

Non linear Temporal Textures Synthesis: a Monte Carlo approach^{*}

Andrea Masiero and Alessandro Chiuso

Department of Information Engineering
University of Padova
Via Gradenigo, 6/b, 35100 Padova, Italy
{andrea.masiero, chiuso}@dei.unipd.it

Abstract. In this paper we consider the problem of temporal texture modeling and synthesis. A temporal texture (or dynamic texture) is seen as the output of a dynamical system driven by white noise. Experimental evidence shows that linear models such as those introduced in earlier work are sometimes inadequate to fully describe the time evolution of the dynamic scene. Extending upon recent work which is available in the literature, we tackle the synthesis using non-linear dynamical models. The non-linear model is never given explicitly but rather we describe a methodology to generate samples from the model. The method requires estimating the “state” distribution and a linear dynamical model from the original clip which are then used respectively as target distribution and proposal mechanism in a rejection sampling step. We also report extensive experimental results comparing the proposed approach with the results obtained using linear models (*Doretto et al.*) and the “closed-loop” approach presented at ECCV 2004 by *Yuan et al.*

1 Introduction

Modeling of complex scenes such as texture has been subject of intensive research in the past years. Models of physical phenomena are widely used in a number of field such as Control, Econometrics, Bioengineering and so on; in computer vision several tasks - such as synthesis, recognition, classification, segmentation - connected to video sequences are facilitated when a model is available (see for instance [3] for a specific example related to textures).

Statistical-based models appeared soon to be the useful tool to tackle the problem; this line of work was pioneered by Julesz (see [7]). After that, much work has been done (see for instance [22, 19, 21, 6] just to cite a few references) which addresses the problem of modeling and synthesis of (static) textured images.

In this work we are interested instead in the “temporal” evolution of textured images which we call temporal or dynamic textures. “Temporal” textures

^{*} This work has been supported in part by RECSYS project of the European Community and by the national project *New methods and algorithms for identification and adaptive control of technological systems* funded by MIUR.

have been first studied in [15] while the terminology “dynamic textures” was introduced in [2]; there a “dynamic textures” was defined to be a sequence of (textured) images $\{\mathbf{I}(t), t = 1, \dots, T\}$, indexed by time t , which can be modeled as the output of a (time-invariant) dynamical system of the form

$$\begin{cases} \mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{w}_t) \\ \mathbf{y}_t = h(\mathbf{x}_t, \mathbf{v}_t) \end{cases} \quad (1)$$

where $\mathbf{y}_t = \text{vec}\{\mathbf{I}(t)\}$, and $\mathbf{w}_t, \mathbf{v}_t$ are zero mean independent white noise processes. The vector \mathbf{x}_t is the “state” process and has dimension n as small as possible.

To be precise, in [2] only identification of linear models have been discussed ($f(\mathbf{x}_t, \mathbf{w}_t) = A\mathbf{x}_t + B\mathbf{w}_t$) and the corresponding output $I(t)$ was called a *linear* dynamic texture.

Similar (model based) approaches can also be found in [5, 15]; interesting results concerning temporal textures have been introduced [17, 18, 8]; recent developments have also been described in [14].

An extension of the methodology found in [2] which proposes to consider “closed-loop” dynamical systems has been recently presented in [20].

In this paper we shall concentrate on models of the form (1). In particular we shall see that linear-gaussian dynamical systems used in [2, 5] are often inadequate to represent and synthesize real-world sequences; this suggests that more sophisticated models are needed.

To do so we shall assume that the output map $h(\cdot, \cdot)$ is linear, i.e. $\mathbf{y}_t = C\mathbf{x}_t + \mathbf{v}_t$, as was done in [2]; under this assumption the state can be directly constructed using standard Principal Component Analysis. This is done in Section 2. Under our assumption that the model is stationary, we can assume that there exists an invariant density $\pi(x)$ for the state vector \mathbf{x} . Using a standard Kernel density estimator $\hat{\pi}(x)$ we verify that $\pi(x)$ departs sharply from a Gaussian density. See Section 3. The estimated density $\hat{\pi}(x)$ provides, at the same time, (i) the motivation for our study and (ii) the basic object which allows to construct a non-linear texture model.

We shall provide experimental evidence showing that the new method yields increased image quality; the estimated invariant density $\hat{\pi}(\mathbf{x})$ shall also provide a tool to measure “quality” of texture sequences synthesized using different procedures. See figures 5, 6 and the related comments in the paper.

Notation

The following notation shall be used throughout: boldface letters \mathbf{x} shall denote random vectors. A discrete time random process, i.e. an indexed collection of random vectors, shall be denoted as $\{\mathbf{x}_t\}$, $t \in \mathbb{Z}$ where t is the time index. Lowercase letters shall denote the sample value of the corresponding random vector (e.g. x_t is the sample value of \mathbf{x}_t). Capital letters shall denote matrices; X^\top denote the transpose of the matrix X , $X(i, j)$ is the element in position i, j of the matrix X . We shall use Matlab notation for row or column selections: e.g.

$X(i : j, :)$ will be the matrix formed by the rows of X with indexes from i to j . $\|X\|_F$ is the Frobenius norm of X , i.e. $\|X\|_F = \sqrt{\sum_{i,j} X(i,j)^2}$

2 Stationary Dynamical Model

Consider a sequence of images $\{I(t)\}$, $t = 1, \dots, T$ (the “data”); each image is represented as an $r \times c$ matrix of positive real numbers denoting (gray-scale) pixel intensity values. A straightforward extension can be done to deal with color images.

We shall think that the images $\{I(t)\}$ are samples from a stochastic process $\{\mathbf{I}(t)\}$; without loss of generality we shall assume that $\{\mathbf{I}(t)\}$ is zero mean; this can always be obtained by simple data pre-processing. Defining the vectorized images $\mathbf{y}_t = \text{vec}(\mathbf{I}(t))$, we shall consider time invariant dynamical models of the form (1) where the output equation is linear, i.e.:

$$\begin{cases} \mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{w}_t) \\ \mathbf{y}_t = C\mathbf{x}_t + \mathbf{v}_t \end{cases} \quad (2)$$

The state-update equation $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{w}_t)$ can also be modeled through a transition kernel $p(x_{t+1}|x_t)$ which models \mathbf{x}_{t+1} conditionally on \mathbf{x}_t .

The dimension p of \mathbf{y}_t is equal to the product $r \cdot c$. Since the output dimension is typically much larger than the number of data (images) available, i.e. $p = r \cdot c \gg T$, it is necessary to perform a first dimensionality reduction step.

This step, which is borrowed from [2], is based on Principal Component Analysis and can be formalized as follows. Let us define the data matrix $Y \triangleq [y_1 \ y_2 \ \dots \ y_T]$ where recall that $y_i := \text{vec}\{I(i)\}$.

Using the Singular Value Decomposition (SVD hereafter) algorithm we can rewrite Y as the product of three matrices $Y = USV^\top$ where U and V are matrices with orthonormal columns, i.e. $U^\top U = I_T$, $V^\top V = I_T$ while S is diagonal $S = \text{diag}(\sigma_1, \dots, \sigma_T)$, $\sigma_i \geq \sigma_j$ for $i \geq j$.

Retaining the first n columns $U(:, 1 : n)$, $V(:, 1 : n)$ of the matrices U and V respectively and the upper left block $S(1 : n, 1 : n)$ of the matrix S provides the best rank n approximation¹ of the matrix Y ,

$$Y \simeq U(:, 1 : n)S(1 : n, 1 : n)V(:, 1 : n)^\top;$$

in fact, as Y_n ranges in the set of rank- n $p \times T$, $\|Y - Y_n\|_F$ is minimized by letting $Y_n = U(:, 1 : n)S(1 : n, 1 : n)V(:, 1 : n)^\top$.

Now we define

$$X \triangleq S(1 : n, 1 : n)V(:, 1 : n)^\top, \quad \hat{C} \triangleq U(:, 1 : n), \quad x_t \triangleq X(:, t), \quad 1 < t < T$$

$$E \triangleq Y - \hat{C}X, \quad v_t \triangleq y_t - \hat{y}_t, \quad \hat{y}_t \triangleq \hat{C}x_t \quad 1 < t < T$$

Observe that, being U unitary, $x_t = \hat{C}^\top y_t$, $1 < t < T$.

The matrix $\hat{C} = U(:, 1 : n)$ containing the first n “singular vectors” can be seen as an estimate of the matrix C in (2); it provides an approximate basis

¹ In the Frobenius norm.

for the image space in the sense that $\hat{y}_t = \hat{C} \cdot (\hat{C}^\top y_t)$ is an approximation of the image y_t which is expressed as a linear function of the columns of \hat{C} . This basis is “data driven” in the sense that it is tailored on the data itself. This is extremely useful and results in a rather compact representation of the data, in the sense that good approximations \hat{y}_t can be obtained with relatively small n .

Of course one may rise several critics to this approach; for instance the SVD does not preserve positivity of the entries of each columns of \hat{C} (i.e. the columns of \hat{C} are not themselves images). Furthermore the technique is linear in nature. Several extensions are possible, starting from dimensionality reduction techniques which preserve positivity based on Kullback-Leibler [9, 4] pseudo distance (or I-divergence), to non-linear versions of SVD such as Kernel CCA [12, 10]. In this paper we shall not discuss these issues which would take us far from our goal.

Instead, we now go back to model (2), and study its statistical properties. We shall assume that the time invariant model (2) admits a unique invariant density $\pi(x)$, i.e. that,

1. the state evolution preserves the density $\pi(x)$
2. for any initial state density $\pi_0(x)$, the state density $\pi_t(x)$ (at time t) of the system (2) initialized at $\pi_0(x)$ converges to $\pi(x)$ as t grows.

We shall not discuss further this assumption nor enter into the mathematical details which are out of the scope of this conference.

Having access to samples $\hat{x}_t = \hat{C}^\top y_t$ of the state process \mathbf{x}_t one may try to estimate the density $\pi(x)$. Our purpose is twofold:

1. show that a linear model is, most of the times, inadequate
2. provide an algorithm to generate samples from a dynamical system of the form (2) which admits $\hat{\pi}(x)$ as invariant density.

3 Non-Parametric Density Estimator

Our purpose is to find an estimator for the state invariant density $\pi(x)$. The state \mathbf{x}_t lives in an n -dimensional space, where n ranges in the interval [20, 50] in typical examples. Therefore we are after estimating a probability density in a high dimensional space, which is known to be a rather difficult problem; see [11]. In our setup the problem is made even more challenging by the fact that usually a rather limited number T of samples is available. Typically, even for long video sequences, T is of the order of 500.

We consider density estimators of the Kernel type:

$$\hat{\pi}(x) = \frac{1}{T k^n} \sum_{t=1}^T \mathcal{K}(x - \hat{x}_t, \Sigma_k) \quad (3)$$

where $\hat{x}_t = \hat{C}^\top y_t$ is the estimated state at time t . \mathcal{K} is a suitable kernel parametrized by the matrix Σ_k and the positive number k which may be seen as a “regularization” parameter which controls the “width” of the Kernel \mathcal{K} . A

typical choice is the Gaussian kernel, where $\mathcal{K}(\cdot, \Sigma_k)$ is a gaussian density with zero mean and covariance matrix $\Sigma_k = k^2 I_n$.

For a complete discussion on kernel estimation, for the choice of Σ_k and of the kernel \mathcal{K} see [13] and [11].

In order to help the reader to gain some intuition we report in figure 1 an example of Kernel density estimators for a scalar Gaussian random variable using respectively $T = 40, 80, 120$ independent samples.

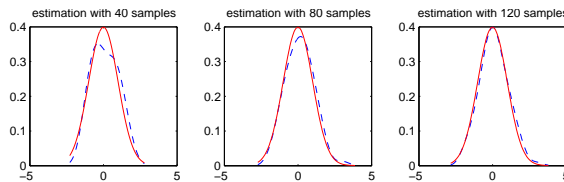


Fig. 1. Kernel estimation of a zero mean, unit variance Gaussian density with 40, 80, 120 samples. Solid: Gaussian density. Dotted: Kernel estimate.

For the sake of illustration we have computed the joint probability density of the first two state components, which are also those corresponding to higher “energy”. In figure 2 we show as a gray level image the value of the estimated state density corresponding to several texture sequences (bash, flame river, smoke, steam, waterfall) available in public databases².

It should be clear from the plots that the joint density of the first two components is rather far from being Gaussian. Therefore, a linear gaussian model such that used in [2], represent only a rough approximation of the data.

A first modification has been proposed in [1] where a linear model forced by non-gaussian noises was used. The noise densities were estimated using a version of ICA. The results presented in [1] were encouraging; several artifacts obtained by the linear Gaussian model were eliminated using the linear model with a non-gaussian noise.

In this paper we take a different approach. Instead than modifying the noise density, we try to construct a transition Kernel $p(x_{t+1}|x_t)$ which, at the same time:

1. captures the dynamic behavior and
2. admits $\hat{\pi}(x)$ as an invariant density

Static Texture Synthesis

In order to demonstrate the effectiveness of our approach we first show how the invariant density estimate $\hat{\pi}(x)$ can be utilized to generate new textured images.

² See <http://www.cc.gatech.edu/cpl/projects/graphcuttextures/> and <http://vismod.media.mit.edu/pub/szumner/temporal-texture/>

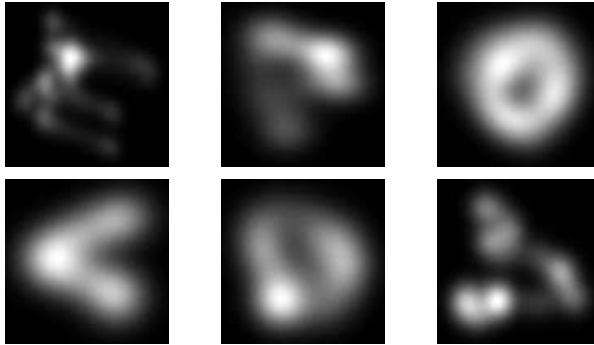


Fig. 2. Estimated joint density of the first and second component of the state in the temporal texture sequences (from left to right, top to bottom: beach, flame, river, smoke, steam, waterfall). A white zone is a high probability zone and viceversa.

The invariant density $\pi(x)$ induces through the map $\mathbf{y}_t = C\mathbf{x}_t$ a density $p(y)$ in the image space. In this sense $\hat{x}_t = \hat{C}^\top y_t$ can be regarded as image features extracted through the “filters” \hat{C}^\top and $\pi(x)$ is a density in the feature space.

Generating a “new” image $y = \text{vec}(I)$ can be done by: (i) sampling a “feature vector” x from $\pi(x)$ and (ii) generating the image according to $y = Cx$. Please note that we can only sample from $\hat{\pi}(x)$, the kernel estimate of $\pi(x)$; given our choice of a gaussian Kernel, $\hat{\pi}(x)$ turns out to be a mixture (with equal weights) of gaussian densities; this makes particularly easy the sampling step which simply consists in: (i) choose at random an index $j \in [1, T]$, (ii) sample from a gaussian with mean x_i and covariance matrix Σ_k .

We report in figure 3 images sampled according to this strategy compared with images from the original sequence. The three images from the original sequence are chosen somewhat arbitrarily and are only intended to visually represent three “typical” images of the sequence.

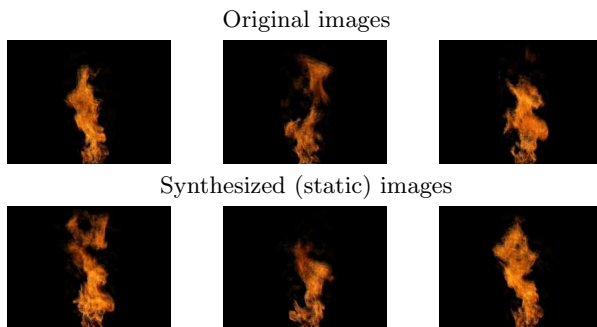


Fig. 3. Three examples of original images vs. (static) synthesized images in a flame sequence.

4 Temporal Texture Synthesis

In the previous Section we have seen that the estimated invariant density can be used as a tool to generate new static texture images with the same spacial statistics of the original texture. However we are interested in generating sequences which preserve both the spacial and the temporal statistics.

Of course there are infinitely many dynamical models of the form (2) which admits $\hat{\pi}(x)$ as invariant density.

As an example consider the simple linear dynamical system $\mathbf{x}_{t+1} = a\mathbf{x}_t + b\mathbf{w}_t$ where \mathbf{w}_t is zero mean, unit variance gaussian white noise. Assuming $|a| < 1$ there exists a unique invariant density, which is gaussian with zero mean and variance $\sigma_x^2 = b^2/(1 - a^2)$. Fixing the invariant density amounts therefore to fixing σ_x^2 . For any given σ_x^2 one can choose any arbitrary a_0 so that $\|a_0\| < 1$ which, together with $b = \sqrt{(1 - a_0^2)\sigma_x^2}$ gives the desired invariant density.

The same happens in our situation; we are given (estimated) the invariant density and want to choose a transition Kernel $p(x_{t+1}|x_t)$ which: (i) has $\hat{\pi}(x)$ as invariant density and (ii) describes well the temporal behavior of the data sequence.

This appear to be a difficult task for general non linear dynamical models $f(\mathbf{x}_t, \mathbf{w}_t)$.

Since an “approximate model” is already available from “linear dynamic textures” of [2], we propose to consider the linear model

$$\begin{cases} \mathbf{x}_{t+1} = A\mathbf{x}_t + \mathbf{w}_t \\ \mathbf{y}_t = C\mathbf{x}_t + \mathbf{v}_t \end{cases} \quad (4)$$

as an “initial estimate” of the more general (nonlinear) model (2) which has to be suitable refined in order to match the estimated invariant density $\hat{\pi}(x)$.

We shall not enter into the details of how A and the noise covariance $\Sigma_w = \text{Var}(\mathbf{w}_t)$ can be estimated. We refer the reader to [2] for the details and assume that estimates \hat{A} , $\hat{\Sigma}_w$ have been computed.

The basic idea (very simple to be honest) is to use a *rejection sampling* technique (see [16]) to construct a dynamical model which admits $\hat{\pi}(x)$ as invariant density.

Let us denote with $p(x; m, \Sigma)$ the density of a gaussian random vector with mean m and variance Σ and define

$$Q(x) := \int \hat{\pi}(z)p(x; \hat{A}z, \hat{\Sigma}_w) dz$$

Note that $Q(x)$ would be the density of the state at time $t + 1$ of the linear model (4) if the state density at time t was $\hat{\pi}(x)$. Note that, if $\hat{\pi}(z)$ was the invariant density associated to (4) then $Q(x)$ would be equal to $\hat{\pi}(x)$. As we have seen however this is not true; our scope is to modify the transition kernel $p(x_{t+1}; \hat{A}x_t, \hat{\Sigma}_w)$ so that $\hat{\pi}(x)$ is in fact invariant for the new kernel.

The rejection sampling technique requires that one chooses a constant c so that

$$\frac{\hat{\pi}(x)}{cQ(x)} \leq 1$$

Note that once $\pi(x)$ and $Q(x)$ are available, the constant c can be computed numerically off-line³.

Assume that, at time s ,

$$x_s \sim \hat{\pi}(x). \quad (5)$$

Repeat the following for $t \geq s$.

Procedure 1 [Texture generation]

1. Sample⁴ x_{t+1}^p from a gaussian density $p(x; \hat{A}x_t, \hat{\Sigma}_w)$, i.e. generate a new sample according to the linear dynamic texture model (4).
2. Accept the sample, i.e. set $x_{t+1} = x_{t+1}^p$ with probability

$$\frac{\hat{\pi}(x_{t+1}^p)}{cQ(x_{t+1}^p)}$$

otherwise go back to step one and repeat until a sample is accepted

It is possible to show that, provided (5) is satisfied, the samples x_t , $t \geq s$ are all distributed according to $\hat{\pi}(x)$.

Remark 1. Note that, if $\hat{\pi}(x)$ was the invariant density of the model (4), then $Q(x) = \hat{\pi}(x)$. It follows that choosing $c = 1$, $\frac{\hat{\pi}(x)}{cQ(x)} = 1$ and in step 2 above each sample x_{t+1}^p would be accepted. This means that our procedure does not modify the transition kernel if not needed.

Speeding up the procedure

The proposal density used in step 1 of Procedure 1 while encoding the dynamical properties captured by the linear dynamical model (4), may result in a very low synthesis speed due to the low acceptance probability in step 2 of Procedure 1.

Therefore in our simulation examples we have used a modified proposal density of the form

$$q(x|\hat{x}_t) \propto \frac{1 + (\alpha^\nu - 1)p(x; \hat{A}\hat{x}_t, \hat{\Sigma}_w)}{\alpha^\nu} \sum_{i=1}^T p(x; \hat{x}_t, \alpha\hat{\Sigma}_k) \quad \alpha \geq 1, \quad \nu > 1 \quad (6)$$

For large values of α (6) reduces to the linear model (4) used in step 1 of Procedure 1, while when α is “small” (6) tends to $\hat{\pi}(x)$.

Therefore for small α ’s, while the synthesis becomes faster since the proposal density is “close” to the invariant density, the dynamical information gets lost. In the extreme case of $\alpha = 1$ one is left with a sequence of independent samples from the invariant density $\pi(x)$.

Choice of a reasonable value of α trading speed and dynamical coherence of the sequence is outside of the scope of this paper and shall not be discussed here. The results presented in the paper are obtained using $\alpha = 3$ and $\nu = 5$. Of course Q in step 2 of Procedure 1 needs to be modified accordingly.

³ For reasons of space we shall not enter into the details of this computation.

⁴ The superscript p stands for “proposed”.

5 Results

We compare synthesized sequences using procedure 1 with few frames from the original sequences, see Figure 4. Unfortunately it is hard to judge the quality of the synthesized sequence from just few frames. Some movies showing the results on different texture sequences of the approach of this paper compared with linear models [2] and closed-loop LDS [20] can be obtained upon request from the authors. The visual quality of the sequence is improved with respect to linear models, but at the price of an increased computational load due to the rejection sampling step. The modified proposal described in (6) does indeed result in a faster procedures which nevertheless remains far slower than those proposed in [2, 20].

The estimated invariant density $\hat{\pi}(x)$ can also be used as a “likelihood function” to analytically measure the image quality. In figures 5 and 6 we compare the likelihood $\hat{\pi}(x_i)$, $i = 1, \dots, T$ of a sequence x_i generated using the approach of this paper with the likelihood obtained from the linear Gaussian model in [2] and the closed-loop LDS of [20]⁵.

In figures 5 and 6 we plot the state densities (first two components of the state) estimated from the original ($\hat{\pi}(x)$) and synthesized ($\hat{\pi}_s(x)$) sequences using linear models [2], the algorithm of this paper and closed-loop LDS [20]. In table 1 we also report the Kullback-Leibler pseudo-distance

$$KL(\hat{\pi} \parallel \hat{\pi}_s) = \int \hat{\pi}(x) \log \frac{\hat{\pi}(x)}{\hat{\pi}_s(x)} dx \quad (7)$$

between $\hat{\pi}$ and $\hat{\pi}_s$. Keep in mind that the densities $\hat{\pi}(x)$ and $\hat{\pi}_s(x)$ are estimated from data and also the integral in (7) had to be computed numerically. For instance the poor results in the “smoke” sequence may be attributed to the fact that the original clip is very short (about 50 frames). Note also that ideally our procedure produce samples from the density $\hat{\pi}$; as the length of the simulated clip goes to infinity, $\hat{\pi}_s$ - the estimated density from the synthesized sequence - becomes closer and closer to $\hat{\pi}$ and therefore the numbers in the third column of Table 1 should tend to zero.

6 Conclusions

In this paper we have shown that linear gaussian models are often inadequate to describe real word “dynamic texture” sequences. This conclusion stems from the fact that the estimated invariant measure $\hat{\pi}(x)$ is rather far from being gaussian. We therefore proposed to tackle dynamic texture synthesis using a class of non-linear dynamical models; the method take advantage of results on linear texture modeling which are available in the literature and borrows tools from

⁵ Of course the results depend on the particular choice of the user parameters (e.g. Kernel width for our approach, past and future horizons in closed-loop LDS etc.). We have optimized the parameters choice of each method so as to obtain the best results in the particular examples we consider.

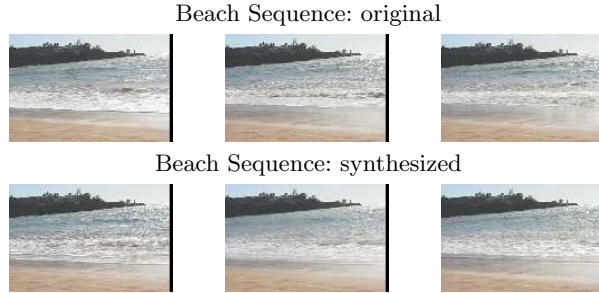


Fig. 4. Three frames from a beach sequence.

movie	linear dynamic texture	Yuan ECCV 2004	our algorithm
flame	0.44915	0.18414	0.099459
grass	0.68545	2.1566	0.62835
pond	0.17316	0.059503	0.12853
river	0.30051	0.079848	0.050529
smoke	2.5953	12.9136	1.0732
waterfall	1.6501	1.7689	0.9594

Table 1. Kullback-Leibler divergence between the state density estimated from frames of the original movie and that estimated from frames synthesized with, respectively, linear dynamic texture [2], Yuan et al. [20] and our algorithm.

multivariate statistics (Kernel density estimators) and Monte Carlo methods (Rejection Sampling). The simulation results show a considerable improvement over linear methods of [2] and a slight improvement over closed-loop LDS of [20].

Furthermore, the estimated invariant density provides a data driven measure of the image quality as a likelihood of synthesized states.

Unfortunately the proposed method suffers several drawbacks: first of all the estimation of the invariant measure $\hat{\pi}(x)$ requires, in principle, a large amount of data which are in practice hardly available. Second the computational load is considerably increased due to the rejection sampling step.

References

1. A. Bissacco, P. Saisan, and S. Soatto, “Dynamic modeling of human gaits,” in *Proc. of the IFAC Symposium on System Identification*, 2003.
2. G. Doretto, A. Chiuso, S. Soatto, and Y. Wu, “Dynamic textures,” *International Journal of Computer Vision*, vol. 51, no. 2, pp. 91–109, February 2003.
3. G. Doretto, D. Cremers, P. Favaro, and S. Soatto, “Dynamic texture segmentation,” in *Proceedings of the International Conference on Computer Vision*, vol. 2, Nice, France, October 2003, pp. 1236–1242.
4. L. Finesso and P. Spreij, “Approximate nonnegative matrix factorization via alternating minimization,” in *Proc. of the 16th International Symposium on MTNS*, Leuven, July 2004, pp. 223–252.

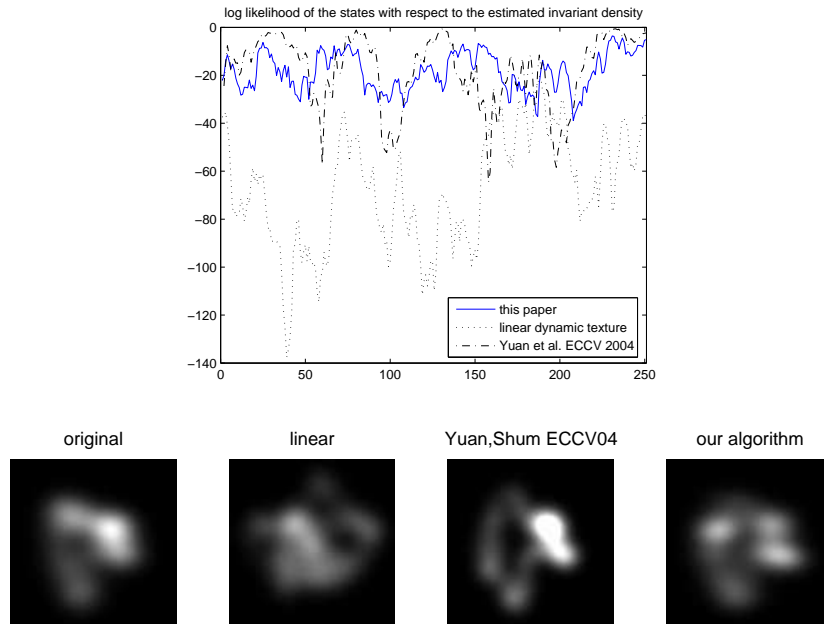


Fig. 5. Log-likelihood (top) and estimated density (bottom) of the first two components of the state (flame sequence) synthesized using the linear algorithm [2](solid line), our algorithm (dotted line) and Yuan et al. algorithm [20](dash-dotted line).

5. A. Fitzgibbon, "Stochastic rigidity: Image registration for nowhere-static scenes," in *Proc. IEEE International Conf. Computer Vision (ICCV)*, vol. 1, Vancouver, Canada, 2001, pp. 662–670.
6. D. Heeger and J. Bergen, "Pyramid-based texture analysis/synthesis," in *SIGGRAPH 95 Conf. Proc.*, 1995.
7. B. Julesz, "Visual pattern discrimination," *IRE Trans. on Information Theory*, vol. 8, pp. 84–92, 1962.
8. V. Kwatra, A. Schdl, I. Essa, G. Turk, and A. Bobick, "Graphcut textures: Image and video synthesis using graph cuts," in *Proc. ACM Transactions on Graphics (SIGGRAPH 2003)*, 2003.
9. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *In Advances in Neural and Information Processing Systems 13 (T.K. Leen, T.G. Dietterich and V. Tresp Eds.)*. MIT Press, 2001, pp. 556–562.
10. B. Scholkopf, A. Smola, and K.-R. Muller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, p. 12991319, 1998.
11. D. W. Scott, *Multivariate Density Estimation, Theory, Practice, and Visualization*. John Wiley and Sons, 1992.
12. J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.
13. B. W. Silverman, *Density estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.

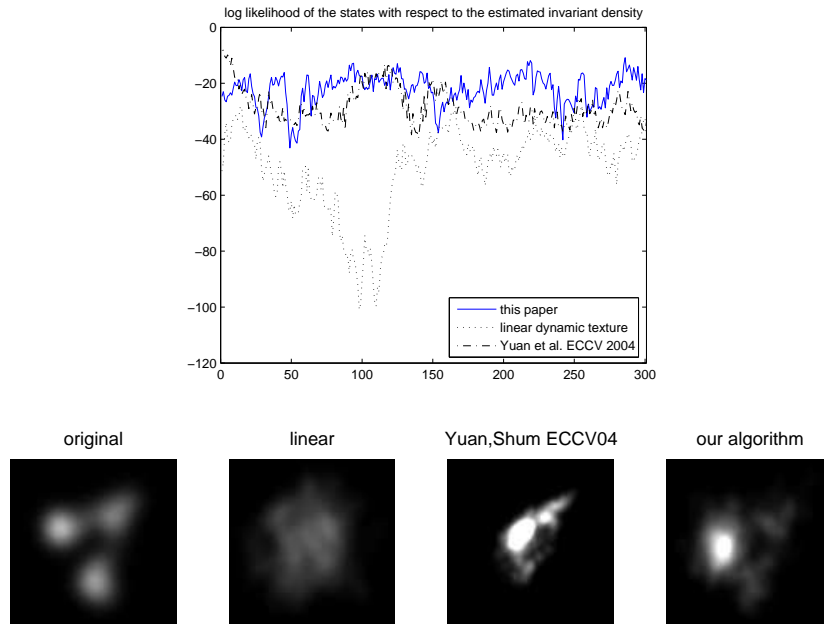


Fig. 6. Log-likelihood (top) and estimated density (bottom) of the first two components of the state (grass sequence) synthesized using the linear algorithm [2](solid line), our algorithm (dotted line) and Yuan et al. algorithm [20](dash-dotted line).

14. M. Sznajder, O. Camps, and C. Mazzaro, "Finite horizon model reduction of a class of neutrally stable systems with applications to texture synthesis and recognition," in *CDC 2004*, 2004.
15. M. Szummer and R. Picard, "Temporal texture modeling," in *IEEE International Conference on Image Processing (ICIP)*, vol. 3, Lausanne, Switzerland, 1996, pp. 823–826.
16. H. Tanizaki, *Nonlinear Filters: Estimation and Applications (Second Edition)*. Springer-Verlag, 1996.
17. Y. Wang and S. Zhu, "A generative method for textured motion: Analysis and synthesis," in *Proc. IEEE European Conf. Computer Vision (ECCV)*, 2002, pp. 583–598.
18. —, "Analysis and synthesis of textured motion: Particles and waves," *IEEE Trans. on Patt. Anal. and Mach. Intell.*, vol. 26, no. 10, pp. 1348–1363, 2004.
19. Y. Wu, S. Zhu, and X. Liu, "Equivalence of Julesz ensembles and frame models," *Int'l Journal of Computer Vision*, vol. 38, no. 3, pp. 247–267, 2000.
20. L. Yuan, F. Wen, C. Liu, and H. Shum, "Synthesizing dynamic texture with closed-loop linear dynamic system," in *Proc. of European Conf. on Computer Vision (ECCV)*, 2004, pp. 603–616.
21. S. Zhu, X. Liu, and Y. Wu, "Exploring texture ensembles by efficient markov chain monte carlo," *IEEE Trans. on Patt. Anal. and Mach. Intell.*, vol. 22, no. 6, pp. 554–569, 2000.
22. S. Zhu, Y. Wu, and D. Mumford, "Minimax entropy principle and its application to texture modeling," *Neural Computation*, vol. 9, no. 8, 1997.