Dynamic Textures

Stefano Soatto[†] Gianfranco Doretto[‡] Ying Nian Wu[§]

† Department of Computer Science
UCLA
4531E Boelter Hall
Los Angeles, CA 90095-1596
soatto@ucla.edu

‡ Department of Computer Science
UCLA
4403 Boelter Hall
Los Angeles, CA 90095-1596
doretto@ucla.edu

§ Department of Statistics
UCLA
6129 Math Sciences
Los Angeles, CA 90095-1554
ywu@stat.ucla.edu

Contact author: Stefano Soatto
phone: (310) 825-4840 fax: (310) 794-5057

Index Terms: textures, dynamic scene analysis, minimum description length, image compression, generative model, prediction error methods, ARMA regression, system identification, learning, E-M algorithm.

Abstract

Dynamic textures are sequences of images of moving scenes that exhibit certain stationarity properties in time; these include sea-waves, smoke, foliage, whirlwind etc. We present a novel characterization of dynamic textures that poses the problems of modeling, learning, recognizing and synthesizing dynamic textures on a firm analytical footing. We borrow tools from system identification to capture the "essence" of dynamic textures; we do so by learning (i.e. identifying) models that are optimal in the

sense of maximum-likelihood or minimum prediction error variance. For the special case of second-order stationary processes, we identify the model sub-optimally in closed-form. Once learned, a model has predictive power and can be used for extrapolating synthetic sequences to infinite length with negligible computational cost. We present experimental evidence that, within our framework, even low-dimensional models can capture very complex visual phenomena.

1. Introduction

Consider a sequence of images of a moving scene. Each image is an array of positive numbers that depend upon the shape, pose and motion of the scene as well as upon its material properties (reflectance) and on the light distribution of the environment. It is well known that the joint reconstruction of *photometry* and *geometry* is an intrinsically ill-posed problem: from any (finite) number of images it is not possible to uniquely recover all unknowns (shape, motion, reflectance and light distribution). Traditional approaches to scene reconstruction rely on fixing *some* of the unknowns either by virtue of assumption or by restricting the experimental conditions, while estimating the others[1].

However, such assumptions can never be validated from visual data, since it is always possible to construct scenes with different photometry and geometry that give rise to the same images[2]. The ill-posedness of the most general visual reconstruction problem and the remarkable consistency in the solution as performed by the human visual system reveals the importance of priors for images [47];

[1]For instance, in stereo and structure from motion one assumes that (most of) the scene has Lambertian reflection properties, and exploits such an assumption to establish correspondence and estimate shape. Similarly, in shape from shading one assumes constant albedo and exploits changes in irradiance to recover shape.

[2]For example, a sequence of images of the sea at sunset could have been originated by a very complex and dynamic shape (the surface of the sea) with constant reflection properties (homogeneous material, water), but also by a very simple shape (e.g. the plane of the television monitor) with a non-homogeneous radiance (the televised spatio-temporal signal). Similarly, the appearance of a moving Lambertian cube can be mimicked by a spherical mirror projecting a light distribution to match the albedo of the cube.

they are necessary to fix the arbitrary degrees of freedom and render the problem well-posed [25]. In general, one can use the extra degrees of freedom to the benefit of the application at hand: one can fix photometry and estimate geometry (e.g. in robot vision), or fix geometry and estimate photometry (e.g. in image-based rendering), or recover a combination of the two that satisfies some additional optimality criterion, for instance the minimum description length of the sequence of video data [39].

Given this arbitrariness in the reconstruction and interpretation of visual scenes, it is clear that there is no notion of a *true* interpretation, and the criterion for *correctness* is somewhat arbitrary. In the case of humans, the interpretation that leads to a correct Euclidean reconstruction (that can be verified by other sensory modalities, such as touch) has obvious appeal, but there is no way in which the correct Euclidean interpretation can be retrieved from visual signals alone.

Therefore, in this paper we will analyze sequences of images of moving scenes solely as visual signals. "Interpreting" and "understanding" a signal amounts to inferring a stochastic model that generates it. The "goodness" of the model can be measured in terms of the total likelihood of the measurements or in terms of its predicting power: a model should be able to give accurate predictions of future signals (akin to so-called prediction error methods in system identification). Such a model will involve a combination of photometry, geometry and dynamics and will be designed for maximum-likelihood or minimal prediction error variance. Notice that we will not require that the reconstructed photometry or geometry be *correct* (in the Euclidean sense), for that is intrinsically impossible without involving (visually) non-verifiable prior assumptions. All we require is that the model must be capable of predicting future measurements. In a sense, we look for an "explanation" of the image data that allows us to recreate and extrapolate it. It can therefore be thought of as the compressed version or the "essence" of the sequence of images.

1.1. Contributions of this work

This work presents several novel aspects in the field of dynamic (or time-varying) textures. On the issue of *representation*, we present a novel definition of dynamic texture that is general (even the simplest instance can capture the statistics of a sequence of images $\{I(t)\}$ that are second-order stationary process[3] with an arbitrary covariance sequence) and precise (it allows making analytical statements and drawing from the rich literature on system identification). On *learning*, we propose two criteria: total likelihood or prediction error. Under the hypothesis of second-order stationarity, we give a closed-form sub-optimal solution of the learning problem. On *recognition*, we show how textures alike tend to cluster in model space, therefore assessing the potential to build a recognition system based on this framework [40]. On *synthesis*, we show that even the simplest linear dynamical model (first-order ARMA[4] model with white zero-mean IID[5] Gaussian input) captures a wide range of dynamic textures. Our algorithm is simple to implement, efficient to learn and fast to simulate; it allows generating infinitely long sequences from short input sequences and to control the parameters in the simulation [12].

Although in our experiments we only consider simple choices of input distributions, more general classes can be taken into account by using particle filtering techniques and more general classes of filter banks. We only use linear dynamical systems because they capture second-order stationarity. Several extensions can be devised, although no closed-form solutions are available. Some of these results may be useful for video compression and for image-based rendering and synthesis of image sequences.

[3] A stochastic process is stationary (of order k) if the joint statistics (up to order k) are time-invariant. For instance a process $\{I(t)\}$ is second-order stationary if its mean $\bar{I} \doteq E[I(t)]$ is constant and its covariance $E[(I(t_1) - \bar{I})(I(t_2) - \bar{I})]$ only depends upon $t_2 - t_1$.

[4] ARMA stands for auto-regressive moving average.

[5] IID stands for independent and identically distributed.

1.2. Prior related work

Statistical inference for analyzing and understanding general images has been extensively used for the last two decades [30]. The statistical characterization of textures was pioneered by Julesz four decades back [23]. Following that, there has been extensive work in the area of 2D texture analysis, recognition and synthesis. Most of the approaches use statistical models [20, 47, 36, 37, 10, 34, 9, 19] while few others rely on deterministic structural models [14, 46]. Another distinction is that some work directly on the pixel values while others project image intensity onto a set of basis functions[6].

There have been many physically based algorithms which target specific dynamic textures [13, 16, 35]. Some simulations have been performed using particle systems [38, 43]. In these approaches a model of the scene is derived from first principles, then approximated, and finally simulated. Such techniques have been successfully applied for synthesizing sequences of natural phenomena such as smoke, fire etc. (see for instance [44, 15] and references therein), but also walking gaits ([21] and references), and mechanical systems ([4] and references). The main advantage of these techniques is the extent in which the synthesis can be manipulated, resulting in great editing power. While physically based models are the most principled and elegant, they have the disadvantage of being computationally expensive and often "highly" customized for particular textures, therefore not allowing automatic ways of inferring new models for a large class of dynamic textures.

An alternative to physically based techniques are image-based ones. In this framework, new texture movies are generated using images without building a physical model of the process that generates the scene. Among these approaches, one can distinguish two subclasses, the so-called "procedural" techniques that forego the use of a model altogether and generate synthetic images by clever concatenation

[6]Most common methods use Gabor filters [6], and steerable filters [17, 42, 20]. One could also infer and choose the best filters as part of the learning process [47].

or repetition of image data, and image-based techniques that "rely on a model," albeit not a physical one. As example of the first subclass, the work of Schödl *et al.* [41], addresses the problem by finding transition points in the original sequence where the video can be looped back in a minimally invasive way. The process involves morphing techniques to smooth out visual discontinuities. Another example is the work of Levoy and Wei [46], where they synthesize temporal textures by generating each new pixel, in the 3D spatio-temporal space of a video sequence, by searching, in the original sequence, a pixel neighborhood that best matches its companion in the synthesized output. Procedural techniques result in a relatively quick solution for the purpose of synthesis. Within this framework, the simulation is generated without explicitly inferring a model, which results in lack of flexibility for other purposes such as editing, classification, recognition, or compression.

There has been comparatively little work in the specific area of image-based techniques that rely on a model. The problem of modeling dynamic textures has been first addressed by Nelson and Polana [31], where they classify regional activities of a scene characterized by complex, non-rigid motion. The same problem has been later considered by Saisan *et al.* in [40].

Bar-Joseph [3] uses multi-resolution analysis (MRA) tree merging for the synthesis and merging of 2D textures and extends the idea to *dynamic textures*. For 2D textures new MRA trees are constructed by merging MRA trees obtained from the input; the algorithm is different from De Bonet's algorithm [10] that operates on a single texture sample. The idea is extended to dynamic textures by constructing MRA trees using a 3D wavelet transform. Impressive results were obtained for the 2D case, but only a finite length sequence is synthesized after computing the combined MRA tree. Our approach captures the essence of a dynamic texture in the form of a dynamical model, and an infinite length sequence can be generated in real time using the parameters computed off-line and, for the particular case of linear

dynamic textures, in closed-form.

Szummer and Picard's work [45] on temporal texture modeling uses a similar approach towards capturing dynamic textures. They use the spatio-temporal auto-regressive model (STAR), which imposes a neighborhood causality constraint even for the spatial domain. This severely restricts the textures that can be captured and does not allow to capture rotation, acceleration and other simple non translational motions. It works directly on the pixel intensities rather than a smaller dimensional representation of the image. We incorporate spatial correlation without imposing causal restrictions, as will be clear in the coming sections, and can capture more complex motions, including ones where the STAR model is ineffective (see [45], from which we borrow some of the data processed in Section 5).

2. Representation of dynamic textures

What is a suitable definition of texture? For a single image, one can say it is a texture if it is a realization from a stationary stochastic process with spatially invariant statistics [47]. This definition captures the intuitive notion of texture. For a sequence of images (time-varying texture), individual images are clearly not independent realizations from a stationary distribution, for there is a temporal coherence intrinsic in the process that needs to be captured. The underlying assumption, therefore, is that individual images are realizations of the output of a dynamical system driven by an independent and identically distributed (IID) process. We now make this concept precise as an operative definition of dynamic texture.

2.1. Definition of dynamic texture

Let $\{I(t)\}_{t=1...\tau}$, $I(t) \in \mathbb{R}^m$, be a sequence of τ images. Suppose that at each instant of time t we can measure a noisy version of the image, $y(t) = I(t) + w(t)$, where $w(t) \in \mathbb{R}^m$ is an independent

and identically distributed sequence drawn from a known distribution[7], $p_w(\cdot)$, resulting in a positive measured sequence $y(t) \in \mathbb{R}^m,\ t = 1 \dots \tau$. We say that *the sequence $\{I(t)\}$ is a (linear) dynamic texture* if there exists a set of n spatial filters $\phi_\alpha : \mathbb{R} \to \mathbb{R}^m$, $\alpha = 1 \dots n$ and a stationary distribution $q(\cdot)$ such that, defining $x(t) \in \mathbb{R}^n$ such that $I(t) = \phi(x(t))$ (where $\phi(\cdot)$ indicates the combination of the output of the n filters $\{\phi_\alpha\}$ respectively applied to each of the n state components; see Section 2.2 for details) we have $x(t) = \sum_{i=1}^{k} A_i x(t-i) + Bv(t)$, with $v(t) \in \mathbb{R}^{n_v}$ an IID realization[8] from the density $q(\cdot)$, for some choice of matrices, $A_i \in \mathbb{R}^{n \times n}$, $i = 1, \dots, k$, $B \in \mathbb{R}^{n \times n_v}$ and initial condition $x(0) = x_0$. Without loss of generality, we can assume $k = 1$ since we can augment the state of the above model to be $\bar{x}(t) \doteq [x(t)^T\ x(t-1)^T \dots x(t-k)^T]^T$. Therefore, a linear dynamic texture is associated to an auto-regressive moving average process (ARMA) with unknown input distribution

$$\begin{cases} x(t+1) = Ax(t) + Bv(t) \\ y(t) = \phi(x(t)) + w(t) \end{cases} \tag{1}$$

with $x(0) = x_0$, $v(t) \overset{IID}{\sim} q(\cdot)$ unknown, $w(t) \overset{IID}{\sim} p_w(\cdot)$ given, such that $I(t) = \phi(x(t))$. To the best of our knowledge, the characterization of a dynamic texture as the output of an ARMA model is novel.

We want to make it clear that this definition explains what we mean by dynamic textures. It could be argued that this definition does not capture the intuitive notion of a dynamic texture, and that is indeed possible. As showed in Section 5, however, we have found that the model (1) captures everything that our

[7] This distribution can be inferred from the physics of the imaging device. For CCD sensors, for instance, a good approximation is a Poisson distribution with intensity related to the average photon count. The reason for including the term $w(t)$ in modeling dynamic textures is because it is necessary for the covariance sequence of $y(t)$ to be arbitrary. Moreover, standard results in stochastic realization [28] state that, under some additional hypotheses (see Section 4), second-order stationary processes can be represented as the output of models of the type (1), which include the output noise term $w(t)$.

[8] In our experiments we have $m = 320 \times 220$ for the color sequences, and $m = 170 \times 110$ for the graylevel sequences, while n ranges from 10 to 50, and n_v ranges from 10 to n.

intuition calls dynamic textures, and decently capture also visual dynamical phenomena that are beyond the purpose of this modeling framework. Furthermore, one can easily generalize the definition to an arbitrary non-linear model of the form $x(t+1) = f(x(t), v(t))$, leading to the concept of a *non-linear dynamic texture*.

2.2. Filters and dimensionality reduction

The definition of dynamic texture above entails a choice of filters $\phi_\alpha,\ \alpha = 1 \dots n$. These filters are also inferred as part of the learning process for a given dynamic texture.

There are several criteria for choosing a suitable class of filters, ranging from biological motivations to computational efficiency. In the simplest case, we can take ϕ to be the identity, and therefore look at the dynamics of individual pixels[9] $x(t) = I(t)$ in (1). We view the choice of filters as a dimensionality reduction step, and seek for a decomposition of the image in the simple (linear) form

$$I(t) = \sum_{i=1}^{n} x_i(t)\theta_i \doteq Cx(t) \ , \tag{2}$$

where $C = [\theta_1, \dots, \theta_n] \in \mathbb{R}^{m \times n}$ and $\{\theta_i\}$ can be an orthonormal basis of $\mathcal{L}^2$, a set of principal components, or a wavelet filter bank.

An alternative non-linear choice of filters can be obtained by processing the image with a filter bank, and representing it as the collection of positions of the maximal response in passband [29]. In this paper we will restrict our attention to linear filters.

[9] As an anonymous reviewer remarked, this would result in an approach similar to Video Textures [41].

3. Learning dynamic textures

Given a sequence of noisy images $\{y(t)\}_{t=1...\tau}$, learning the dynamic texture amounts to identifying the model parameters A, B, C and the distribution of the input $q(\cdot)$ in the model (1). This is a form of *system identification problem* [26], where one has to infer a dynamical model from a time series. However, in the literature of dynamical systems, it is commonly assumed that the distribution of the input is known. In the context of dynamic textures, we have the additional complication of having to infer the distribution of the input along with the dynamical model. The learning, or system identification, problem can then be posed as follows.

3.1. Maximum likelihood learning

The maximum-likelihood formulation of the dynamic texture learning problem can be posed as follows:

$$\begin{aligned}
&\text{given } y(1), \dots, y(\tau), \text{ find} \\
&\hat{A}, \hat{B}, \hat{C}, \hat{q}(\cdot) = \arg \max_{A,B,C,q} \log p(y(1), \dots, y(\tau)) \\
&\text{subject to (1) and } v(t) \overset{IID}{\sim} q.
\end{aligned}$$

The inference method depends crucially upon what type of representation we choose for q. Note that the above inference problem involves the hidden variables $x(t)$ multiplying the unknown parameter A and realizations $v(t)$ multiplying the unknown parameter B, and is therefore intrinsically non-linear even if the original state space model is linear. In general, one could use iterative techniques that alternate between estimating (sufficient statistics of) the conditional density of the state and maximizing the likelihood with respect to the unknown parameters, in a fashion similar to the expectation-maximization

(EM) algorithm [11]. In order for such iterative techniques to converge to a unique minimum, *canonical model realizations* need to be considered, corresponding to particular forms for the matrices A and B. We discuss such realizations in Section 4, where we also present a *closed-form sub-optimal solution* for a wide class of dynamic textures.

3.2. Prediction error

As an alternative to maximum-likelihood, one can consider estimating the model that results in the least prediction error, for instance in the sense of mean square. Let $\hat{x}(t+1|t) \doteq E[x(t+1)|y(1),\dots,y(t)]$ be the best one-step predictor that depends upon the unknown parameters A, B, C, q. One can then pose the problem as

$$\hat{A}, \hat{B}, \hat{C}, \hat{q} \doteq \lim_{t\to\infty} \arg\min E\|y(t+1) - C\hat{x}(t+1|t)\| \quad \text{subject to (1).} \tag{3}$$

Unfortunately, explicit forms of the one-step predictors are available only under restricted assumptions, for instance linear models driven by white Gaussian noise which we consider in Section 4. For details the reader is referred to [28].

3.3. Representation of the driving distribution

So far we have managed to defer addressing the fact that the unknown driving distribution belongs, in principle, to an infinite-dimensional space, and therefore something needs to be said about how this issue is dealt with algorithmically.

We consider three ways to approach this problem. One is to transform this into a finite-dimensional

inference problem by choosing a parametric class of densities. This is done in the next section, where we postulate that the unknown driving density belongs to a finite-dimensional parameterization of a class of exponential densities, and therefore the inference problem is reduced to a finite-dimensional optimization. The exponential class is quite rich and it includes, in particular, multi-modal as well as skewed densities, although with experiments we show that even a single Gaussian model allows achieving good results. When the dynamic texture is represented by a second-order stationary process we show that a closed-form sub-optimal solution can be obtained.

The second alternative is to represent the density q via a finite number of fair samples drawn from it; the model (1) can be used to represent the evolution of the conditional density of the state given the measurements, and the density is evolved by updating the samples so that they remain a fair realization of the conditional density as time evolves. Algorithms of this sort are called "particle filters" [27], and in particular the CONDENSATION filter [8] is the best known instance in the Computer Vision community.

The third alternative is to treat (1) as a semi-parametric statistical problem, where one of the "parameters" (q) lives in the infinite-dimensional manifold of probability densities that satisfy certain regularity conditions, endowed with a Riemannian metric (corresponding to the Fisher's information matrix), and to design gradient descent algorithms with respect to the natural connection, as it has been done in the context of independent component analysis (ICA) by Amari and Cardoso [1]. This avenue is considerably more laborious and we are therefore not considering it in this study.

4 A closed-form solution for learning second-order stationary processes

It is well known that a stationary second-order process with arbitrary covariance can be modeled as the output of a linear dynamical system driven by white, zero-mean Gaussian noise [28]. In our case, we will therefore assume that there exist a positive integer n, a process $\{x(t)\}$, $x(t) \in \mathbb{R}^n$, with initial condition $x_0 \in \mathbb{R}^n$ and symmetric positive-definite matrices $Q \in \mathbb{R}^{n \times n}$ and $R \in \mathbb{R}^{m \times m}$ such that

$$\begin{cases} x(t+1) = Ax(t) + v(t) & v(t) \sim \mathcal{N}(0, Q); \quad x(0) = x_0 \\ y(t) = Cx(t) + w(t) & w(t) \sim \mathcal{N}(0, R) \end{cases} \tag{4}$$

for some matrices $A \in \mathbb{R}^{n \times n}$ and $C \in \mathbb{R}^{m \times n}$. The problem of *system identification* consists in estimating the model parameters A, C, Q, R from the measurements $y(1), \ldots, y(\tau)$. Note that B and $v(t)$ in the model (1) are such that $BB^T = Q$, and $v(t) \sim \mathcal{N}(0, I_{n_v})$ where I_{n_v} is the identity matrix of dimensions $n_v \times n_v$.

4.1 Uniqueness and canonical model realizations

The first observation concerning the model (4) is that the choice of matrices A, C, Q is not unique, in the sense that there are infinitely many such matrices that give rise to exactly the same sample paths $y(t)$ starting from suitable initial conditions. This is immediately seen by substituting A with TAT^{-1}, C with CT^{-1} and Q with TQT^T, and choosing the initial condition Tx_0, where $T \in \mathcal{GL}(n)$ is any invertible $n \times n$ matrix. In other words, the basis of the state-space is arbitrary, and any given process has *not* a unique model, but an *equivalence class* of models $\mathcal{R} \doteq \{[A] = TAT^{-1}, [C] = CT^{-1}, [Q] =$

$TQT^T, \mid T \in \mathcal{GL}(n)\}$. In order to be able to identify a unique model of the type (4) from a sample path $y(t)$, it is therefore necessary to choose a representative of each equivalence class: such a representative is called a *canonical model realization*, in the sense that it does not depend on the choice of basis of the state space (because it has been fixed).

While there are many possible choices of canonical models (see for instance [24]), we are interested in one that is "tailored" to the data, in the sense explained below. Since we are interested in data dimensionality reduction, we will make the following assumptions about the model (4):

$$m >> n;\ \mathrm{rank}(C) = n\ , \tag{5}$$

and choose the canonical model that makes the columns of C orthonormal:

$$C^T C = I_n\ , \tag{6}$$

where I_n is the identity matrix of dimension $n \times n$. As we will see shortly, this assumption results in a unique model that is tailored to the data in the sense of defining a basis of the state space such that its covariance $P \doteq \lim_{t\to\infty} E[x(t)x^T(t)]$ is asymptotically diagonal (see equation (11)).

The problem we set out to solve can then be formulated as follows: *given* measurements of a sample path of the process: $y(1), \ldots, y(\tau);\ \tau >> n$, estimate $\hat{A}, \hat{C}, \hat{Q}, \hat{R}$, a canonical model of the process $\{y(t)\}$. Ideally, we would want the maximum-likelihood solution:

$$\hat{A}(\tau), \hat{C}(\tau), \hat{Q}(\tau), \hat{R}(\tau) = \arg \max_{A,C,Q,R} p(y(1) \ldots y(\tau)). \tag{7}$$

Asymptotically optimal solution of this problem, in the maximum-likelihood sense, do exist in the literature of system identification theory [28], and is given by the subspace identification algorithm (the so-called N4SID, for which a detailed description of the implementation can be found in [33]). The main reason why in Section 4.2 we propose a sub-optimal solution of the problem is because, given the dimensionality of our framework, the N4SID algorithm requires a memory storage far beyond the capabilities of the current state of the art workstations. The result derived in Section 4.2 is a closed-form sub-optimal solution in the sense of Frobenius that takes 30 seconds to run in a common PC when $m = 170 \times 110$ and $\tau = 140$.

Before presenting the solution of the learning problem (7), we point out an unspoken hypothesis that has been made so far in the paper, i.e. the fact the framework we propose entails the filtering in space and time to be separable, which means that we perform filtering in space and time in two separate stages. The reason for this choice is nothing else than computational simplicity of the resulting algorithm.

4.2 Closed-form solution

Let $Y_1^\tau \doteq [y(1), \dots, y(\tau)] \in \mathbb{R}^{m \times \tau}$ with $\tau > n$, and similarly for $X_1^\tau \doteq [x(1), \dots, x(\tau)] \in \mathbb{R}^{n \times \tau}$ and $W_1^\tau \doteq [w(1), \dots, w(\tau)] \in \mathbb{R}^{m \times \tau}$, and notice that

$$Y_1^\tau = CX_1^\tau + W_1^\tau; \quad C \in \mathbb{R}^{m \times n};\ C^T C = I\ , \tag{8}$$

by our assumptions (5) and (6). Now let $Y_1^\tau = U \Sigma V^T$; $U \in \mathbb{R}^{m \times n}$; $U^T U = I$; $V \in \mathbb{R}^{\tau \times n}$, $V^T V = I$ be the singular value decomposition (SVD) [18] with $\Sigma = \mathrm{diag}\{\sigma_1, \dots, \sigma_n\}$, and $\{\sigma_i\}$ be the singular values, and consider the problem of finding the best estimate of C in the sense of Frobenius:

$\hat{C}(\tau), \hat{X}(\tau) = \arg\min_{C,X_1^\tau} \|W_1^\tau\|_F$ subject to (8). It follows immediately from the fixed rank approximation property of the SVD [18] that the unique solution is given by

$$\boxed{\hat{C}(\tau) = U} \quad \boxed{\hat{X}(\tau) = \Sigma V^T} \tag{9}$$

$\hat{A}$ can be determined uniquely, again in the sense of Frobenius, by solving the following linear problem: $\hat{A}(\tau) = \arg\min_A \|X_1^\tau - AX_0^{\tau-1}\|_F$, where $X_0^{\tau-1} \doteq [x(0), \ldots, x(\tau-1)] \in \mathbb{R}^{n\times\tau}$ which is trivially done in closed-form using the state estimated from (9):

$$\boxed{\hat{A}(\tau) = \Sigma V^T D_1 V (V^T D_2 V)^{-1} \Sigma^{-1}} \tag{10}$$

where $D_1 = \begin{bmatrix} 0 & 0 \\ I_{\tau-1} & 0 \end{bmatrix}$ and $D_2 = \begin{bmatrix} I_{\tau-1} & 0 \\ 0 & 0 \end{bmatrix}$. Notice that $\hat{C}(\tau)$ is uniquely determined up to a change of sign of the components of C and x. Also note that

$$E[\hat{x}(t)\hat{x}^T(t)] \equiv \lim_{\tau\to\infty} \frac{1}{\tau} \sum_{k=1}^{\tau} \hat{x}(t+k)\hat{x}^T(t+k) = \Sigma V^T V \Sigma = \Sigma^2 \,, \tag{11}$$

which is diagonal as mentioned in Section 4.1. Finally, the sample input noise covariance Q can be estimated from

$$\boxed{\hat{Q}(\tau) = \frac{1}{\tau} \sum_{i=1}^{\tau} \hat{v}(i)\hat{v}^T(i)} \tag{12}$$

where $\hat{v}(t) \doteq \hat{x}(t+1) - \hat{A}(\tau)\hat{x}(t)$. Should $\hat{Q}$ not be full rank, its dimensionality can be further reduced by computing the SVD $\hat{Q} = U_Q \Sigma_Q U_Q^T$ where $\Sigma_Q = \mathrm{diag}\{\sigma_Q(1), \ldots, \sigma_Q(n_v)\}$ with $n_v \leq n$, and letting $\hat{B}$ be such that $\hat{B}\hat{B}^T = \hat{Q}$.

In the algorithm above we have assumed that the order of the model n was given. In practice, this needs to be inferred from the data. Following [2], we propose to determine the model order empirically from the singular values $\sigma_1, \sigma_2, \ldots$, by choosing n as the cutoff where the singular values drop below a threshold. A threshold can also be imposed on the difference between adjacent singular values.

Notice that the model we describe in this paper can also be used to perform maximum-likelihood denoising of the original sequence. It is immediate to see that the denoised sequence is given by

$$\hat{I}(t) \doteq \hat{C}\hat{x}(t) \ , \tag{13}$$

where $\hat{C}$ is the maximum-likelihood estimate of C and $\hat{x}(t)$ is obtained from $\hat{x}(t+1) = \hat{A}\hat{x}(t) + \hat{B}\hat{v}(t)$.

4.3 Asymptotic properties

The solution given above is, strictly speaking, *incorrect because the first SVD does not take into account the fact that the state $X(\tau)$ has a very particular structure* (i.e. it is the state of a linear dynamical model). It is possible, however, to adapt the algorithm to take this into account while still achieving a closed-form solution that can be proven to be asymptotically efficient, i.e. to approach the maximum-likelihood solution. The resulting algorithm is known in the literature as N4SID, and its asymptotic properties have been proven by Bauer in [5]. Such an optimal algorithm, however, is computationally expensive, and the gain in the quality of the final model, for the experiments reported below, is negligible.

```
function [x0,Ymean,Ahat,Bhat,Chat] = dytex(Y,n,nv)

% Suboptimal Learning of Dynamic Textures;
% (c) UCLA, March 2001.

  tau = size(Y,2); Ymean = mean(Y,2);
  [U,S,V] = svd(Y-Ymean*ones(1,tau),0);
  Chat=U(:,1:n); Xhat = S(1:n,1:n)*V(:,1:n)';
  x0=Xhat(:,1);
  Ahat = Xhat(:,2:tau)*pinv(Xhat(:,1:(tau-1)));
  Vhat = Xhat(:,2:tau)-Ahat*Xhat(:,1:(tau-1));
  [Uv,Sv,Vv] = svd(Vhat,0);
  Bhat  = Uv(:,1:nv)*Sv(1:nv,1:nv);

function [I] = synth(x0,Ymean,Ahat,Bhat,Chat,tau)

% Synthesis of Dynamic Textures;
% (c) UCLA, March 2001.

  [n,nv] = size(Bhat);
  X(:,1) = x0;
  for t = 1:tau,
    X(:,t+1) = Ahat*X(:,t)+Bhat*randn(k,1);
    I(:,t) = Chat*X(:,t)+Ymean;
  end;
```

Figure 1: *Matlab code implementation of the closed-form sub-optimal learning algorithm proposed in Section 4 (function* `dytex`*), and the synthesis stage (function* `synth`*). In order to perform stable simulations, the synthesis function assumes that the poles of the linear system (i.e. the eigenvalues of* `Ahat`*) are within the unit circle.*

5. Experiments

We coded the algorithm described in Section 4 using Matlab: learning a graylevel sequence of 140 frames with $m = 170 \times 110$ takes about 30 seconds on a desktop PC (1GHz), while it takes about 5 minutes for 150 color frames and $m = 320 \times 220$. Synthesis can be performed at frame rate. The Matlab routines implementing the learning and synthesis algorithms are reported in Figure 1. The dimension of the state n and input n_v is given as an input argument. In our implementation, we have used τ between 50 and 150, n between 10 and 50 and n_v between 10 and 30.

5.1. Synthesis

Figure 2 shows that an "infinite length" texture sequence can be synthesized from a typically "short" input sequence by just drawing IID samples $v(t)$ from a Gaussian distribution. The frames belong to the `spiraling-water` sequence. From a 120 frame-long training sequence a 300 frames synthesized sequence[10] of dimensions 85×65 pixels has been generated using $n = 20$ principal components. This sequence has been shown in [45] as an example where the STAR model fails in capturing non-translational motion. Our model, on the other hand, has no difficulty in capturing the spatio-temporal statistics of the input sequence, as shown in Figure 2.

Figures 3 to 7 show the behavior of the algorithm on a representative set of experiments (the training sequences of Figures 3 to 6 have been borrowed from the MIT Temporal Texture database[11]). In each case, on the first row we show a few images from the original dataset, on the second row we show their compressed version (see Section 5.3), and on the third row we show a few extrapolated samples. On the last row we show the overall compression error as a function of the dimension of the state space (left) as well as the prediction error as a function of the length of the learning set (right). For very regular sequences, the prediction error decreases monotonically; however, for highly complex scenes (e.g. `talking-face`, `smoke`), it is not monotonic. In these simulations x_0 was set to $\hat{x}(1)$, which explains why the first 100 frames of the 300 synthesized frames resemble the ones of the training sequence. Notice that the example of the talking face (Figure 7) has been included to show the output of the proposed technique when the underlying hypothesis of the input sequence being a realization of a second-order stationary process is violated. Of course the model fails to capture the non-stationary nature of the sequence, giving rise to a synthesized sequence that shows some artifacts. This is, therefore, an example

[10] Movies available online at `http://www.cs.ucla.edu/~doretto/projects/dynamic-textures.html`.
[11] `ftp://whitechapel.media.mit.edu/pub/szummer/temporal-texture/`

where our technique fails. Nevertheless, it is surprising to note that such a simple model can be pushed so far in modeling complex visual phenomena.

As explained in Section 4, we choose the model order n and learn the parameters of the model. A crucial question is how long should the input sequence be in order to capture the temporal dynamics of the process? To answer this question experimentally we plot the prediction error at the bottom right of the Figures 3 to 7 as a function of the length, τ, of the input (training) sequence. This means that for each length, τ, we predict the frame $\tau + 1$ (not part of the training set) and compute the prediction error per pixel in gray levels. We do so many times in order to infer the statistics of the prediction error, i.e. mean and variance at each τ. Figure 8 shows an error-bar plot including mean and standard deviation of the prediction error per pixel for the `steam` sequence. The average error decreases and becomes stable after approximately 80 frames. Notice that the plot of Figure 8 has also the meaning of model verification in the sense that this plot of the prediction error validates *a-posteriori* the model inferred with a maximum-likelihood sub-optimal solution, and is informative for challenging the model.

Another important parameter to compare various texture synthesis models is the time it takes to synthesize them. It is well established that models using Gibbs sampling [47] and other sampling methods to draw samples from complex distributions are computationally intensive. Moreover, there is always uncertainty on whether the samples have converged. Deterministic methods to extend and extrapolate sequences have to go back and query the input texture in one way or another to obtain information that generates the next frame [14, 46] [12]. In our model, learning is performed in closed-form (30 seconds for 100 graylevel samples), and *synthesis is instantaneous* (frame-rate), even in our Matlab implementation. Moreover, we can even control the size of parameters to obtain a particular synthesis speed, and change

[12] In [46] for each new pixel a search is conducted for a similar neighborhood pattern in the original texture.

the model parameters (e.g. the eigenvalues of $\hat{A}$) to manipulate the original dynamics [12]. Notice that, in certain cases corresponding to certain natural phenomena (e.g the sequence of `smoke-far`) it may be possible, and actually correct, for the learning process to return a marginally stable system, capturing the "explosive" nature of the input sequence. In fact, the system captures the exact nature of the ongoing process of the training set and generalizes it in time during a synthesis simulation. Analytically, the poles of the training sequences can be very close to the unit circle, and for the case of "unstable" dynamic textures, in order to make stable synthesis simulations, we just relocate the unstable system poles within the unit circle. To accomplish this task we found that by simply reducing to 0.99 the distance of the unstable poles from the origin (while maintaining their phase constant), we obtain stable synthesized sequences that very well resemble the original training set.

Finally, Figure 9 and Figure 10 show some more results on synthesis. Here, the dimension of the state has been set to $n = 50$, and x_0 has been drawn from a zero-mean Gaussian distribution with covariance inferred from the estimated state $\hat{X}(\tau)$. For the experiments in Figure 9, the training sequences has been borrowed again from the MIT Temporal Texture database, the length of these sequences ranges from $\tau = 100$ to $\tau = 150$ frames, and the synthesized sequences are 300 frames long. For the experiments in Figure 10, the training sets are color sequences that have been captured by the authors except for the `fire` sequence that comes from the Artbeats Digital Film Library[13]. The length of the sequences is $\tau = 150$ frames, the frames are 320×220 pixels, and the synthesized sequences are 300 frames long. The extension of the learning algorithm to the case of color images can be done in several ways. The one we used for our experiments implies that the column vector $y(t)$, at time t, in equation (1), contains the three unfolded RGB channels ordered one after the other. Representation of color in a more suitable

[13] `http://www.artbeats.com`

space [22] may lead to a more efficient use of our model in terms of ability to capture information from the training sequence.

5.2. Recognition

According to our definition in Section 2.1, each texture is characterized by an ARMA model. Therefore, in order to compare textures, we need to first define a base measure in the space of linear dynamical systems, and then to characterize probability distributions in that space.

Defining an appropriate base measure in the space of ARMA models is not trivial, since each model entails a combination of an input density and state and output transition matrices that have a very particular Riemannian structure (they are *not* a linear space). We should define an inner product in the space of models (which involves Stiefel manifolds), and a distance as the length of the geodesic joining two models. This is beyond the scope of this paper, and we refer the reader to [7, 40] for details on how to do so. Here we compute the Kullback-Leibler divergence between different realizations of the textures and show how similar textures cluster together in model space. The problem is formalized as follows.

Let $I(t)$, $t = 0, 1, \ldots$ be an infinitely long sequence of images. This can be modeled as a stochastic process which takes values in a subset of $\mathbb{R}^m$ for an m-dimensional image. Let $\bar{I}^\tau \doteq (I(1), I(2), \ldots, I(\tau))$ be a sequence of images and let $p(\bar{I}^\tau)$ be the corresponding probability density function (p.d.f.). The p.d.f. $p(\bar{I}^\tau)$ is completely determined by the parameters that define the model (1). Now, let p_1 and p_2 be two p.d.f.s that correspond to two different dynamic textures. The K-L divergence, $KL(p_1\|p_2)$, between p_1 and p_2 is defined as $KL(p_1\|p_2) \doteq \lim_{\tau\to\infty} KL^\tau(p_1\|p_2)$ where $KL^\tau(p_1\|p_2) = \frac{1}{\tau} E_{p_1}[\log(p_1(\bar{I}_1^\tau)/p_2(\bar{I}_1^\tau))]$ and $E_{p_1}[\cdot]$ is the expectation taken with respect to p_1.

In Figure 11 we display the distance, the quantity $KL^\tau(p_1\|p_2)$, between different dynamic textures

plotted against the length τ. We have taken different realizations of the textures `river` and `steam` and have computed the distance of the former realizations against themselves and the latter[14]. It is evident that alike textures tend to cluster together. Therefore, in principle a comprehensive database of parameters learned from commonly occurring dynamic textures can be maintained and a new temporal sequence can be categorized after learning its parameters and computing the distance. Notice that it would not be possible to build a recognition framework using procedural techniques like [41, 46]. An extensive assessment of the recognition capabilities of our system as well as extensions to non-global representations, e.g. segmentation ones, are beyond the scope of the paper. Some of these issues are discussed in [40, 7].

5.3. Compression

In this section we present a preliminary comparison between storage requirements for the estimated parameters relative to the original space requirement of the texture sequences, to get an estimate of the sequence compression capabilities of our model. A thorough assessment of the compression capabilities of this model is a research program in its own right. Our intention, here, is to point out the potential of the model for compression, as a further motivation for the model.

The storage requirement of the original dataset is $\mathcal{O}(m\tau)$, while the components of the model that are necessary to re-create an approximation of the sequence are A, C, Q and the input sequence $v(t)$. Therefore, one would need n^2 numbers (for $\hat{A}$), $m \times n - n(n-1)/2$ (for $\hat{C}$, counting the orthogonality constraints), $n \times n_v$ numbers for $\hat{Q}$ and, finally, $n_v \times \tau$ numbers for the input sequence $\hat{v}(t)$. Thus, the storage requirement of our model is $\mathcal{O}(mn + n^2 + nn_v + n_v\tau)$, where $n << m$, $\tau > n$, and n_v is the

[14]We obtain a similar plot if we compute the distance of the latter against the former although the K-L divergence by definition is not commutative.

effective rank of $\hat{Q}$. If we consider the fact that typical values for acceptable "lossy" could be $n = 30$ and $n_v = 20$, it is immediate to convince ourselves that the effectively compression power comes especially when long sequences are considered, i.e. when $\tau >> n$, since the matrix C is responsible for the higher storage occupancy while the other components are negligible (it is enough to notice that the sequence has to be 15000 frames long to have $n_v\tau = mn$, when $m = 100 \times 100$).

Of course, a more systematic evaluation of the potential of this model for compression is due. We would like to point out that our algorithm provides compression based on the *temporal* characteristics, and therefore it operates *on top* of MPEG encoding and provides further compression. For very long sequences (large τ) , the algorithm presented above can be modified in order to avoid computing the SVD of a very large matrix. In particular, the model can be identified from a shorter subsequence, and then the identified model can be used to compute the input (in innovation form [32]) using a simple linear Kalman filter. For real-time transmission or broadcasting, the innovation can be estimated in real-time using a Kalman filter and transmitted in lieu of the sequence of images, after the initial model is identified and transmitted to the receiver. This would ensure real-time coding and decoding – after an initial batch – for applications such as teleconferencing or remote video broadcasting.

6. Discussion

We have introduced a novel representation of dynamic textures and associated algorithms to perform learning and synthesis of sequences from training data. To our great surprise even the simplest choice of a first-order ARMA model driven by white zero-mean Gaussian noise can capture complex visual phenomena. The algorithm is simple to implement, efficient to learn and fast to simulate. Having a model that represents data is good in general because provides its compact representation and manipulation (via

model parameters). Therefore, the framework presented in this paper stimulates several areas of future investigation, ranging from video compression, classification, recognition to image-based rendering, synthesis, and editing of image sequences.

Acknowledgments

This research is supported in part by NSF grants IIS-9876145, IIS-9877127 and DMS-0072538. We wish to thank Prabhakar Pundir for his assistance on reviewing current literature and with data collection, and Alessandro Chiuso for discussions on subspace identification.

References

[1] S. Amari and J. Cardoso. Blind source separation - semiparametric statistical approach. *IEEE Transactions on Signal Processing*, 45:692–700, nov 1997.

[2] K.S. Arun and S.Y. Kung. Balanced approximation of stochastic systems. *SIAM Matrix Analysis and Applications*, pages 42–68, January 1990.

[3] 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*, 7(2):120–135, 2001.

[4] R Barzel. *Physically-Based Modeling for Computer Graphics: A Structured Approach*. Academic Press, Inc., 1992.

[5] D. Bauer, M. Deistler, and W. Scherrer. Consistency and asymptotic normality of some subspace algorithms for systems without observed inputs. *Automatica*, 35(7):1243–54, July 1999.

[6] J. Bigun and J. M. du Buf. N-folded symmetries by complex moments in gabor space and their application to unsupervised texture segmentation. In *IEEE transactions on Pattern Analysis and Machine Intelligence,16(1)*, pages 80–87, january 1994.

[7] A. Bissacco, A. Chiuso, Y. Ma, and S. Soatto. Recognition of human gaits. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Kauai, HA, December 2001.

[8] A. Blake and M. Isard. Active contours. *SpringerVerlag*, 1998.

[9] G. Cross and A. Jain. Markov random field texture models. In *IEEE transactions on Pattern Analysis and Machine Intelligence*, volume 5, pages 25–40, 1983.

[10] J. de Bonet and P. Viola. Multiresolution sampling procedure for analysis and synthesis of texture images. In *Proceedings IEEE Conf. On Computer Vision and Pattern Recognition*, 1998.

[11] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. R. Statist. Soc. B*, 39:185–197, 1977.

[12] G. Doretto and S. Soatto. Editable dynamic textures. Technical Report 020001, UCLA Computer Science Department, January 2002.

[13] D. Ebert, W. Carlson, and R. Parent. Solid spaces and inverse particle systems for controlling the animation of gases and fluids. *The Visual Computer,10(4)*, pages 179–190, September 1994.

[14] A. Efros and T. Leung. Texture synthesis by non-parametric sampling. In *Seventh International Conference on Computer Vision, Corfu, Greece*, 1999.

[15] N Foster and R Fedkiw. Practical animation of liquids. In *Proceedings of SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 15–22. ACM, ACM Press / ACM SIGGRAPH, 2001.

[16] A. Fournier and W. Reeves. A simple model of ocean waves. In *ACM SIGGRAPH Proceedings*, pages 75–84, 1986.

[17] W. Freeman and E. Adelson. The design and use of steerable filters. In *IEEE transactions on Pattern Analysis and Machine Intelligence, 13(9)*, pages 891–906, 1991.

[18] G. Golub and C. Van Loan. *Matrix Computations*. Jon Hopkins University Press, 2 edition, 1989.

[19] M. Hassner and J. Sklansky. The use of markov random fields as models of texture. *Image Modeling. Academic Press Inc.*, 1981.

[20] D. Heeger and J. Bergen. Pyramid-based texture analysis /synthesis. In *ACM SIGGRAPH Conference Proceedings*, August 1995.

[21] J K Hodgins and W L Wooten. Animating human athletes. In Y Shirai and S Hirose, editors, *Robotics Research: The Eighth International Symposium*, pages 356–367, Berlin, Germany, 1998. Springer-Verlag.

[22] R.W. Hunt. *The Reproduction of Colour*. Fisher Books, 5th edition, 1996.

[23] B. Julesz. Visual pattern discrimination. *IRE Trans info theory, IT-8*, 1962.

[24] T. Kailath. *Linear Systems*. Prentice Hall,Englewood Cliffs, NJ., 1980.

[25] A. Kirsch. An introduction to the mathematical theory of inverse problems. *Springer-Verlag, New York*, 1996.

[26] A. Lindquist and G. Picci. the stochastic realization problem. *SIAM J. Control Optim. 17*, pages 365–389, 1979.

[27] J. Liu, R. Chen, and T. Logvinenko. A theoretical framework for sequential importance sampling and resampling. Technical report, Stanford University, Department of Statistics, January 2000.

[28] L. Ljung. *System Identification -Theory for the User*. Prentice Hall, Englewood Cliffs, NJ, 1987.

[29] S. Mallat. A theory of multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:674–693, 1989.

[30] D. Mumford and B. Gidas. Stochastic models for generic images. In *Technical report, Division of Applied Mathematics, Brown University*, 1998.

[31] R. C. Nelson and R. Polana. Qualitative recognition of motion using temporal texture. *Computer Vision, Graphics, and Image Processing. Image Understanding*, 56(1):78–89, July 1992.

[32] P. Van Overschee and B. De Moor. Subspace algorithms for the stochastic identification problem. *Automatica*, 29:649–660, May 1993.

[33] P. Van Overschee and B. De Moor. N4sid: subspace algorithms for the identification of combined deterministic-stochastic systems. *Automatica*, 30:75–93, January 1994.

[34] R. Paget and D. Longstaff. A nonparametric multiscale markov random field model for synthesising natural textures. In *Fourth International Symposium on Signal Processing and its Applications*, volume 2, pages 744–747, August 1996.

[35] D. Peachey. Modeling waves and surf. In *SIGGRAPH Conference Proceedings*, pages 65–74, 1986.

[36] K. Popat and R. Picard. Novel cluster-based probability model for texture synthesis, classification, and compression. In *Proceedings SPIE visual Communications and Image Processing, Boston*, 1993.

[37] J. Portilla and E. Simoncelli. Texture representation and synthesis using correlation of complex wavelet coefficient magnitudes. In *CSIC,Madrid*, April 1999.

[38] W. Reeves. Particle systems: a technique for modeling a class of fuzzy objects. In *ACM Trans. Graphics*, volume 2, pages 91–108, 1983.

[39] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

[40] P. Saisan, G. Doretto, Y. N. Wu, and S. Soatto. Dynamic texture recognition. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Kauai, HA, December 2001.

[41] A. Schodl, R. Szeliski, D. Salesin, and I. Essa. Video textures. In *Proceedings of ACM SIGGRAPH Conference, New Orleans, LA*, January 2000.

[42] E. Simoncelli, W. Freeman, E. Adelson, and D. Heeger. Shiftable multi-scale transforms. In *IEEE Trans Information Theory, 38(2)*, pages 587–607, March 1992.

[43] K. Sims. Particle animation and rendering using data parallel computation. *Computer Graphics*, 24(4):405–413, 1990.

[44] J. Stam and E. Fiume. Depicting fire and other gaseous phenomena using diffusion processes. In *SIGGRAPH Conference Proceedings*, pages 129–136, 1995.

[45] M. Szummer and R. W. Picard. Temporal texture modeling. In *IEEE International Conference on Image Processing, Lausanne, Switzerland*, volume 3, Sept 1996.

[46] L. Y. Wei and M. Levoy. Fast teture synthesis using tree structured vector quantization. In *SIGGRAPH Conference Proceedings*, 2000.

[47] S. Zhu, Y. Wu, and D. Mumford. Minimax entropy principle and its application to texture modeling. *Neural Computation*, 9:1627–1660, 1997.

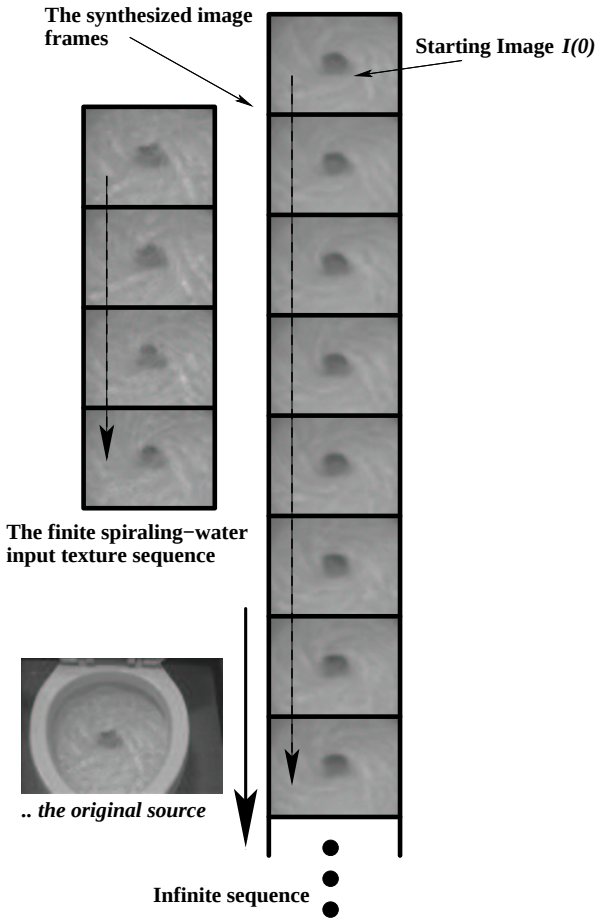

Figure 2: **Spiraling-water.** *The figure shows how an "infinite length" texture sequence is synthesized from a typically "short" input texture sequence by just drawing samples from* $v(t)$*. The data set used comes from the* MIT Temporal Texture database*. The particular structure of this sequence (*`spiraling-water` *synthesized using* $n = 20$ *principal components,* $\tau = 120$*,* $m = 85 \times 65$*), is amongst the ones that cannot be captured by the STAR model [45].*

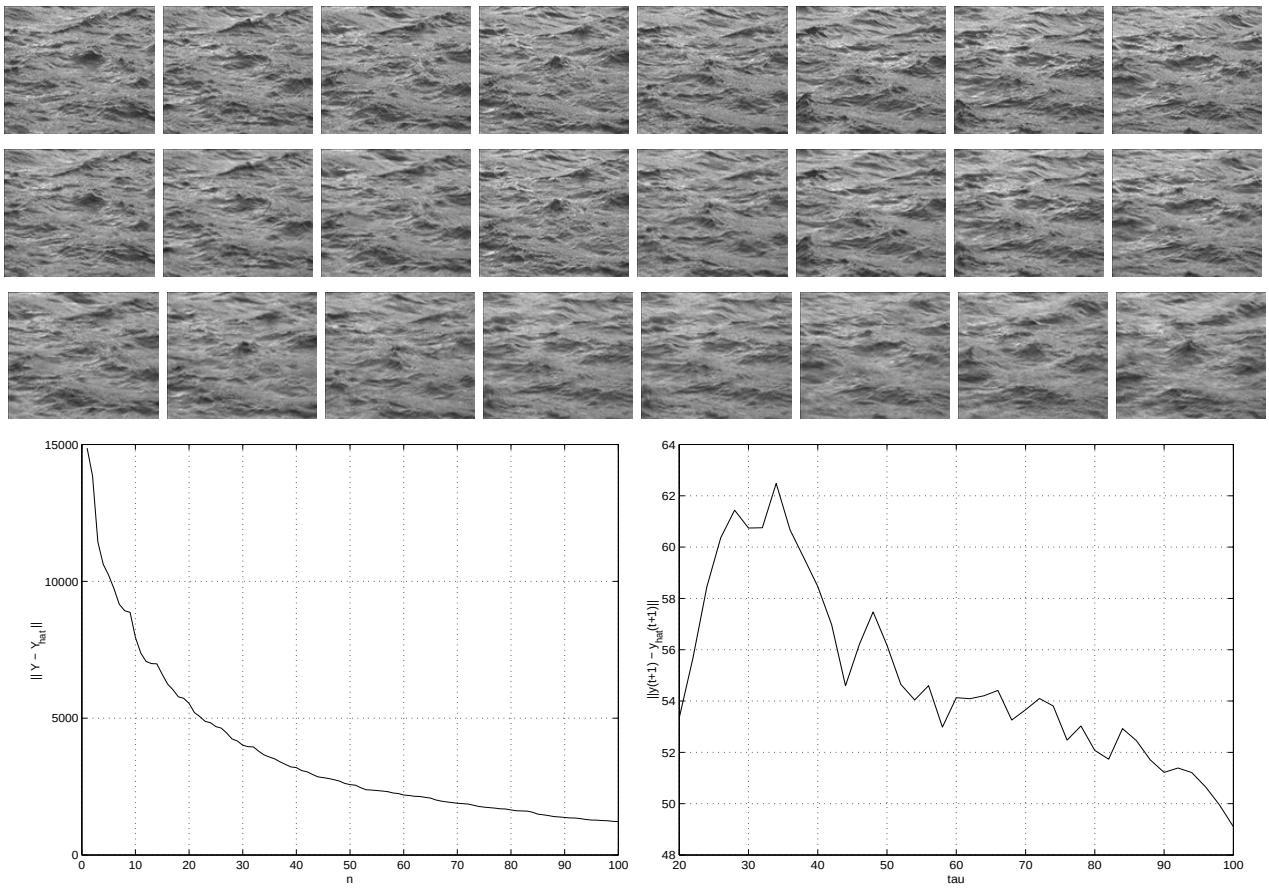

Figure 3: **River.** *From top to bottom: samples of the original sequence, corresponding samples of the compressed sequence (compression ratio:* 2.53*), samples of extrapolated sequence (using* $n = 50$ *components,* $\tau = 120$*,* $m = 170 \times 115$*), compression error as a function of the dimension of the state space* n*, and extrapolation error as a function of the length of the training set* τ*. The data set used comes from the* MIT Temporal Texture database*.*

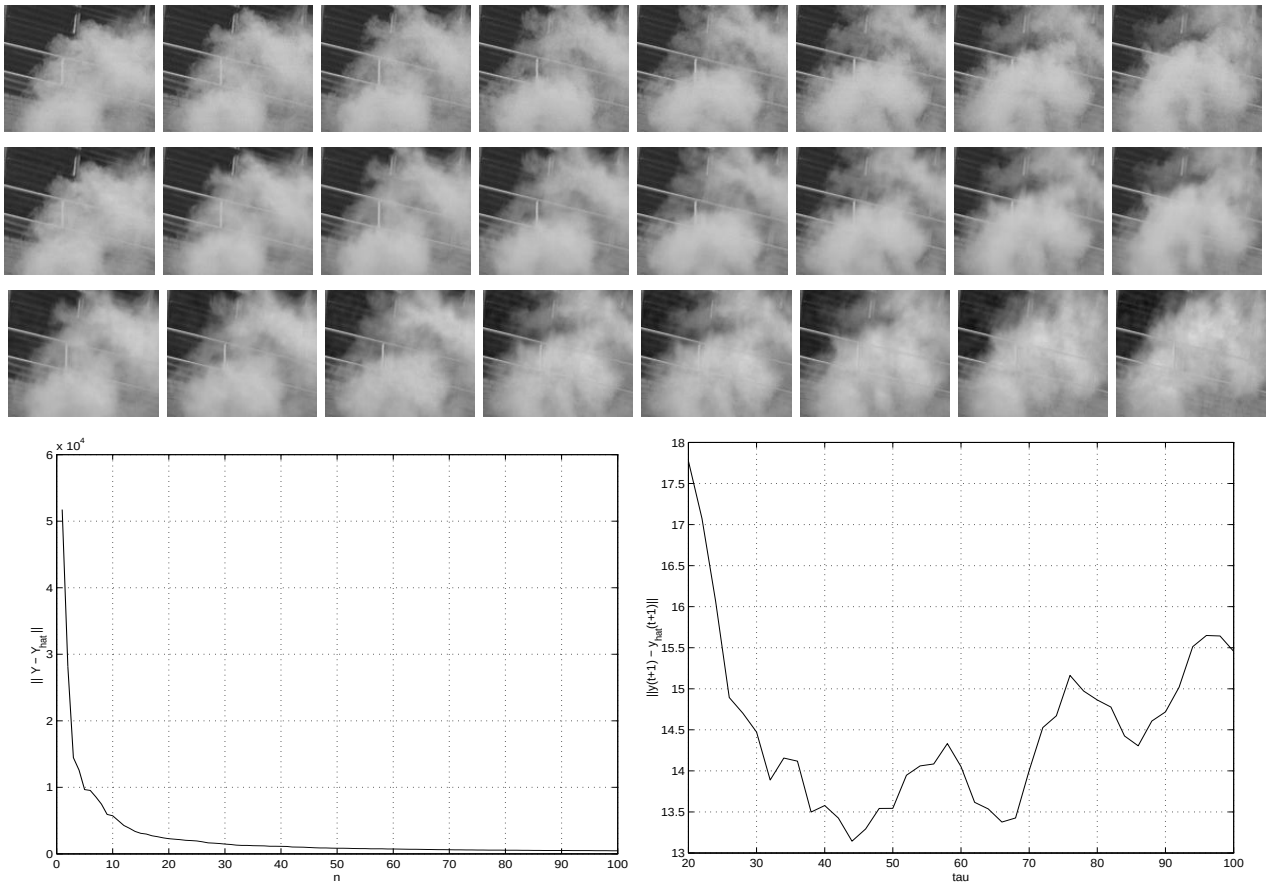

Figure 4: **Smoke.** *From top to bottom: samples of the original sequence, corresponding samples of the compressed sequence (compression ratio:* 2.53*), samples of extrapolated sequence (using* $n = 30$ *components,* $\tau = 150$*,* $m = 170 \times 120$*), compression error as a function of the dimension of the state space* n*, and extrapolation error as a function of the length of the training set* τ*. The data set used comes from the* MIT Temporal Texture database*.*

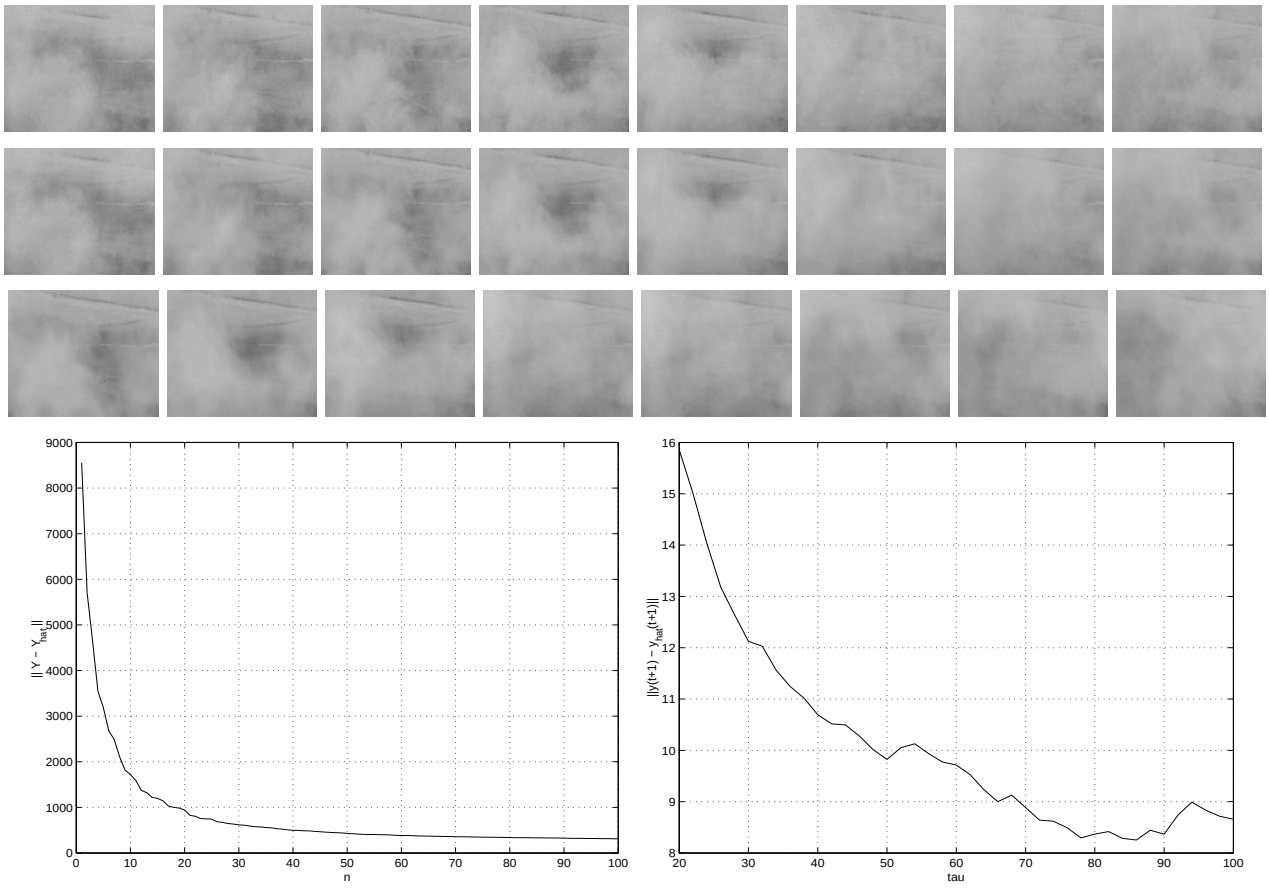

Figure 5: **Steam.** *From top to bottom: samples of the original sequence, corresponding samples of the compressed sequence (compression ratio:* 2.53*), samples of extrapolated sequence (using* $n = 30$ *components,* $\tau = 120$*,* $m = 176 \times 96$*), compression error as a function of the dimension of the state space* n*, and extrapolation error as a function of the length of the training set* τ*. The data set used comes from the* MIT Temporal Texture database*.*

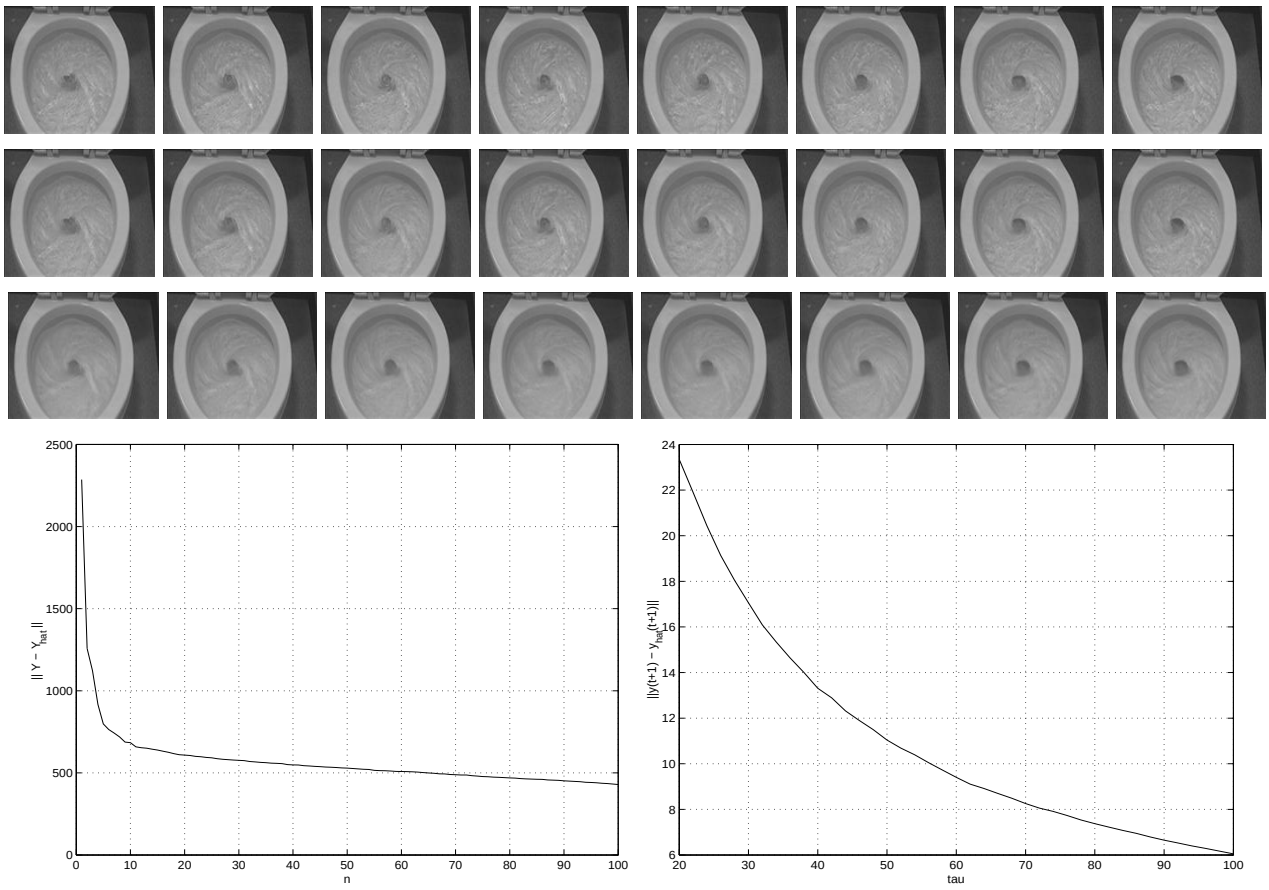

Figure 6: **Toilet.** *From top to bottom: samples of the original sequence, corresponding samples of the compressed sequence (compression ratio:* 2.53*), samples of extrapolated sequence (using* $n = 30$ *components,* $\tau = 120$*,* $m = 170 \times 115$*), compression error as a function of the dimension of the state space* n*, and extrapolation error as a function of the length of the training set* τ*. The data set used comes from the* MIT Temporal Texture database*.*

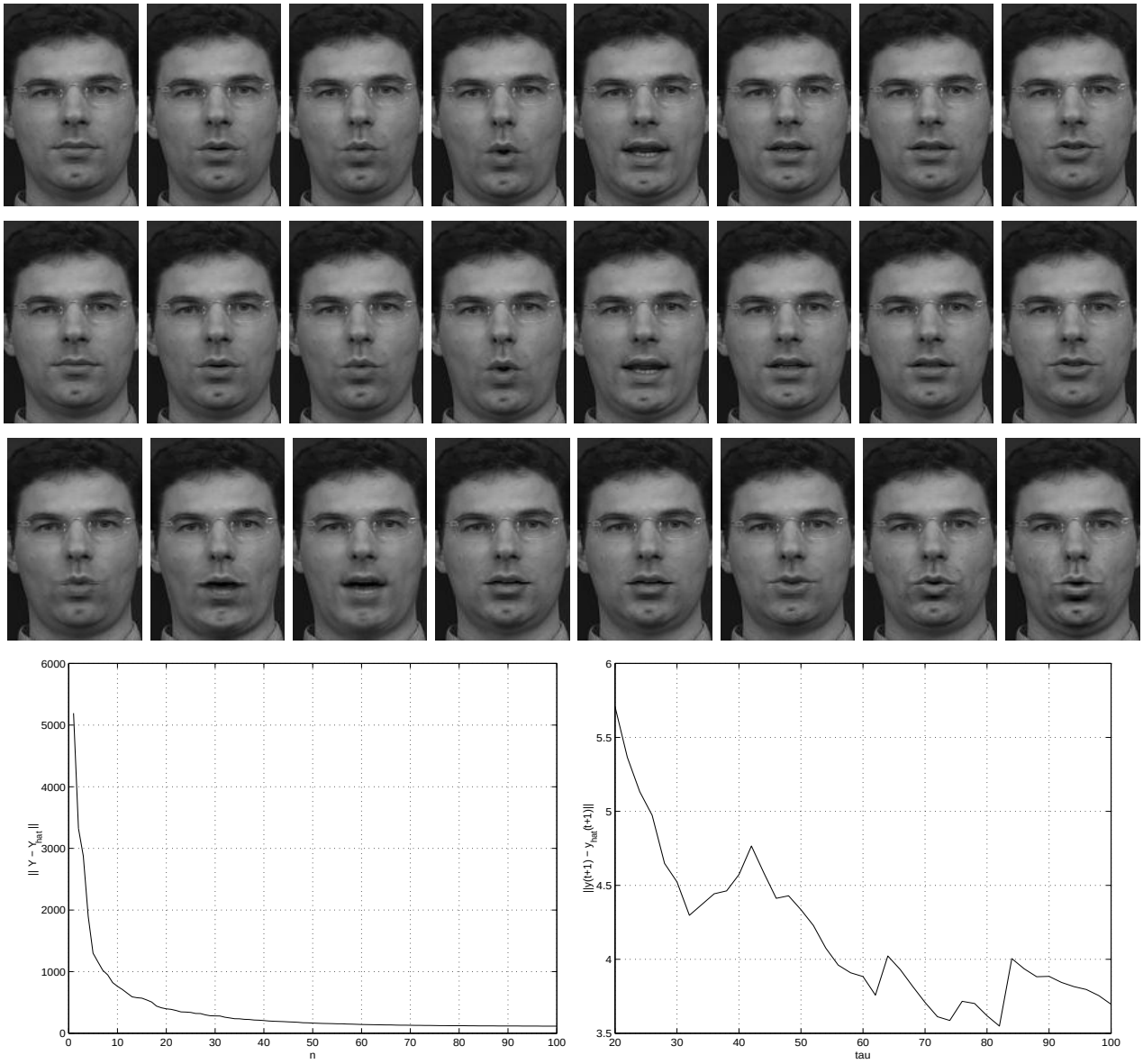

Figure 7: **Talking-face.** *From top to bottom: samples of the original sequence, corresponding samples of the compressed sequence (compression ratio:* 2.53*), samples of extrapolated sequence (using* $n = 40$ *components,* $\tau = 125$*,* $m = 105 \times 170$*), compression error as a function of the dimension of the state space* n*, and extrapolation error as a function of the length of the training set* τ*. This sequence has been proposed to show a synthesized sequence learned from a training set that violates the hypothesis of being a realization of a second-order stationary process. The result shows some artifacts, meaning that not all the information has been captured from the proposed model.*

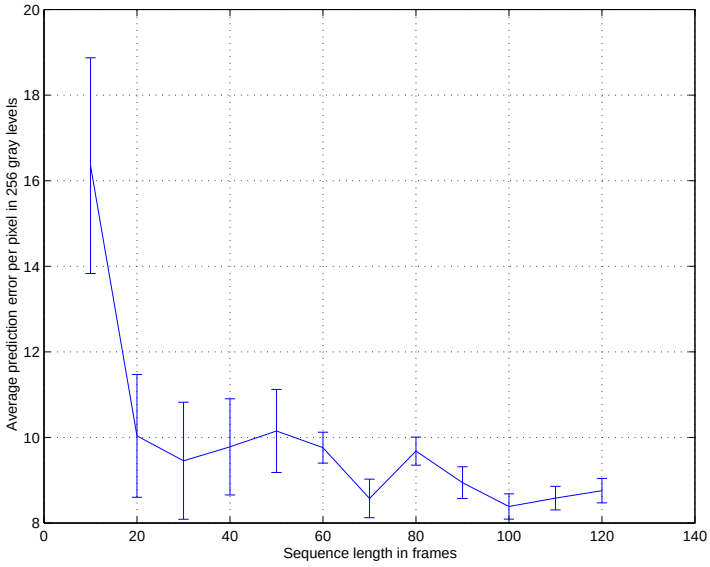

Figure 8: **Model verification** : *to verify the quality of the model learned, we have used a fixed number of principal components in the representation (20) and considered sub-sequences of the original data set of length varying from* 10 *to* 120*. We have used such sub-sequences to learn the parameters of the model in the Maximum-Likelihood sense, and then used the model to predict the next image. Using one criterion for learning (ML) and another one for validation (prediction error) is informative for challenging the model. The average prediction error per pixel is shown as a function of the length of the training sequence (for the* `steam` *sequence), expressed in gray levels within a range of* 256 *levels. The average error per pixel decreases and becomes stable after about 80 frames. Mean and standard deviation for* 100 *trials are shown as an error-bar plot.*

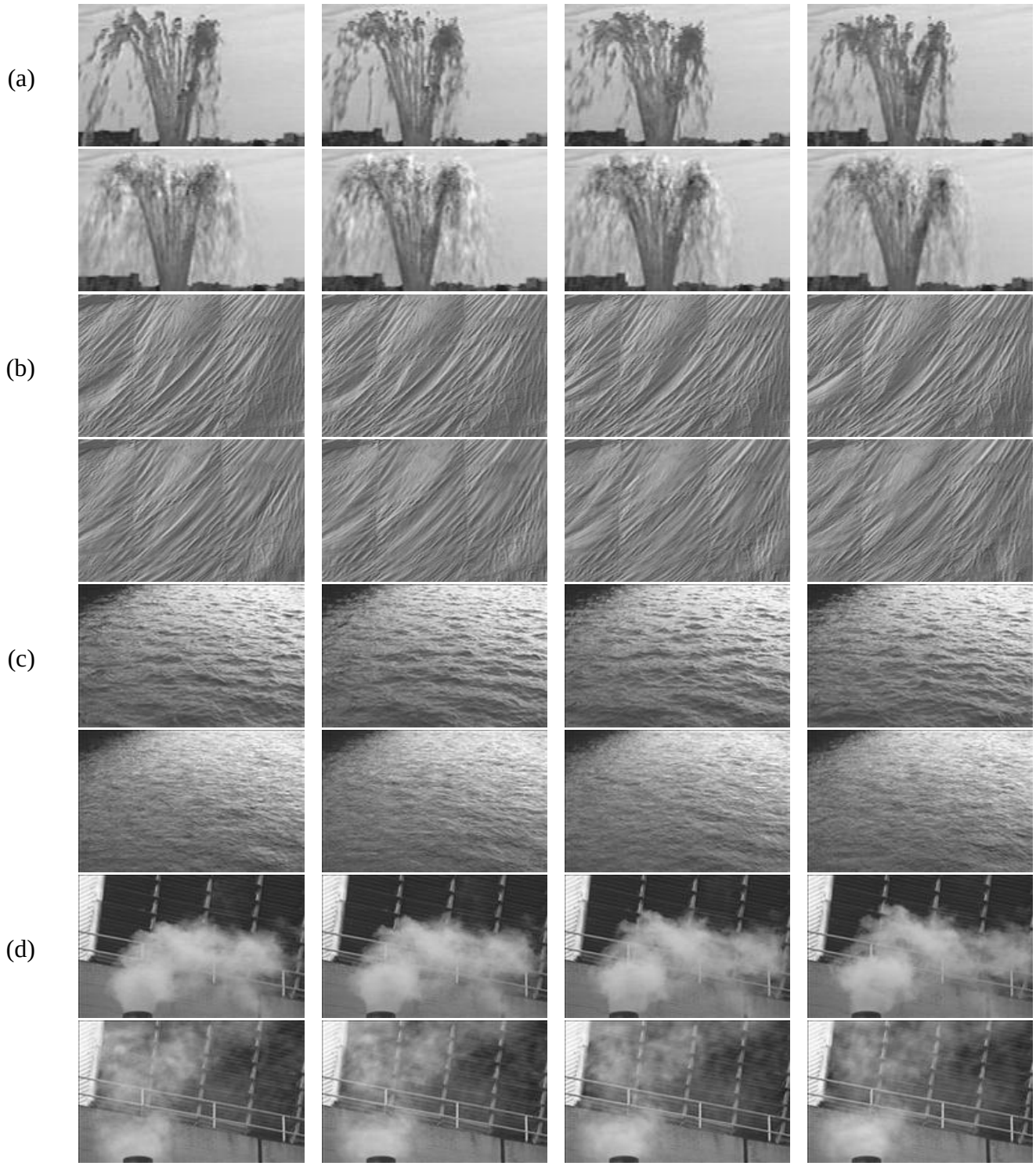

Figure 9: **Fountain, Plastic, River-far, Smoke-far.** *(a)* `fountain` *sequence* ($\tau = 100$, $m = 150 \times 90$)*, (b)* `plastic` *sequence* ($\tau = 119$, $m = 190 \times 148$)*, (c)* `river-far` *sequence* ($\tau = 120$, $m = 170 \times 115$)*, (d)* `smoke-far` *sequence* ($\tau = 150$, $m = 170 \times 115$)*. For each of them the top row are samples of the original sequence (borrowed from the MIT Temporal Texture database), the bottom row shows samples of the extrapolated sequence. All the data are available on-line at* `http://www.cs.ucla.edu/~doretto/projects/dynamic-textures.html`.

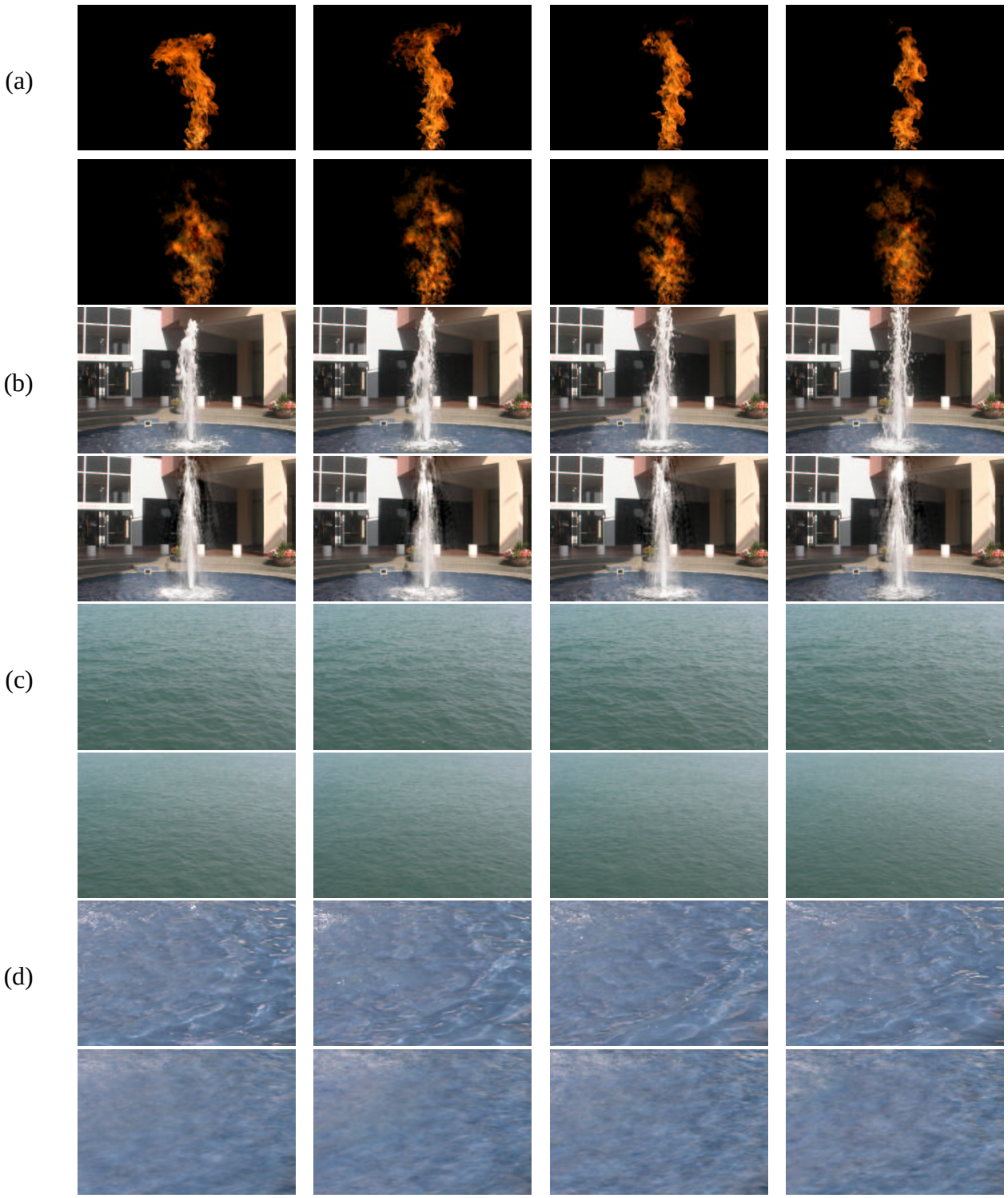

Figure 10: **Fire, Color-fountain, Ocean, Water [COLOR].** *(a)* `fire` *sequence* ($\tau = 150$, $m = 360 \times 243$)*, (b)* `color-fountain` *sequence* ($\tau = 150$, $m = 320 \times 220$)*, (c)* `ocean` *sequence* ($\tau = 150$, $m = 320 \times 220$)*, (d)* `water` *sequence* ($\tau = 150$, $m = 320 \times 220$)*. For each of them the top row are samples of the original sequence, the bottom row shows samples of the extrapolated sequence. All the data are available on-line at* `http://www.cs.ucla.edu/~doretto/projects/dynamic-textures.html`.

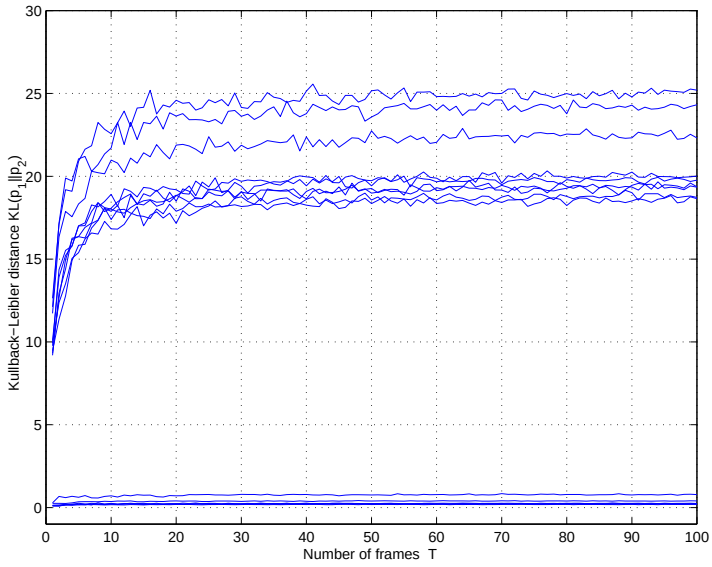

Figure 11: *The figure demonstrates that textures belonging to the same class tend to cluster together in the sense of Kullback-Leibler. In particular for this figure distances are computed amongst three realizations of the* `river` *sequence and three of the* `steam` *sequence w.r.t. the former. The cluster of graphs on top refer to "steam w.r.t. river" type of distances and the ones below refer to the "river w.r.t. river" type. The K-L divergences are computed using Monte-Carlo methods.*