Synthesizing Dynamic Texture with Closed-loop Linear Dynamic System

Lu Yuan[1,2⋆], Fang Wen[1], Ce Liu[3⋆], and Heung-Yeung Shum[1]

[1] Visual Computing Group, Microsoft Research Asia, Beijing 100080, China,
[2] Department of Electronic Engineering, Tsinghua University, Beijing 100084, China,
[3] CSAIL, Massachusetts Institute of Technology, Cambridge, 02139, USA

Email: yuanl02@mails.tsinghuat.edu.cn, t-fangw@microsoft.com
celiu@mit.edu, hshum@microsoft.com

Abstract. Dynamic texture can be defined as a temporally continuous and infinitely varying stream of images that exhibit certain temporal statistics. Linear dynamic system (LDS) represented by the state-space equation has been proposed to model dynamic texture[12]. LDS can be used to synthesize dynamic texture by sampling the system noise. However, the visual quality of the synthesized dynamic texture using noise-driven LDS is often unsatisfactory. In this paper, we regard the noise-driven LDS as an open-loop control system and analyze its stability through its pole placement. We show that the noise-driven LDS can produce good quality dynamic texture if the LDS is oscillatory. To deal with an LDS not oscillatory, we present a novel approach, called closed-loop LDS (CLDS) where feedback control is introduced into the system. Using the succeeding hidden states as an input reference signal, we design a feedback controller based on the difference between the current state and the reference state. An iterative algorithm is proposed to generate dynamic textures. Experimental results demonstrate that CLDS can produce dynamic texture sequences with promising visual quality.

1 Introduction

Dynamic texture, also known as temporal texture or video texture, can be defined as a temporally continuous and infinitely varying stream of images that exhibit certain temporal statistics. For example, it could be a continuously gushing fountain, a flickering fire, an escalator, and a smoke puffing slowly from the chimney – each of these phenomena possesses an inherent dynamics that a general video may not portray. Dynamic texture can be used for many applications such as personalized web pages, screen savers and computer games.

Dynamic texture analysis and synthesis have recently become an active research topic in computer vision and computer graphics. Similar to 2D texture synthesis, the goal of dynamic texture synthesis is to generate a new sequence that is similar to, but somewhat different from, the original video sequence.

⋆ This work was done when the authors worked in Microsoft Research Asia.

Ideally the generated video can be played endlessly without any visible discontinuities. Moreover, it is desirable that dynamic texture be easily edited and be efficient for storage and computation.

Most recent techniques on dynamic texture synthesis can be categorized into nonparametric and parametric methods. The nonparametric methods generate dynamic textures by directly sampling original pixels[15], frames[11], wavelet-structures[1], 2D patches[8] and 3D voxels[8] from a training sequence and usually produce high-quality visual effects. Compared with the nonparametric approaches, the parametric methods provide better model generalization and understanding of the essence of dynamic texture. The typical parametric models include Szummer and Picard's STAR model[13], Wang and Zhu's movetons representation[14], and Soatto et al's linear dynamic system (LDS) model[12]. They are very helpful for the tasks such as dynamic texture editing[4], recognition[10], segmentation[3] and image registration[5]. However, most parametric models are less likely to generate dynamic textures with the same quality as the nonparametric models, in particular for the videos of natural scenes.

In this paper, we argue that a carefully designed parametric model can compete well with any existing nonparametric model, e.g. [8], in terms of visual quality of the synthesized dynamic texture. Our work is inspired by the former work of noise-driven LDS for dynamic texture synthesis [12]. From a viewpoint of control theory, the noise-driven LDS is indeed an open-loop control system which is likely to be contaminated by noise. Based on the analysis of the stability of the open-loop LDS, we find that it is the problems of the pole placement and model fitting error that prevent the previous approaches from generating satisfactory dynamics textures.

In consequence, we propose a novel closed-loop LDS (CLDS) approach to synthesizing dynamic texture. In our approach, the difference between the reference input and the synthesized output is computed as a feedback control signal. Using a feedback controller, the whole closed-loop system can minimize the model fitting error and improve the pole placement. Our experimental results demonstrate that our approach can produce visually promising dynamic texture sequences.

2 Dynamic Texture Synthesis using LDS

Under the hypothesis of temporal stationarity, Soatto et al[12] adopted a linear dynamic system (LDS) to analyze and synthesize dynamic texture. The state-space representation of their model is given by

$$\begin{cases} \boldsymbol{x}_t = A\boldsymbol{x}_{t-1} + \boldsymbol{v}_t, \quad \boldsymbol{v}_t \sim \mathcal{N}(0, \Sigma_v) \\ \boldsymbol{y}_t = C\boldsymbol{x}_t + \boldsymbol{w}_t, \quad \boldsymbol{w}_t \sim \mathcal{N}(0, \Sigma_w) \end{cases} \tag{1}$$

where $\boldsymbol{y}_t \in \mathbb{R}^n$ is the observation vector; $\boldsymbol{x}_t \in \mathbb{R}^r, r \ll n$ is the hidden state vector; A is the system matrix; C is the output matrix and $\boldsymbol{v}_t, \boldsymbol{w}_t$ are Gaussian white noises.

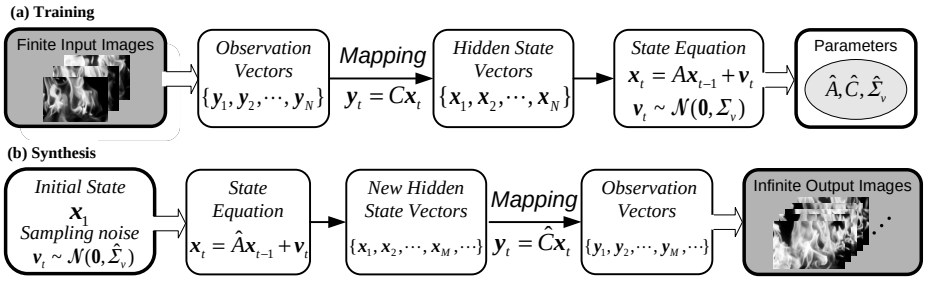


Fig. 1. Framework of dynamic texture analysis and synthesis using LDS. (a) In the training process, a finite sequence of input images is used to train an LDS model with the system matrix A, the observation matrix C and the system noise $\boldsymbol{v}_t$ (with its variance Σ_v). (b) In the synthesis process, a new (possibly infinite) sequence is generated using the learnt LDS and by sampling system noise.

Fig.1 illustrates the dynamic texture analysis and synthesis framework using the above LDS. To train the LDS model, a finite sequence of images $\{I_t\}_{t=1}^N$ subtracted by the mean image are concatenated to column vectors as observation vectors $\{\boldsymbol{y}_t\}_{t=1}^N$. These observation vectors are mapped into hidden state vectors $\{\boldsymbol{x}_t\}_{t=1}^N$ in a lower dimensional space by SVD analysis. We use these hidden states to fit a linear dynamic system and identify the parameters of the system through maximum likelihood estimation (MLE). The learnt LDS model is then used to synthesize a new sequence. Given an initial hidden state $\boldsymbol{x}_1$, the sampling noise $\boldsymbol{v}_t \sim \mathcal{N}(0, \hat{\Sigma}_v)$ can drive the system matrix $\hat{A}$ to generate new state vectors, which are then mapped to the high-dimensional observation vectors to form a new sequence of dynamic texture.

The above dynamic texture model has a firm analytic footing in system identification. However, it often fails to produce satisfactory results for simple sequences such as those shown in **Fig.7**(b)(f). An open question is: how well one can synthesize a new sequence from the learnt LDS.

3 Analysis of Open-loop LDS

According to control theory, the noise-driven dynamic texture synthesis process ($\boldsymbol{x}_t = A\boldsymbol{x}_{t-1} + \boldsymbol{v}_t$) illustrated in **Fig.1**(b) can be regarded as a basic open-loop LDS, as shown in **Fig.3**(a). The stability of an open-loop LDS is determined by its poles[6].

The poles of the system are exactly defined as the eigenvalues of system matrix A [9], which take complex values. In other words, the location of the pole on the 2D pole plot (i.e. the polar coordinate system) is determined by the corresponding eigenvalue. If the magnitude of an eigenvalue is less than one, the pole is inside the unit circle in the pole plot, called a *stable pole* since this eigen-component will vanish eventually. If the magnitude equals one, the pole is on the unit circle, called an *oscillatory pole* because the magnitude of this

eigen-component will remain constant while the phase may change periodically. If the magnitude is greater than one, the pole is outside the unit circle, called an *unstable pole* because this eigen-component will magnify to infinity. Clearly, a control system cannot contain any unstable poles.

We consider the state equation $\boldsymbol{x}_t = A\boldsymbol{x}_{t-1}$ ignoring the noise term, where $A \in \mathbb{R}^{r\times r}$. By eigenvalue decomposition, we have $A = Q\Lambda Q^{-1}$, where $\Lambda = \text{diag}\{\sigma_1, \sigma_2, \ldots, \sigma_r\}$ and $Q \in \mathbb{R}^{r\times r}$. $\{\sigma_i\}_{i=1}^r$ are the eigenvalues of A, namely poles. Given an initial state vector $\boldsymbol{x}_1$, the state vector at time t is given by

$$\boldsymbol{x}_t = A^{t-1}\boldsymbol{x}_1 = Q\Lambda^{t-1}Q^{-1}\boldsymbol{x}_1 = Q \cdot \text{diag}(\sigma_1^{t-1}, \cdots, \sigma_r^{t-1}) \cdot Q^{-1}\boldsymbol{x}_1.$$

For any eigenvalue or pole σ_i we have

$$\lim_{t\to\infty} \left|\sigma_i^{t-1}\right| = \begin{cases} 0 & |\sigma_i| < 1, \\ 1 & |\sigma_i| = 1, \\ \infty & |\sigma_i| > 1. \end{cases}$$

There are three interesting cases with different pole distributions.

(a) All the poles of the dynamic system are stable, i.e. $\forall i \in \{1, 2, \cdots, r\} : |\sigma_i| < 1$. Then $\lim_{t\to\infty} \|\boldsymbol{x}_t\|_2 = 0$, where $\|\boldsymbol{x}_t\|_2$ represents the energy of the state vector. All the stable poles contribute to the decay of the dynamic system. This system is called a *stable system* (in accordance to the name of poles).

(b) Some poles are oscillatory and the rest are stable, i.e. $\forall i : |\sigma_i| \leqslant 1$ and $\exists j : |\sigma_j| = 1$. Then $\lim_{t\to\infty} \|\boldsymbol{x}_t\|_2 = c$, where c is a constant. We call it an *oscillatory system*.

(c) There exist unstable poles, namely $\exists i : |\sigma_i| > 1$. Then $\lim_{t\to\infty} \|\boldsymbol{x}_t\|_2 = \infty$. The unstable poles contribute to the divergence of the dynamic system. With such an *unstable system*, one cannot generate an infinite sequence.

These cases are further illustrated by dynamic texture synthesis results from three typical sequences.

(a') The poles of the FOUNTAIN sequence are all stable (**Fig.2**(d)). Thus, the generated dynamic sequence decays gradually as shown in **Fig.2**(a).

(b') Since most poles of the ESCALATOR sequence are oscillatory and the rest are stable (**Fig.2**(e)), the whole dynamic system is oscillatory and the generated sequence forms a satisfactory loop as shown in **Fig.2**(b).

(c') The FIRE sequence has two unstable poles (**Fig.2**(f)), which lead to the divergence of the dynamic system. This explains why the intensity of the synthesized fire saturates as shown in **Fig.2**(c).

The key observation from the synthesis and synthesized examples above is that for dynamic texture synthesis, the learnt LDS must be an *oscillatory system*. Unfortunately, such a requirement on the learnt system matrix A is too demanding to be satisfied in practice.

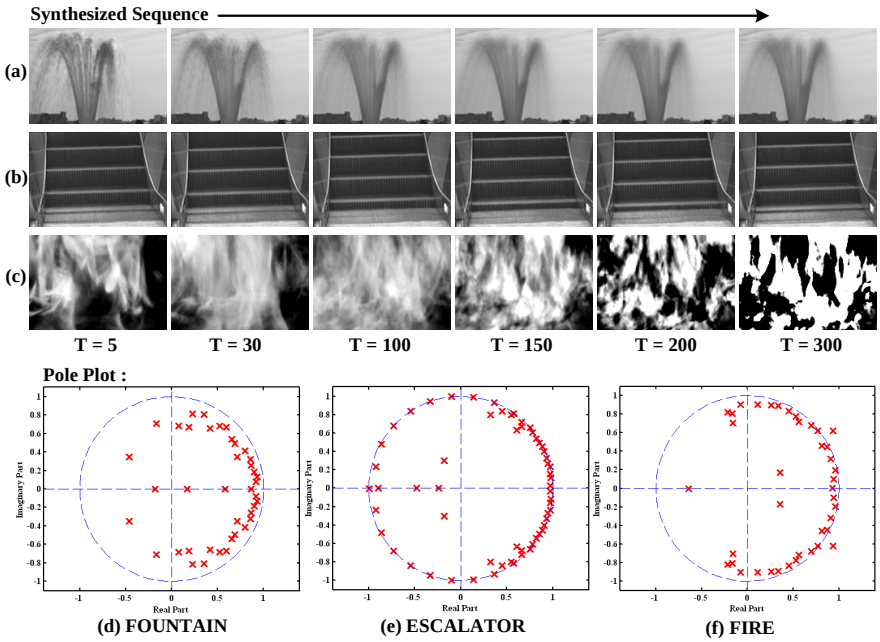


Fig. 2. Effect of poles of system: (a)(b)(c) show the synthesized sequences at different times. (a) Decaying FOUNTAIN sequence. (b) Oscillatory ESCALATOR sequence. This is the desirable result for dynamic texture. (c) Divergent FIRE sequence. (d) The positions of all the FOUNTAIN's poles (denoted by red crosses) are inside the unit circle. (e) Most poles of ESCALATOR are on the unit circle and others are inside the unit circle. (f) Two poles of FIRE are outside the unit circle and others are within or on the unit circle.

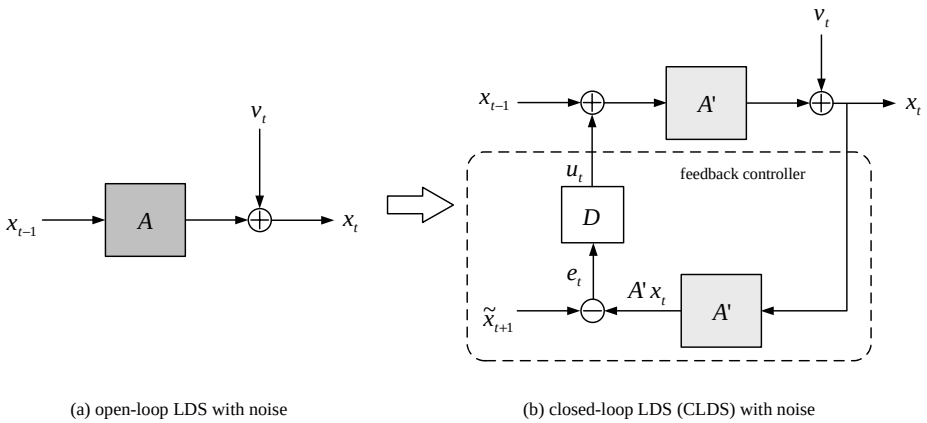
4 Closed-loop LDS

If the learnt LDS from the input video sequence is not an oscillatory system, we propose to change the open-loop LDS to a closed-loop LDS (CLDS), as illustrated in **Fig.3**(b), so that we can still synthesize dynamic texture with good visual quality. The overall goal of feedback control is to cause the output variable of a dynamic process to follow a desired reference variable accurately, regardless of any external disturbance in the dynamics of the process[7].

4.1 A simple CLDS

The key component in the proposed CLDS is the definition of the error term which turns to be the control signal, given as

$$\begin{cases} \boldsymbol{x}_t = A'\boldsymbol{x}_{t-1} + A'\boldsymbol{u}_t + \boldsymbol{v}_t, \quad v_t \sim \mathcal{N}(0, \Sigma_v) \\ \boldsymbol{u}_t = D\boldsymbol{e}_t \\ \boldsymbol{e}_t = \tilde{\boldsymbol{x}}_{\boldsymbol{t+1}} - A'\boldsymbol{x}_t \\ \boldsymbol{y}_t = C\boldsymbol{x}_t + \boldsymbol{w}_t, \quad \boldsymbol{w}_t \sim \mathcal{N}\left(0, \Sigma_w\right) \end{cases} \tag{2}$$



(a) open-loop LDS with noise (b) closed-loop LDS (CLDS) with noise

Fig. 3. Comparison of frameworks between open-loop LDS and closed-loop LDS: In the figure, only the state-space representation is illustrated. In both models, $\boldsymbol{x}_t$ represents the hidden state at time t in low-dimensional state-space in two frameworks; $\boldsymbol{v}_t$ represents the noise of the system. A' represents the system matrix of our system while A is the system matrix of open-loop LDS. $\boldsymbol{u}_t$ is the control signal; D is the control matrix in our system and $\boldsymbol{e}_t$ is the difference between reference input and output.

where $\boldsymbol{y}_t$ is the observation vector; $\boldsymbol{x}_t$ is the state vector; C is the output matrix; $\boldsymbol{e}_t$ is the difference between the reference and the rough estimate of the next state; D is the proportional control matrix of the error term and A' is the system matrix. In the state-space equation above, the current state $\boldsymbol{x}_t$ comprises three parts: 1) the rough estimation of the current state, i.e. $A'\boldsymbol{x}_{t-1}$, 2) the control signal $\boldsymbol{u}_t$ proportional to the difference between the reference input for next state $\tilde{\boldsymbol{x}}_{\boldsymbol{t+1}}$ and the rough estimation of the next state $A'\boldsymbol{x}_t$ from the output current state $\boldsymbol{x}_t$, and 3) sampled noise $\boldsymbol{v}_t$.

Clearly the open-loop LDS discussed previously (**Fig.3**(a)) is a special case of the proposed CLDS (**Fig.3**(b)) when D is zero.

4.2 A practical model

The CLDS above is of the simplest form, a first-order system and a proportional controller. In practice, estimation of the current state vector $\boldsymbol{x}_t$ using the first-order model is often insufficient. Instead, we employ a number of previous state vectors $\{\boldsymbol{x}_{t-p}, \cdots, \boldsymbol{x}_{t-1}\}$. The control signal $\boldsymbol{u}_t$ is based on the difference between the reference input and estimate of the succeeding state vectors $\{\boldsymbol{x}_{t+1}, \cdots, \boldsymbol{x}_{t+q}\}$. Then the state-space equation of CLDS is given by

$$\begin{cases} \boldsymbol{x}_t = \sum_{i=1}^{p} A_i \boldsymbol{x}_{t-i} + A_0 \boldsymbol{u}_t + \boldsymbol{v}_t, \quad \boldsymbol{v}_t \sim \mathcal{N}(0, \Sigma_v) \\ \boldsymbol{u}_t = \sum_{j=1}^{q} D_j \left(\tilde{\boldsymbol{x}}_{\boldsymbol{t+j}} - \sum_{i=1}^{p} A_i \boldsymbol{x}_{t+j-i} \right) \end{cases} \tag{3}$$

After substituting $\boldsymbol{u}_t$, we obtain the following equation:

$$\boldsymbol{x}_t = \sum_{i=-p}^{-1} \Phi_{p+1+i}\boldsymbol{x}_{t+i} + \sum_{j=1}^{q} \Phi_{p+j}\tilde{\boldsymbol{x}}_{t+j} + \boldsymbol{v}_t, \quad \boldsymbol{v}_t \sim \mathcal{N}(0, \Sigma_v) \tag{4}$$

where Φ_k represents the linear combination of A_i and D_j. In the training we replace the reference states $\tilde{\boldsymbol{x}}_{t+j}(j = 1, \ldots, q)$ with $\boldsymbol{x}_{t+j}(j = 1, \ldots, q)$ in Equation (4). Then the state equation (4) becomes a non-causal system, whose parameters can be derived by least squares estimation (LSE) [9].

$$\begin{aligned} &[\Phi_1, \Phi_2, \cdots \Phi_{p+q}] = G_0 \left[G_{-p}^T, \cdots, G_{-1}^T, G_1^T, \cdots, G_q^T \right]^{-1} \\ &\Sigma_v = \tfrac{1}{N-p-q} \left(R_{0,0} - \sum_{i=-p}^{q} \Phi_{i+p+1} R_{i,0} \right) \end{aligned} \tag{5}$$

where $G_i = [R_{i,-p}, \cdots, R_{i,-1}, R_{i,1}, \cdots, R_{i,q}], -p \le i \le q$ and $R_{i,j} = \sum_{t=1-i}^{N-j} \boldsymbol{x}_{t+i}\boldsymbol{x}_{t+j}^T$.

We follow the order determination algorithm described in [16] to choose the appropriate size of the temporal neighborhood $\Omega = p + q$ in Equation (4). In our implementation, the cost function of model fitting is defined as $\sigma_{p+q} = \frac{1}{N-p-q} \sum_{t=p+1}^{N-q} \theta_t^T \theta_t$, where $\theta_t = \boldsymbol{x}_t - \sum_{i=-p}^{-1} \Phi_{p+1+i}\boldsymbol{x}_{t+i} - \sum_{j=1}^{q} \Phi_{p+j}\boldsymbol{x}_{t+j}$. We initialize $p = 1, q = 1$ and then gradually increase p and q respectively until $\frac{\sigma_{p+q+1}}{\sigma_{p+q}} > \mu$.

5 Synthesis using CLDS

It is nontrivial to directly sample a closed-loop system. To sample CLDS, We first sample a reference sequence, and then use the system equation to smooth the discontinuity of the reference sequence. The synthesis algorithm is shown in the **Fig.4**.

Using the hidden state vectors $\{\boldsymbol{x}_t\}_{t=1}^N$ generated from the original image sequence $\{I_t\}_{t=1}^N$, we compute the state-to-state similarities and store them in a matrix $\boldsymbol{S} = \langle \boldsymbol{x}_{1:N}, \boldsymbol{x}_{1:N} \rangle = \{s_{i,j}\} (i, j = 1, 2, \ldots, N)$, where $s_{ij} = \frac{\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle}{\sqrt{\langle \boldsymbol{x}_i, \boldsymbol{x}_i \rangle \langle \boldsymbol{x}_j, \boldsymbol{x}_j \rangle}}$ and $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$ is the inner product of column vectors.

From the similarity matrix $\boldsymbol{S}$, we obtain the state transfer probability matrix $\boldsymbol{P}$ through an exponential function

$$P_{i,j} = P(\boldsymbol{x}_j | \boldsymbol{x}_i) = \begin{cases} \exp\{\gamma s_{i+1,j}\}, & i \neq j \\ 0, & i = j \end{cases} .$$

All the probabilities for any row of $\boldsymbol{P}$ are normalized so that $\sum_j P_{i,j} = 1$. In our implementation, γ was set as a number between 1 and 50.

By sampling original state vectors $\{\boldsymbol{x}_i\}_{i=1}^N$, we generate a reference sequence $\{\boldsymbol{x}_{k_1:k_1+\tau-1}, \boldsymbol{x}_{k_2:k_2+\tau-1}, \cdots \boldsymbol{x}_{k_h:k_h+\tau-1}\}$ by concatenating h short state clips such as $\boldsymbol{x}_{k_1:k_1+\tau-1}$ and $\boldsymbol{x}_{k_2:k_2+\tau-1}$. Each clip has τ frames and two neighboring clips

1. To synthesize M frames, set the clip size τ and select the first clip $\{\boldsymbol{x}_{k_1}, \boldsymbol{x}_{k_1+1}, \cdots, \boldsymbol{x}_{k_1+\tau-1}\}$ randomly from original states $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N\}$
2. Sample the next clip $\{\boldsymbol{x}_{k_2}, \boldsymbol{x}_{k_2+1}, \cdots, \boldsymbol{x}_{k_2+\tau-1}\}$ with $P(\boldsymbol{x}_{k_2}|\boldsymbol{x}_{k_1+\tau-1})$ in $\boldsymbol{P}$
3. Repeat 2, until h state clips $\{\boldsymbol{x}_{k_1:k_1+\tau-1}, \boldsymbol{x}_{k_2:k_2+\tau-1}, \cdots, \boldsymbol{x}_{k_h:k_h+\tau-1}\}$ are sampled. The previous $M(M \leq h \times \tau)$ reference states are used as the initial synthesis states $\{\boldsymbol{x}_1^{(0)}, \boldsymbol{x}_2^{(0)}, \cdots, \boldsymbol{x}_M^{(0)}\}$
4. Iterate $n = 1, 2, \cdots$ until $\left|\delta^{(n)} - \delta^{(n-1)}\right| < \varepsilon$
   (a) Sample noise $\boldsymbol{v}_t^{(n)}$ and update $\boldsymbol{x}_t^{(n)}$ by Equation(6) from $t = 1, 2, \cdots, M$
   (b) compute iterative error $\delta^{(n)}$
5. Output $\{\boldsymbol{x}_1^{(n)}, \boldsymbol{x}_2^{(n)}, \cdots, \boldsymbol{x}_M^{(n)}\}$

Fig. 4. Synthesis algorithm for the closed-loop LDS

$\boldsymbol{x}_{k_1:k_1+\tau-1}$ and $\boldsymbol{x}_{k_2:k_2+\tau-1}$ are chosen so that the last state $\boldsymbol{x}_{k_1+\tau-1}$ in clip $\boldsymbol{x}_{k_1:k_1+\tau-1}$, and the first state $\boldsymbol{x}_{k_2}$ in clip $\boldsymbol{x}_{k_2:k_2+\tau-1}$, are close according to state-transfer matrix $\boldsymbol{P}$. This is similar, in spirit, to the idea of video texture[11]. However, there exist visible jumps between these short clips in this initial concatenated sequence. We employ an iterative refinement algorithm to successively improve the discontinuity in the output sequence. Specifically, we use the system equation iteratively to smooth out the discontinuity to obtain $\boldsymbol{x}_t$.

$$\boldsymbol{x}_t^{(n)} = \sum_{i=-p}^{-1} \Phi_{p+1+i} \boldsymbol{x}_{t+i}^{(n)} + \sum_{j=1}^{q} \Phi_{p+j} \boldsymbol{x}_{t+j}^{(n-1)} + \boldsymbol{v}_t^{(n)}, \quad \boldsymbol{v}_t^{(n)} \sim \mathcal{N}(0, \Sigma_v) \qquad (6)$$

where $\boldsymbol{x}^{(n)}, \boldsymbol{x}^{(n-1)}$ represent the values of $\boldsymbol{x}$ before and after the n^{th} iteration respectively. When the whole state sequence satisfy the statistic process of Equation (4), the sampled values of the whole sequence is unchanged after iteration. Therefore, the n^{th} iterative error can be defined as

$$\delta^{(n)} = \sum_{t=1}^{M} \left\| \boldsymbol{x}_t^{(n)} - \boldsymbol{x}_t^{(n-1)} \right\|_2.$$

The final state sequence is obtained iteratively until the iterative error can hardly drop down. Then we map these states in the low-dimensional space to the observations in the high-dimensional space and obtain the new image sequences.

In **Fig.5**, we show the initially concatenated reference video sequence, the intermediate sequence after 5 iterations and the final synthesized sequence. Although jumps are clearly visible between short clips in the initial sequence, CLDS was able to smooth over the whole sequence to produce visually appealing dynamic texture for the SMOKE-NEAR example.

Fig. 5. Improvement by iteration in synthesis: (a)-(c) respectively shows 4 frames around a jump before the 1^{st} iteration, after 5 iterations and after 30 iterations.

6 Experimental Results

We have synthesized many dynamic textures using our proposed CLDS. Our results are compared with those generated using previous nonparametric and parametric methods. All the original sequences are borrowed from the MIT Temporal Textures database[4] and Kwatra et al.'s web site[5]. All the results (videos) can be found in the project's web page[6]. In our experiments, we set the clip size τ varying from 5 to 20. Through manually choose μ and ε, the neighborhood size Ω is between 2 and 4, and iteration number n is between 5 and 50.

As shown in **Fig.7**, our approach outperforms not only the noise-driven LDS by Soatto et al[12], but also the improved open-loop LDS by Doretto et al[2] in terms of the visual quality of the synthesis results. For instance, to synthesize dynamic textures for SMOKE-FAR and FIRE sequences, results (**Fig.7**(b)(f)) using the basic open-loop LDS algorithm are less likely acceptable because the system is "unstable". Although Doretto et al attempted to solve this problem by scaling down the poles, the fitting error cannot be ignored and causes that the synthesized video (shown in **Fig.7**(c)(g)) still looks different from the original sequence. Our results show that the generated SMOKE-FAR and FIRE sequences closely resemble the original sequences as shown in **Fig.7**(d)(h).

Our results also compare favorably with those generated from nonparametric methods. Kwatra[8] has generated perhaps the most impressive dynamic texture synthesis results to date using the graph cut technique. Our results on Bldg9-FOUNTAIN and WATERFALL shown in **Fig.8** demonstrate that we can achieve similar visual quality using CLDS.

[4] ftp://whitechapel.media.mit.edu/pub/szummer/temporal-texture/
[5] http://www.cc.gatech.edu/cpl/projects/graphcuttextures/
[6] http://research.microsoft.com/asia/download/disquisition/dynamictexture(ECCV04_Supp).html

7 Discussion

Our experimental results demonstrate that CLDS can produce dynamic texture sequence with good visual quality. There are indeed two problems in the open-loop LDS which are addressed in CLDS, i.e. achieving oscillatory poles and minimizing fitting error.

The effect of CLDS can be observed from how it alters pole placement. Given a synthesized sequence using CLDS, we can compute its effective poles and compare with the poles of the corresponding open-loop LDS. From the FOUNTAIN and FIRE example, we observe significant improvement on pole placement. For the FOUNTAIN sequence shown in **Fig.6**(a), most stable poles have been moved towards the unit circle. Two poles are placed exactly on the unit circle, making the whole system oscillatory. For the FIRE sequence shown in **Fig.6**(b), two unstable poles from the open-loop LDS have been moved to the unit circle, making it possible to synthesize an infinite sequence of dynamic texture. Note that some stable poles of open-loop LDS close to the origin have been moved closer to the origin with the CLDS. This may cause some blurring in the synthesized results.

Another way to measure the effect of CLDS is to compute the model fitting error. We define the model fitting error as $\delta = \frac{1}{N \times r} \sum_{t=1}^{N} \|\boldsymbol{x}_t - \hat{\boldsymbol{x}}_{\boldsymbol{t}}\|_2^2$, $(\boldsymbol{x}_t \in \mathbb{R}^r)$, where $\hat{\boldsymbol{x}}_{\boldsymbol{t}}$ is the estimate of $\boldsymbol{x}_t$. In the basic LDS, $\hat{\boldsymbol{x}}_{\boldsymbol{t}} = A\boldsymbol{x}_{t-1}$. Doretto[2] relocated unstable poles to the inside of the unit circle. Therefore, the system matrix A is transformed to $\tilde{A}$ and $\hat{\boldsymbol{x}}_{\boldsymbol{t}} = \tilde{A}\boldsymbol{x}_{t-1}$. For CLDS, instead, $\hat{\boldsymbol{x}}_{\boldsymbol{t}} = A'\boldsymbol{x}_{t-1} + A'\boldsymbol{u}_t$, where $\boldsymbol{u}_t$ is defined in Equation (2). **Table 1** shows significant improvement in fitting error of CLDS over the basic LDS and Doretto's method. Doretto's pole relocation scheme cannot reduce the model fitting error. Large fitting error implies that the dynamics of the synthesized sequence (**Fig.7**(c)(g)) would deviate from the original training set (**Fig.7**(a)(e)).

Simple scaling of poles cannot simultaneously satisfy the two goals in generating dynamic texture using the open-loop LDS: making the system oscillatory and minimizing fitting error. On the other hand, CLDS can achieve both goals.

Table 1. Comparison of fitting errors of three methods

Original Sequence	FIRE	SMOKE-FAR	SMOKE-NEAR
Basic LDS method	55264	230.7	402.6
Doretto's method	55421	250.0	428.2
Our CLDS method	1170	21.4	34.4

8 Summary and Future Work

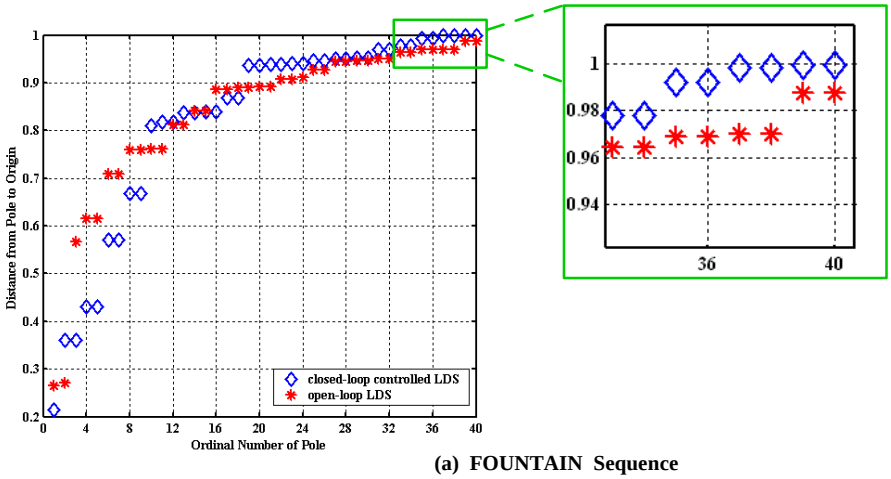
In this paper, we have analyzed the stability of open-loop LDS used in the dynamic texture model of [12]. We have found that the open-loop LDS must be oscillatory in order to generate an infinite sequence of dynamic texture from a

finite sequence of training images. For an LDS that is not oscillatory (stable or unstable), we propose to use feedback control to make it oscillatory. Specifically, we propose a closed-loop LDS (CLDS) to make the system oscillatory and minimize the fitting error. Our experimental results demonstrate our model can be used to synthesize a large variety of dynamic textures with promising visual quality.

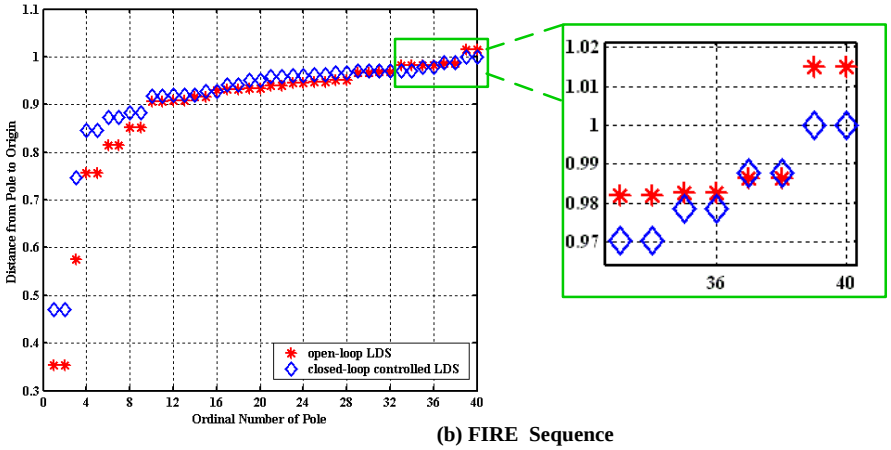
In future work, we plan to investigate whether a better controller (e.g. PID or Proportional-Integral-Derivative) would improve the synthesis quality. Another challenging problem is to model and synthesize non-stationary dynamic texture.

References

1. Z. Bar-Joseph, R. El-Yaniv, D. Lischinski and M. Werman. Texture mixing and texture movie synthesis using statistical learning. *IEEE Transactions on Visualization and Computer Graphics*, vol. 7, pp. 120-135, 2001.
2. G. Doretto, A. Chiuso, Y. N. Wu and S. Soatto. Dynamic Textures. *International Journal of Computer Vision*, vol. 2, pp. 91-109, 2003.
3. G. Doretto, D. Cremers, P. Favaro and S. Soatto. Dynamic Texture Segmentation. *In Proceedings of ICCV'03*, pp. 1236-1242, 2003.
4. G. Doretto and S. Soatto. Editable Dynamic Textures. *In Proceedings of CVPR'03*, pp. 137-142, 2003.
5. A. W. Fitzgibbon. Stochastic rigidity: Image registration for nowhere-static scenes. *In Proceedings of ICCV'01*, vol. 1, pp. 662-670, 2001.
6. G. F. Franklin, J.D. Powell, A. Emami-Naeini. Feedback Control of Dynamic Systems(4th Edition). Prentice Hall, pp. 201-252, 706-797, 2002.
7. B. C. Kuo. Automatic Control Systems(6th Edition). Prentice Hall. pp. 5-15, 1991.
8. V. Kwatra, A. Schödl, I. Essa, G. Turk and A. Bobick. Graphcut Textures: Image and Video Synthesis Using Graph Cuts. *In Proceedings of Siggraph'03*, pp. 277-286, 2003.
9. L. Ljung. System Identification – Theory for the User(2nd Edition). Prentice Hall, 1999.
10. P. Saisan, G. Doretto, Y. N. Wu and S. Soatto. Dynamic Texture Recognition. *Proceedings of CVPR'01*, vol. 2, pp. 58-63, 2001.
11. A. Schödl, R. Szeliski, D. H. Salesin and I. Essa. Video Textures. *In Proceedings of Siggraph'00*, pp. 489-498, 2000.
12. S. Soatto, G. Doretto and Y. N. Wu. Dynamic textures. *In Proceedings of ICCV'01*, vol. 2, pp. 439-446, 2001.
13. M. Szummer and R. W. Picard. Temporal Texture Modeling. *IEEE International Conference on Image Processing*, vol. 3, pp. 823-826, 1996.
14. Y. Z. Wang and S. C. Zhu. A Generative Method for Textured Motion: Analysis and Synthesis. *In Proceedings of ECCV'02*, vol. 1, pp. 583-597, 2002.
15. L. Y. Wei and M. Levoy. Fast Texture Synthesis using Tree-structured Vector Quantization. *In Proceedings of Siggraph'00*, pp. 479-488, 2000.
16. W. W. S. Wei. Time Series Analysis : Univariate and Multivariate Methods. Addison-Wesley, New York. 1990.



(a) FOUNTAIN Sequence



(b) FIRE Sequence

Fig. 6. Comparison of pole placements between open-loop LDS and closed-loop LDS: The distance from each pole to the origin is plotted for FOUNTAIN and FIRE sequences. The improvement in pole placement is zoomed in. Some stable poles of FOUNTAIN and all unstable poles of FIRE are relocated onto the unit circle, from which the distance to the origin is 1. With CLDS, poles are moved towards the unit circle, making the system more oscillatory.

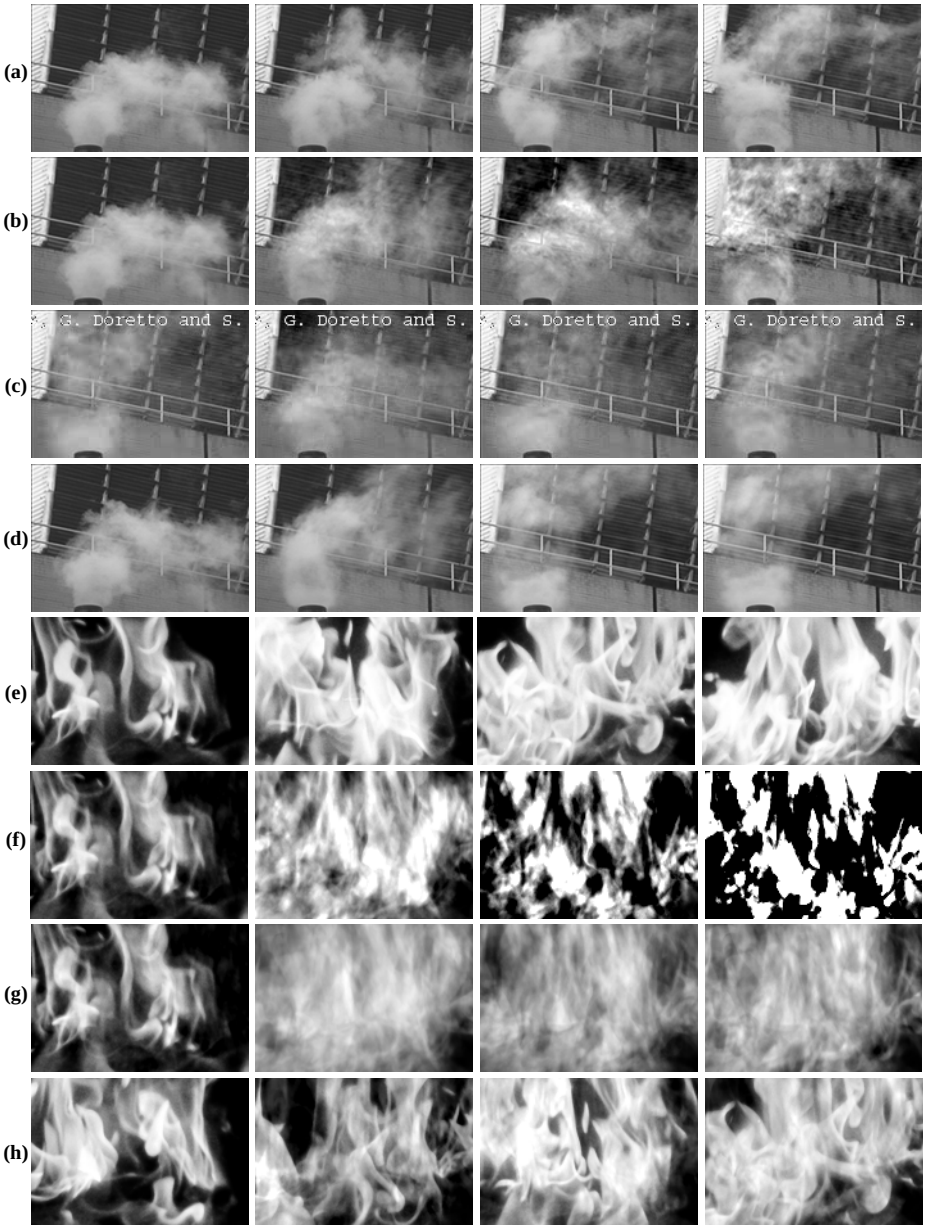


Fig. 7. Comparison of results between previous LDS methods and ours. (a) Original SMOKE-FAR sequence (100 frames). (b)-(d) Synthesized SMOKE-FAR sequence (300 frames) respectively by the basic noise-driven LDS, Doretto's method (borrowed from Doretto's web site) and our algorithm. (e) Original FIRE sequence (70 frames). (f)-(h) Synthesized FIRE sequence (300 frames) respectively by the basic noise-driven LDS. , Doretto's method (with our implementation) and our algorithm.

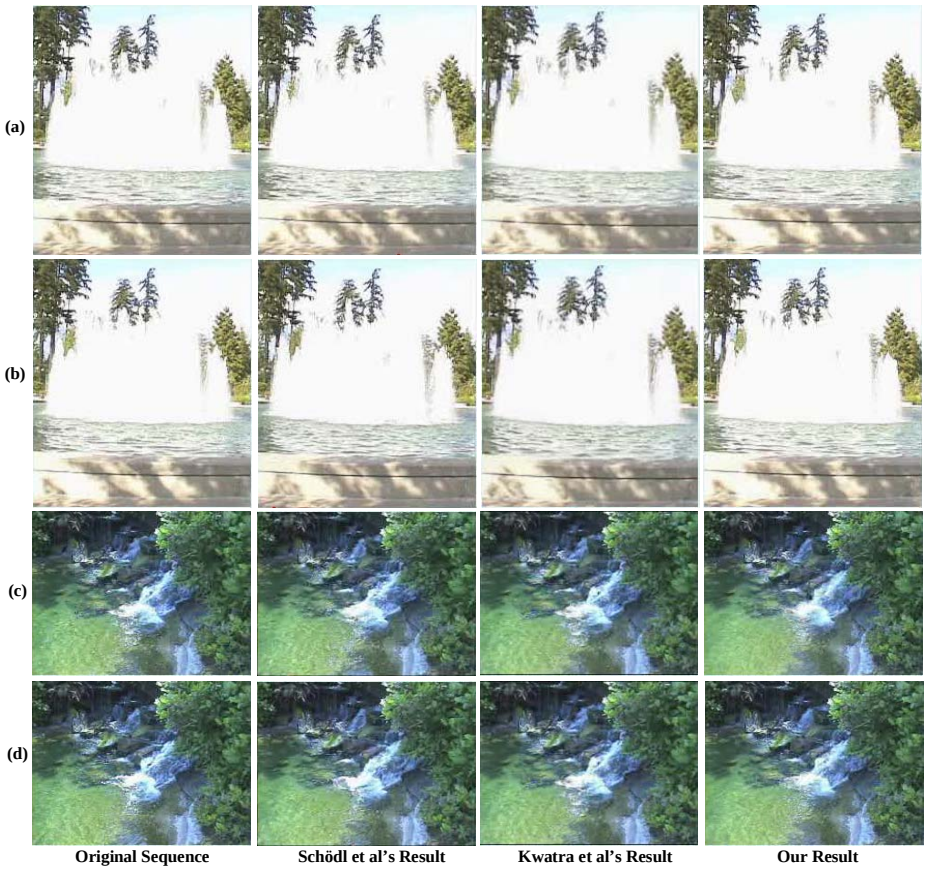


Fig. 8. Comparison of results between classic nonparametric methods and ours. From the left column to the right column, it respectively shows the original sequence, the synthesized sequence by Schödl et al's method[11], by Kwatra et al's method[8] and by our method. (a)-(b) are the 93th, 148th frame of Bldg9-FOUNTAIN. (c)-(d) are the 40th, 2085th frame of WATERFALL(the 40th, 87th frame for original video.