, Chan et al. [12][13] introduce mixture of DTs to model multiple, co-occurring DTs and achieve superior performance in the problems of video clustering and motion segmentation.

In spite of the previous works, modeling DTs involved in real-world videos is still a challenging problem. For example, due to camera switching or instantaneous scene changes, a video sequence may exhibit complex dynamic nature as shown in Figure 1. Apparently, the dynamic is globally nonlinear which cannot be captured by single linear dynamic model as did in [1][6][7]. Nonlinear models in [8][9] cannot be given explicitly and handle these multiple linear DTs. Previous methods using multiple linear systems will fail to deal with such problem either. Mixture of DTs [12][13] can model multiple DTs in spatial domain, but cannot be applied in the case of temporally multiple DTs. Switching-based linear systems [10][11] have difficulties in learning a large number of switches along the transitions.

In this paper, we consider the real-world nonlinear DTs as temporally multiple linear DTs and propose piecewise linear dynamic systems (PLDS) for modeling and synthesis. Instead of manually splitting video into simpler DT pieces and then modeling each with an LDS, PLDS simultaneously decides the temporal segmentation, models each DT segment with an LDS and the whole DT by switching between the LDS'. Our method is inspired by [14], which synthesizes human dance motion with a collection of LDS' and uses a transition matrix to represent the relationship between motion textons. The advantages of PLDS include: (1) it can capture the stochastic and dynamic nature of temporally multiple DTs; (2) it can synthesize complex DTs without decay or divergence; (3) it can be effectively learned by an EM-like algorithm.

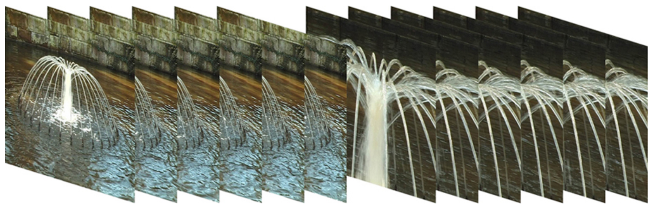

Fig. 1. A sequence that consists of temporally two DTs.

The rest of this paper is organized as follows. Section 2 reviews the basic DT synthesis using LDS. Section 3 describes the PLDS to model temporally multiple DTs and the learning algorithm. Section 4 presents some experimental results. Finally we conclude the paper and present some future works in Section 5.

2. DT SYNTHESIS USING LDS

We first give a brief introduction to LDS model and show how it can be used to synthesize DT [1]. For a given sequence of length T, we represent each frame with a column observation vector $y_t \in \mathbb{R}^m, 1 \leq t \leq T$. Let $Y = [y_1, y_2, ..., y_T]$, we model y_t as the output of an LDS as

$$\left\{ \begin{array}{lcll} x_{t+1} & = & Ax_t + Bv_t, & Bv_t \sim \mathcal{N}(0, Q), \\ y_t & = & y_0 + Cx_t + w_t, & w_t \sim \mathcal{N}(0, R). \end{array} \right. \tag{1}$$

where $x_t \in \mathbb{R}^n, n \ll m$ is the hidden state vector at time t, $A \in \mathbb{R}^{n\times n}$ models the dynamic of hidden states, $C \in \mathbb{R}^{m\times n}$ maps the hidden state to the output of the system. $y_0 \in \mathbb{R}^m$ is the mean of observation vectors, $Bv_t \sim \mathcal{N}(0, Q)$ and $w_t \sim \mathcal{N}(0, R)$ are Gaussian white noises, where $Q = BB^T$.

Doretto et al. [1] introduced a fast but suboptimal method for computing system parameters, i.e., (y_0, C, A, B, R), using a PCA-based approach. In the training process, an SVD

$$Y - Y_0 = U\Sigma V^T, \tag{2}$$

is performed where $Y_0 = [y_0, ..., y_0]$ and $y_0 = \frac{1}{T}\sum_{t=1}^T y_t$. System parameters are then estimated as

$$C = U \quad \text{and} \quad X = \Sigma V^T, \tag{3}$$

where $X = [x_1, ..., x_T]$ are estimated hidden states of the system. These hidden states are used to compute a suboptimal parameter matrix A using least squares estimation as

$$A = X_{2:T}X_{1:T-1}^{\dagger}, \tag{4}$$

where † represents pseudoinverse. Also,

$$Q = \frac{1}{T-1}\sum_t v'_t(v'_t)^T, \tag{5}$$

where $v'_t = x_{t+1} - Ax_t$. The model fitting error is defined as $\frac{1}{T-1}\sum_t ||v'_t||_2^2$.

The learnt LDS model is then used to reconstruct or synthesize a new sequence. Given an initial hidden state x_1, the sampling noise Bv_t can drive the system matrix A to generate new states, which are then mapped to high-dimensional observation vectors to form a new sequence of DT. However, this model often fails to produce videos with satisfactory quality or arbitrary length even for simple input sequences [6].

3. NONLINEAR DT SYNTHESIS

3.1. PLDS Model

For more accurate representation of realistic DTs, we propose the PLDS model. Given video sequence $Y_{1:T}$ consisting of temporally multiple DTs as shown in Figure 1, suppose N_t kinds of DTs are involved that can be modeled by $S = \{S_1, S_2, ..., S_{N_t}\}$, where S_i denote an LDS as well as its effective parameters $S_i = \{y_0^i, C^i, A^i, B^i\}$. Our objective is to automatically divide the sequence into N_p pieces (apparently $N_p \geq N_t$) and model each piece by one of the N_t LDS' such that the total fitting error is minimized.

The PLDS have additional discrete parameters besides S, the labels $L = \{l_1, l_2, ..., l_{N_p}\}$ and the segmentation points $H = \{h_1, h_2, ..., h_{N_p}\}$ of the N_p pieces. As illustrated in Figure 2, l_i indicates which LDS is used to model the i-th video piece, and h_i denotes the beginning frame of the i-th piece. Practically, we set a minimum length constraint for all pieces, i.e., $h_{i+1} - h_i \geq T_{min}, \forall i$.

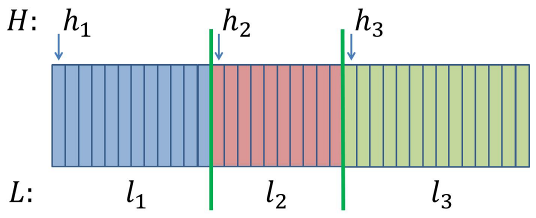

Fig. 2. An example illustrating how a sequence be divided and pieces be labeled using discrete parameters H and L.

3.2. Algorithm to Fit PLDS

The PLDS learns parameters $\{H, L, S\}$ to find the best division of the sequence and the most exact labels of pieces that get the minimum fitting error, by optimizing the objective,

$$\min_{H,L,S}\sum_{i=1}^{N_p} m(Y_{h_i:h_{i+1}-1}, S_{l_i}). \tag{6}$$

For any s, t and i, $m(Y_{s:t}, S_i)$ is the fitting error which needs a redefinition. That is, given any LDS S_i and a piece of video frames $Y_{s:t} = [y_s, y_{s+1}, ..., y_t]$, how well the LDS fit the video clip. In [14], probability likelihood $P(Y_{s:t}|S_i)$ is adopted. For video reconstruction, we should measure the difference between reconstructed and input videos. So the fitting error in Section 2 is also not proper and we can use other measures such as the average norm error or PSNR between reconstructed and original frames. Experiments show that strict measures can improve the results, such as the maximum norm error or the minimum PSNR, i.e.,

$$m(Y_{s:t}, S_i) = \max_{s \le k \le t} \parallel Y_k^{rec} - Y_k \parallel_2, \qquad (7)$$

or

$$m(Y_{s:t}, S_i) = -\min_{s \le k \le t} \text{PSNR}(Y_k^{rec}, Y_k), \qquad (8)$$

where $Y_{s:t}^{rec}$ is the reconstructed sequence of LDS S_i with initial hidden state $x_s^{rec} = (C^i)^\dagger \cdot (Y_s - y_0^i)$. We observe a complexity reduction that $Y_{s:t}^{rec}$ can be used to directly obtain $m(Y_{s:t'}, S_i)$ for any $t' \in (s,t)$ by just taking a part of frames $Y_{s:t'}^{rec}$ from $Y_{s:t}^{rec}$.

We adopt an EM-like iterative strategy to learn the model parameters $\{H, L, S\}$. Our algorithm is inspired by the parameter estimation algorithm proposed in [14] and each iteration step includes two parts, inferencing $\{H, L\}$ and fitting multiple LDS' S. In inferencing part, a dynamic programming algorithm is performed to efficiently compute the optimal $\{H, L\}$ while S is available due to the fitting part of the last iteration step. In fitting part, we update S using $\{H, L\}$ obtained from the inferencing part of the current iteration step. The algorithm loops until it converges to a local optimum or a maximum number of iterations is reached.

Details of dynamic programming algorithm of the inferencing part are given below. We define $G_n(t)$ as the minimum possible fitting error when dividing the sequence $Y_{t:T}$ into n pieces. Let $E_n(t)$ and $F_n(t)$ be the label and ending frame of the first piece of $Y_{t:T}$ when achieving $G_n(t)$.

1. Initialization

$$G_1(t) = \min_{1 \le i \le N_t} m(Y_{t:T}, S_i), \quad 1 \le t \le T - T_{min} + 1,$$

$$E_1(t) = \arg \min_{1 \le i \le N_t} m(Y_{t:T}, S_i).$$

2. Loop for $n = 2 : \frac{T}{T_{min}}, \quad t = T - n \cdot T_{min} + 1 : -1 : 1$

$$G_n(t) = \min_{\substack{1 \le i \le N_t \\ t+T_{min}-1 \le b \le T-(n-1)T_{min}}} \max(G_{n-1}(b+1), m(Y_{t:b}, S_i)),$$

$$E_n(t), F_n(t) = \arg \min_{i,b} \max(G_{n-1}(b+1), m(Y_{t:b}, S_i)).$$

Notice that $m(Y_{t:b}, S_i)$ can be directly obtained after computing $m(Y_{t:T-(n-1)T_{min}}, S_i)$.

3. Final solution

$$G(1) = \min_{1 \le n \le \frac{T}{T_{min}}} G_n(1),$$

$$N_p = \arg \min_{1 \le n \le \frac{T}{T_{min}}} G_n(1).$$

4. Backtracking
Backtrack the segmentation points H and the labels L using $E_n(t)$ and $F_n(t)$. For T frames and N_t LDS', the computational complexity is $O(N_t T^2)$.

After Inferencing $\{H, L\}$, we concatenate pieces that have the same label into a new sequence and learn an LDS for it using the algorithm described in Section 2. There are small modifications when dealing with boundaries of the pieces. Simultaneously, we keep all the starting hidden states of the pieces used for reconstructing the whole sequence using the method in Section 2. All the learnt LDS' compose a new S. That is, we get S updated in fitting part of each iteration.

Initial $\{H, L, S\}$ is given by a greedy approach. Assume $Y_{1:s}$ has been divided into pieces, each has been labeled, and $S = \{S_1, S_2, ..., S_i\}$. Then an LDS in S which best fits subsequent $T' = \min(T_{min}, T - s)$ frames $Y_{s+1:s+T'}$ is chosen and we label the frames with it if the fitting error is under a specified threshold ρ. Otherwise, no LDS is chosen and we learn a new LDS S_{i+1} using $Y_{s+1:s+T'}$. In any case the piece $Y_{s+1:s+T'}$ is labeled and we add subsequent frames to the piece one by one until the fitting error is above ρ. Then we get a new piece and add S_{i+1} into S, and repeat until the entire sequence is processed. The threshold ρ and T_{min} are two important parameters in initialization step.

4. EXPERIMENTS

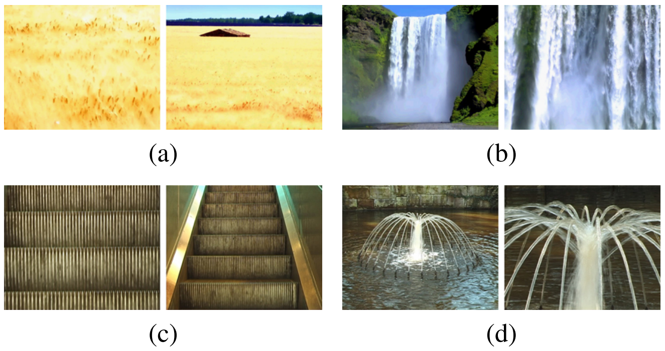

Fig. 3. Four sequences that consist of multiple DTs. Two representative frames are shown for each sequence.

We have verified our proposed algorithm on some texture sequences.[1] Figure 3 shows four sequences involved in our experiments that consist of temporally multiple DTs. The sequences (a) and (b) are directly cut from real videos. The sequences (c) and (d) are manually concatenated using two DTs from the DynTex database [15]. Each of the eight subsequences has 60 frames. In our experiments, we adopt the minimum PSNR as the measure of fitting error. We set the thresh-

[1]All experimental data and results are available from our web page `http://vipl.ict.ac.cn/paperpage/PLDS/`

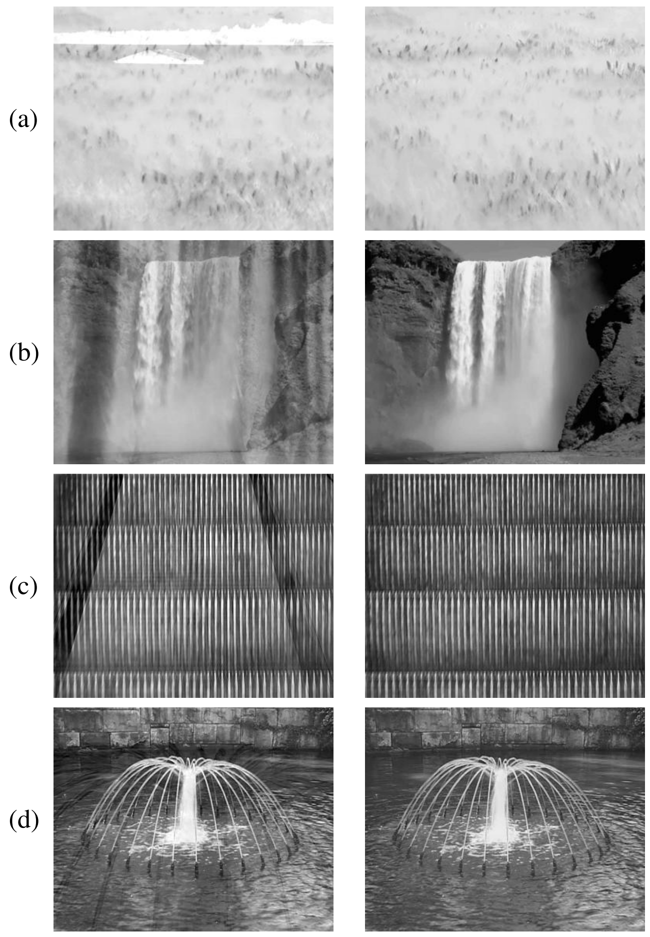

Fig. 4. Comparison between basic LDS method and our PLD-S method. The left column are frames synthesized by basic LDS, and the right are by our PLDS.

old ρ in model initialization varying from 15 to 50. T_{min} is 25 and iteration number between 2 and 4 is enough.

Figure 4 shows visual quality comparisons between basic LDS method [1] and our PLDS method. Our approach outperforms the LDS, because single LDS cannot model multiple DTs. The results of basic LDS method show a phenomenon that two DTs are mixed into the same frames. The quantitative comparisons are shown in Table 1, where our method gives significantly smaller norm error between reconstructed and original frames. The number 61 in each H reveals that the four sequences are correctly divided.

We also compare our results with those generated by the CLDS method [6]. Because the random selection procedure and smooth operation in CLDS make it infeasible to compute the reconstruction errors, we only give qualitative comparisons in Figure 5. We can see that the smooth operation in CLDS does not always work, especially for synthesizing nonlinear DTs.

Sequence	Basic LDS	PLDS	H (A total of 120)
(a)	89.7	31.5	(1,28,61)
(b)	248.6	26.2	(1,61,86)
(c)	239.4	68.0	(1,29,61,88)
(d)	142.8	90.2	(1,61,90)

Table 1. Quantitative comparison between LDS and PLDS. The second and third columns show norm errors. The fourth column provides segmentation points H after fitting PLDS.

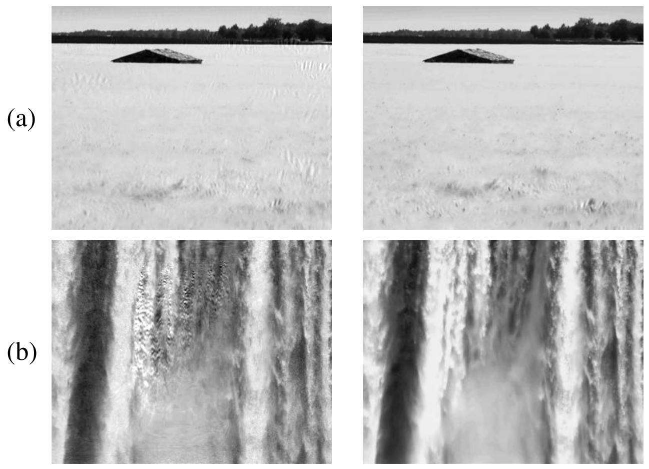

Fig. 5. Comparison between CLDS and PLDS. Frames on the left are synthesized by CLDS, and the right by PLDS.

5. CONCLUSION AND DISCUSSION

In this paper, we propose a piecewise linear DTs model to synthesize temporally multiple DTs. An EM-like algorithm iterating between sequence division and LDS' fitting is adopted to learn the model parameters. Experimental results verify that our method can capture the stochastic and dynamic nature of temporally multiple DTs by correct division and labeling.

Some future work will be pursued to make the proposed method more applicable. The number of parameters can be reduced by replacing the SVD in the method with higher-order tensor decomposition, such as HOSVD [16]. To model and synthesize more real-world videos, we can apply PLD-S in both temporal and spatial domains. Other possibilities include integrating the method into video coding framework.

6. ACKNOWLEDGEMENT

This work is partially supported by the National Science Foundation of China under contract No. 61272319, the National Basic Research Program of China (973 Program) under contract 2009CB320900, and Beijing Natural Science Foundation (New Technologies and Methods in Intelligent Video Surveillance for Public Security) under contract No. 4111003.

7. REFERENCES

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

[2] G. Doretto and S. Soatto, “Editable dynamic textures,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2003, vol. 2.

[3] G. Doretto, D. Cremers, P. Favaro, and S. Soatto, “Dynamic texture segmentation,” in *Proceedings of the IEEE International Conference on Computer Vision*, October 2003.

[4] A. Stojanovic, M. Wien, and J.-R. Ohm, “Dynamic texture synthesis for h.264/avc inter coding,” in *Proceedings of the IEEE International Conference on Image Processing*, October 2008, pp. 1608–1611.

[5] B. Zhou, F. Zhang, and L. Peng, “Compact representation for dynamic texture video coding using tensor method,” *IEEE Transactions on Circuits and Systems for Video Technology*, 2012.

[6] L. Yuan, F. Wen, C. Liu, and H.Y. Shum, “Synthesizing dynamic texture with closed-loop linear dynamic system,” in *Proceedings of the European Conference on Computer Vision*, 2004, pp. 603–616.

[7] A.B. Chan and N. Vasconcelos, “Classifying video with kernel dynamic textures,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2007, pp. 1–6.

[8] A. Masiero and A. Chiuso, “Non linear temporal textures synthesis: a monte carlo approach,” in *Proceedings of the European Conference on Computer Vision*, 2006, pp. 283–294.

[9] A. Basharat and M. Shah, “Time series prediction by chaotic modeling of nonlinear dynamical systems,” in *Proceedings of the IEEE International Conference on Computer Vision*. IEEE, 2009, pp. 1941–1948.

[10] R.H. Shumway and D.S. Stoffer, “Dynamic linear models with switching,” *Journal of the American Statistical Association*, vol. 86, pp. 763–769, 1991.

[11] Z. Ghahramani and G.E. Hinton, “Switching state-space models,” *University of Toronto Technical Report CRG-TR-96-3, Department of Computer Science*, 1996.

[12] A.B. Chan and N. Vasconcelos, “Mixtures of dynamic textures,” in *Proceedings of the IEEE International Conference on Computer Vision*, October 2005, vol. 1, pp. 641–647.

[13] A.B. Chan and N. Vasconcelos, “Modeling, clustering, and segmenting video with mixtures of dynamic textures,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 5, pp. 909–926, May 2008.

[14] Y. Li, T. Wang, and H.Y. Shum, “Motion texture: A two-level statistical model for character motion synthesis,” *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 465–472, 2002.

[15] R. Péteri, S. Fazekas, and M.J. Huiskes, “Dyntex: A comprehensive database of dynamic textures,” *Pattern Recognition Letters*, vol. 31, no. 12, pp. 1627–1632, 2010.

[16] R. Costantini, L. Sbaiz, and S. Susstrunk, “Higher order svd analysis for dynamic texture synthesis,” *IEEE Transactions on Image Processing*, vol. 17, no. 1, pp. 42–52, January 2008.