UCLA-CSD-TR#020001

Editable Dynamic Textures

Gianfranco Doretto Stefano Soatto

Department of Computer Science, UCLA, Los Angeles - CA 90095

Abstract

We present a simple and efficient algorithm for modifying the temporal behavior of "dynamic textures," i.e. sequences of images that exhibit some form of temporal regularity, such as flowing water, steam, smoke, flames, foliage of trees in wind.

Keywords: Animation, Texture Synthesis, Video, Computer Vision, Image Processing, Compression.

1 Introduction

Our goal in this work is to *design algorithms for synthesizing and editing realistic sequences of images of dynamic scenes that exhibit some form of temporal stationarity.* While we will make this concept precise in Section 2.1, in brief, such scenes include flowing water, steam, smoke, flames, foliage of trees in wind, crowds, dense traffic flow etc. This is essentially a rendering task, and in particular we are interested in synthesizing the *temporal* behavior of the scene.

1.1 Video-based Modeling

The goal thus described is traditionally approached either by *physics-based* (PHB) techniques or by *image-based* (IMB) techniques. In PHB techniques, 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 [16, 6] and references therein) as well as walking gaits ([8] and references) and mechanical systems ([2] and references). The main advantage of these techniques is the extent in which the synthesis can be manipulated, resulting in great editing power (see e.g. [12]).

While conceptually PHB models are the most principled and elegant, they have the disadvantage of being computationally expensive, and of not taking into account the ultimate beneficiary of the simulation: the viewer. In fact, physical details in the model can be perceptually irrelevant, whereas some

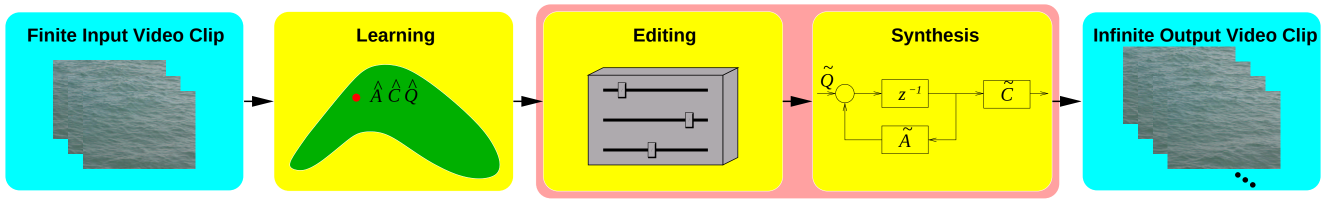

Figure 1: System diagram. *A finite length input video clip is fed to the* Learning module *which performs a closed-form suboptimal estimation of the model parameters. These are represented as a point on the model space manifold. The model parameters are then fed to the* Editing module *which allows manipulating groups of parameters while enforcing causality, stability, minimum-phase and other constraints that are necessary to yield a realistic perceptual outcome. The parameters are then fed* on-line *to the* Synthesis module *that interactively synthesizes live video.*

simplifications of the physical model can drastically alter the final perceived process to great detriment of realism[1].

IMB techniques address this issue at the outset, by generating synthetic sequences of images without building a physical model of the process that generate the scene. Among IMB one can distinguish between so-called "procedural" techniques that forego the use of a model altogether and generate synthetic images by clever concatenation or repetition of image data [17, 14], and IMB techniques that rely on a model, albeit not a physical one. In particular, IMB dynamical models are not models of the *scene*, but models of the *visual signal*, i.e. the image sequence itself. Such models can be deterministic or stochastic, and in general they fail to capture the correct geometry (shape), photometry (radiance) and dynamics (motion) of the scene. Instead, they capture a mixture of the three that is equivalent once visualized as an image. To the best of our knowledge, the only work in this category is [15]. We call this category *video-based modeling*[2].

A common shortcoming of all IMB techniques is their lack of flexibility. While the outcome of procedural algorithms or the simulation of video-based models produces in general very realistic results, the procedure is extremely hard to modify in ways that produce realistic or even meaningful results. Most IMB techniques merely allow the editor to extend the original sequence in space and time. Even the seemingly trivial task of speeding up or slowing down a fire, or a walking gait, is a challenge within the IMB framework, because of the lack of physical parameters that can be manipulated.

The focus of this paper is on editing video-based models.

1.2 Other Related Work

Since our emphasis is on time-varying textures, we do not address the vast literature on 2D (static) textures here. The problem of modeling dynamic textures has been first addressed by [10]. Procedural

[1]Indeed, since the complexity of the physical world greatly exceeds the complexity of the visual signal (which is the ultimate product of the simulation), great efforts in building a physically realistic model may go to waste during visualization and human perception.

[2]Since from images alone one cannot disentangle the correct (arbitrary) geometry, photometry and dynamics of the scene, one could argue that a phenomenological model, i.e. a model of the visual signal, is the most one can afford, and is sufficient as long as the final goal is to produce an image. The lesson learned from [15] is that, although the visual world can be very complex, the world as perceived by humans is relatively simple. For instance, images of complex nonlinear dynamical process are indistinguishable to the naked eye from simple linear Gaussian systems of very low order [15].

techniques have then been proposed by [17] and [14, 19]. Time-varying texture synthesis algorithms have also been proposed as extensions of 2D texture algorithms (see for instance [1] with [4] and [7]). Dynamic texture synthesis has been addressed in [15]. [5] addresses modeling for the purpose of matching, with no synthesis.

1.3 System Overview, Contributions of this Work and its Organization

Referring to Figure 1, the overall system described in this paper takes two types of input and produces one output. The inputs are (a) a finite video clip of a "dynamic texture" and (b) a certain set of values for a preset number of editing parameters. The notion of "dynamic texture" will be made precise in Section 2, and includes foliage of trees in wind, water, smoke etc. The output of the system is an infinite video of a dynamic texture similar to the original one, that can be interactively modified by acting on the editing parameters (b). In Section 4 we describe the interactive modification of the speed of the simulation (including positive and negative speeds), the spatial scales and the "intensity" of the simulation.

The overall system is composed of several "modules." The first module ("Learning") takes as input the original video clip and produces a representation of a dynamic texture in the form of a parametric dynamical model. This is borrowed from prior of [15] and is summarized in Sect. 2.

The novel content of this paper is in the second module, "Editing," which is described in Section 3. This module allows the editor to (1) simulate a novel sequence that has the same temporal statistics as the input video clip, and to (2) interactively modify the temporal characteristics of the simulation in order to achieve the desired perceptual effect.

To the best of our knowledge, this paper represents the first attempt to interactively modify the temporal statistics of a video-based model of a dynamic texture.

2 Dynamic Texture Modeling

What is a "dynamic texture"? In the *spatial* domain, the word "texture" suggests some form of statistical regularity or homogeneity. In the *temporal* domain, statistical regularity is captured by the notion of *stationarity*. 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 $t_2 - t_1$. Following [15], therefore, we say that *a sequence of images* $\{I(t)\}_{t=1\dots\tau} \in \mathbb{R}^m$ *represents a dynamic texture if it is a realization of a stationary stochastic process*[3]. In the experiments reported in Section 4, $m = 320 \times 220$. In order to make the paper self-contained, we briefly review the basic model of dynamic texture that we will later use for editing in Section 3.

2.1 Stationarity and Linear Gaussian Models

In this paper we restrict our attention to processes that are *second-order* stationary. These processes result in what are called *linear dynamic textures*. The name stems from the fact that any second-order stationary process can be represented as the output of a linear dynamical system driven by white, zero-mean

[3]The reader who is not familiar with the basic properties of stochastic processes can consult any standard textbook on the topic, for instance [11].

Gaussian noise [9]. These are called *linear Gaussian models*. Therefore, if we call $y(t) = I(t) + w(t)$ a sequence of images corrupted by white, zero-mean Gaussian noise $\{w(t)\} \in \mathbb{R}^m; w(t) \sim \mathcal{N}(0, Q)$, the assumption of second-order stationarity of $\{y(t)\}$ corresponds to the existence of a positive integer n, a process $\{x(t)\} \in \mathbb{R}^n$ 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

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

with $I(t) = Cx(t)$, for some matrices[4] $A \in \mathbb{R}^{n\times n}$ and $C \in \mathbb{R}^{m\times n}$.

Among dynamic textures, linear ones are the very simplest. And yet, they offer great modeling potential (see Section 4) while being amenable to linear analysis that results in simple analytical and computational tools, as we shall now see.

2.2 Learning Linear Dynamic Textures

The learning module in Figure 1 takes as input a finite, noisy sequence of images $\{y(1), \dots, y(\tau)\}$ and returns the model parameters A, C, Q, R. Ideally, we would want the maximum likelihood solution from the finite sample:

$$\hat{A}(\tau), \hat{C}(\tau), \hat{Q}(\tau), \hat{R}(\tau) = \arg \min_{A,C,Q,R} p(y(1) \dots y(\tau)) \tag{2}$$

The algorithm that achieves the optimum asymptotically as $\tau \to \infty$ has been derived in [18]. Unfortunately, the memory storage of the algorithm is quite taxing, and already $m = 320 \times 220$ and $\tau = 100$ challenge most desktop PCs. A simplified, suboptimal algorithm has been proposed by [15], and is described in pseudocode (Matlab) below.

```
function [x0,Ahat,Qhat,Chat,Vhat]=dytex(Y,n,k,tau)
 [U,S,V] = svd(Y-mean(Y,2)*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)));
 Vh = Xhat(:,2:tau)-Ahat*Xhat(:,1:(tau-1));
 [Uv,Sv,Vv] = svd(Vh,0);
 Bhat  = Uv(:,1:k)*Sv(1:k,1:k);Vhat = Vv(:,1:k);
 Qhat = Bhat*Bhat';
```

Variations of the algorithm that use independent components instead of principal components have been presented in [13], and both have been applied to modeling and recognizing walking gaits [3]. Once properly implemented, this algorithm runs in a few seconds on a high-end PC. The order of the model n can be chosen based on a tradeoff between realism and computational cost by looking at the profile of the energy of the principal components (i.e. the plot of the singular values of the covariance of $\{y(t)\}$). See [15] for details.

[4]Indeed, there are infinitely many matrices A, C, Q, R that give rise to the same sample paths $y(t)$, forming an equivalence class. While the interested reader can consult [15] on how to obtain a unique (canonical) representative of the equivalence class, this non-uniqueness does not have any impact on images synthesized from the model, and will therefore be ignored in this paper.

2.3 Synthesizing Dynamic Textures

Once a model has been learned[5] $\hat{A}, \hat{C}, \hat{Q}$, new sequences can be trivially generated by simulating the model (1). This entails choosing an initial condition $\hat{x}_0$ (for instance one of the original images), drawing a sample of an IID Gaussian process with covariance $\hat{Q}$, performing one step of the iteration $\hat{x}(t+1) = \hat{A}\hat{x}(t) + \hat{v}(t)$ and computing the new synthesized image as $\hat{I}(t) = \hat{C}\hat{x}(t)$. Pseudocode that implements the synthesis module in Figure 1 is reported below.

```
function [Ihat] = synth(x0,Y,Ahat,Qhat,Chat,tau);
 [n,k] = size(Bhat);
 Xhat(:,1) = x0;
 for t = 1:tau,
  Xhat(:,t+1) = Ahat*Xhat(:,t)+sqrt(Qhat)*randn(k,1);
  Ihat(:,t) = Chat*Xhat(:,t)+Y;
 end;
```

The two algorithms thus described allow generating synthetic sequences that match the spatio-temporal statistics of the original sequence, while never repeating[6] the original data. However, one would like to be able to *manipulate* the simulation and alter the statistics of the synthetic sequence. In other words, one would like to insert an *editing module*, as in Figure 1, before the synthesis. This is described in the next section.

3 Dynamic Texture Animation

The learning module described in Sect. 2.2 produces matrices $\hat{A}, \hat{C}, \hat{Q}$ that the synthesis module uses to generate the synthetic sequence $\{\hat{I}(t)\}$. In principle, any modification of the system parameters, for instance $\tilde{A}, \tilde{C}, \tilde{Q}$, results in a novel synthetic sequence $\{\tilde{I}(t)\}$ whose spatio-temporal statistics is altered with respect to the original sequence. Unfortunately, casual manipulation of the model parameters rarely results in sequences $\{\tilde{I}(t)\}$ that have any resemblance with realistic phenomena. First, the parameters in the model (1) cannot be chosen arbitrarily. In fact, $\tilde{A}$ must be stable (eigenvalues within the complex unit circle), $\tilde{C}$ must have orthogonal columns (in order to obtain a canonical realization, see footnote 4), $\tilde{Q}$ must be symmetric and positive-definite. In this section we describe how to manipulate the model parameters so that the resulting simulation is *admissible* (i.e. stable), and how to "map" model parameters onto phenomenological changes in the synthesized sequence.

Notice that the matrix R is associated with the measurement noise and is therefore meaningless in the synthesis process (unless someone wants to purposefully generate noisy images). Therefore, in the following subsections we discard R and describe how to change or invert the speed of a movie by manipulating A, or how to change the "intensity" of a pattern by acting on Q, or how to create visual effects by changing the spatial frequencies encoded in C.

[5]For simplicity, we omit the length of the training set τ.
[6]Assuming a perfect random number generator.

3.1 Visual Components and Spatial Scales

The learning procedure described in Section 2.2 produces a matrix $\hat{C}$ that has as its columns the first n principal components of the dataset (the singular vectors of the covariance). These components are by construction an orthonormal basis that spans a subspace in $\mathbb{R}^{m\times n}$. Therefore, an infinitely long synthesized dynamic texture can be viewed as a partial span of the subspace generated by the columns of $\hat{C}$. In practice, the state $x(t)$ assumes values in a bounded subset of $\mathbb{R}^n$ centered in 0.

The principal components are also sorted from the first to the last column of C in such a way that the spatial frequency they represent increases. Therefore, the first components (first columns of C) represent the coarse spatial scales of the texture pattern and the last components (last columns of C) represent the finest scales.

These considerations lead to the first type of manipulation, that is the spatial scale of the simulation. One can deform the actual subspace spanned by the principal components by re-weighting each component. This is simply done by substituting the matrix $\hat{C}$ with the matrix $\tilde{C} \doteq \hat{C}W$, where $W \in \mathbb{R}^{n\times n}$ is a diagonal matrix with the non-negative real numbers $w_1, \ldots, w_n$ as its diagonal entries. The last part of the synthesis algorithm is therefore substituted by

$$\hat{I}(t) = \hat{C}W\hat{x}(t). \tag{3}$$

3.2 Altering Speed

In this section we address the problem of "speeding up" or "slowing down" a synthetic process. Note that doubling the speed of a movie does not merely mean running the dynamical system at a double frame rate but, rather, to let the system produce half as many frames and give the visual perception that the speed has been doubled. This, however, has to be done while preserving the dynamic constraints of the model (1), and cannot be achieved by merely skipping frames or subsampling the original sequence.

Let us consider the decomposition $\hat{A} = V\Lambda V^{-1}$, where Λ is the diagonal matrix of eigenvalues and V is a matrix whose columns are the corresponding eigenvectors. If we write the eigenvalues in polar coordinates as $\{|\lambda_i| \exp(\mathbf{j}\psi_i)\}_{i=1\ldots n}$, where $\mathbf{j}$ is the imaginary unit, the normalized frequencies of the system are represented by $\{\psi_i\}_{i=k_1\ldots k_h}$, if h is the number of complex conjugate poles of the system. In order to change the speed of the movie one has to replace each frequency ψ_i with $\Omega\psi_i$, i.e. multiply the frequencies by the constant factor Ω:

$$\tilde{\psi}_i \doteq \Omega\psi_i. \tag{4}$$

Since we deal with discrete-time linear dynamical models, there is a limit for the normalized frequencies, that cannot exceed π. In principle, this imposes a limit on the maximum achievable speed. In fact, the complex conjugate poles at higher frequency, say ψ_K, impose the constraint $\Omega \leq \pi/\psi_K$. In practice, it is possible to achieve higher speeds by just eliminating the poles at higher frequency (i.e. set them equal to 0) once they reach the limit. Of course, the higher the speed, the higher the number of poles annihilated, the higher is the possibility of having a degraded quality of the resulting movie.

So far we have not mentioned the role of the values $\{|\lambda_i|\}_{i=1\ldots n}$, i.e. the distance of the poles from 0. These parameters affect the duration of the modes of the system once they have been excited. The lower they are, the shorter the duration of the modes and vice-versa. As mentioned above, all poles have to be inside the unit circle for the simulation to be stable. Intuitively, this means that modes with lower duration have to be heavily excited to be present if compared to modes with higher duration (poles closer to the unit circle). In practice, we have found that the dynamic textures we dealt with did have the complex conjugate poles very close to the unit circle, and varying their distance did not produce any worthwhile visual effect that is not achievable by playing with the visual components.

3.3 Reversing Time

In order to reverse the time axis, so as to invert the visual flow of a given dynamic texture, one may be tempted to simulate the model (1) "backwards" by starting from a given $x(t)$ and obtaining $x(t-1)$ from $x(t) = Ax(t-1)$ via $x(t-1) \doteq A^{-1}x(t)$. Unfortunately, this does not work since the resulting model has unstable dynamics (eigenvalues outside the complex unit circle). In practice, the simulation goes to overflow after a few iterations.

In order to gain some intuition on how to reverse the movie, consider a system with only a pair of complex conjugate poles $\lambda_{1,2} = |\lambda| \exp(\pm \mathbf{j}\psi)$, and eigenvectors of the matrix $\hat{A}$ given by $v_1 = v_2{}^*$, where * denotes the complex conjugate. It is straightforward to show that the free evolution of the system (i.e. for $\tilde{Q} = 0$) is equal to $x(t) = V\Lambda^t V^{-1}x(0) = |\lambda|^t(e^{\mathbf{j}\psi t}v_1\tilde{v}_1 + e^{-\mathbf{j}\psi t}v^*{}_1\tilde{v}_1^*)$, where $\tilde{v}_1$ is the first row of V^{-1}. Since we are not interested in altering the magnitude of the modes, we consider only the harmonic part of the state, i.e. $\bar{x}(t) \doteq e^{\mathbf{j}\psi t}v_1\tilde{v}_1 + e^{-\mathbf{j}\psi t}v^*{}_1\tilde{v}_1^*$ and observe that, if we change V with V^*, we obtain the quantity $\bar{x}'(t) \doteq e^{\mathbf{j}\psi t}v^*{}_1\tilde{v}_1^* + e^{-\mathbf{j}\psi t}v_1\tilde{v}_1 = \bar{x}(-t)$.

The above discussion can be extended to an arbitrary number of poles, and the reader should easily convince herself that, in order to reverse the time axis while maintaining the same temporal dynamics (speed), all we need to do is to substitute $\hat{A} = V\Lambda V^{-1}$ with

$$\tilde{A} \doteq V^* \Lambda V^{*-1}. \tag{5}$$

3.4 Intensity

In the absence of driving noise[7] $v(t)$, the output of the model (1) converges to a constant $y(t) \to \bar{y}$ that can take one of three equally uninteresting values: zero if A is stable, infinity if it is unstable[8], and a constant that depends on the initial condition if A is critically stable (some of the eigenvalues are on the complex unit circle). Therefore, for the synthetic sequence to have any practical interest, we must consider the role of $\{v(t)\}$, which is a white, zero-mean Gaussian IID process with covariance Q.

The covariance Q controls the intensity of the noise that drives the simulation. When Q is non-zero, the input $v(t)$ excites the modes of the state $x(t)$ and causes it to evolve as a discrete-time Brownian motion. The "size" of the eigenvalues of Q, i.e. the intensity of the driving noise, determines how far away from the initial condition the Brownian motion travels. In particular, since the estimated $\hat{Q}$ is symmetric and positive definite, there exists an orthonormal matrix[9] U and a diagonal matrix with positive values Λ_U such that $\hat{Q} = U\Lambda_U U^T$. The eigenvalues of Q can be altered by changing the elements of Λ_U, which changes the intensity of the individual components of the driving noise.

In Section 4 we show some experiments where we simply re-scale all the elements of Λ by a positive constant α, thus increasing ($\alpha > 1$) or decreasing ($\alpha < 1$) the intensity of each component of the driving noise by the same factor. In practice, we run the simulation after substituting $\tilde{Q}$ to $\hat{Q}$, where simply

$$\tilde{Q} \doteq \alpha \hat{Q}. \tag{6}$$

[7]Recall that in the synthesis phase we already have $w(t) = 0$.

[8]This is true only if the model is minimal, i.e. observable and controllable. The definition of these concepts is beyond the scope of this paper. Suffices here to say that the models learned from data as explained in Sect. 2 are minimal by construction.

[9]$UU^T = U^TU = I$.

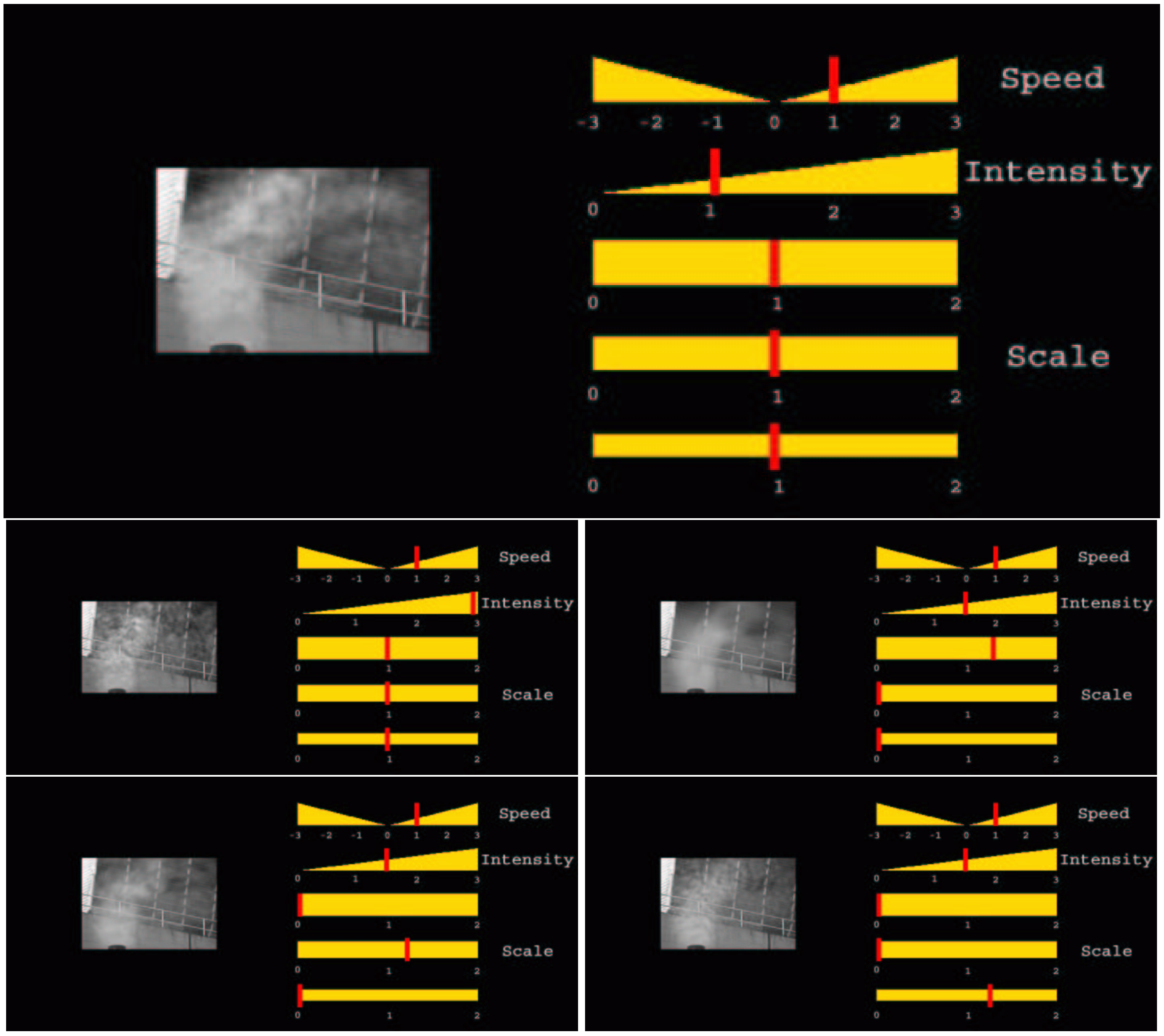

Figure 2: *Smoke. Top: a frame of the simulation (left) with the original set of parameters (right). Second from top: "turbulent smoke" (left), "hazy smoke" (right). Bottom: "patchy smoke" (right). The corresponding choice of parameters is shown on the right. The smoke can be sped up, slowed down and reversed.*

Finally, the modified synthetic sequence can just be generated by calling the function `synth` with parameters $\tilde{A}, \tilde{C}, \tilde{Q}$ instead of $\hat{A}, \hat{C}, \hat{Q}$.

4 Results

This section describes a set of representative experiments that illustrate the editing procedure we have proposed. The results are best seen in the movie you can download at `http://www.cs.ucla.edu/˜`

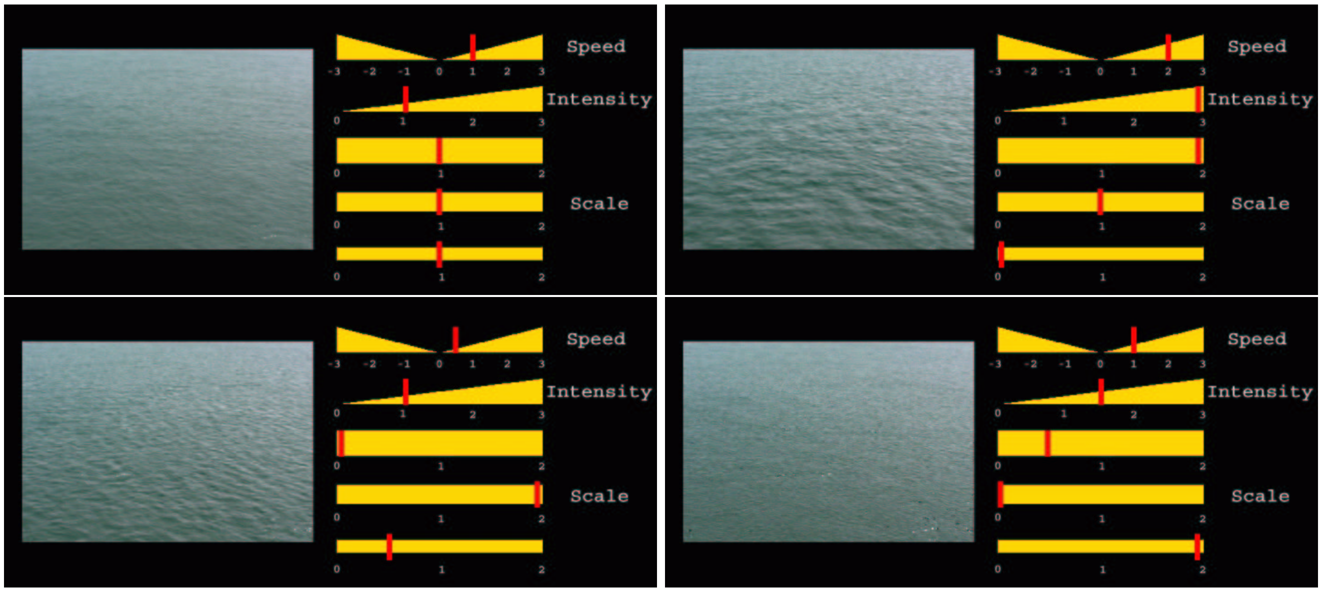

Figure 3: *Ocean waves [COLOR]. Top: "neutral" view, corresponding to the original parameters (left), "rough sea" (right). Bottom: "lake effect" (left), "rain" (right).*

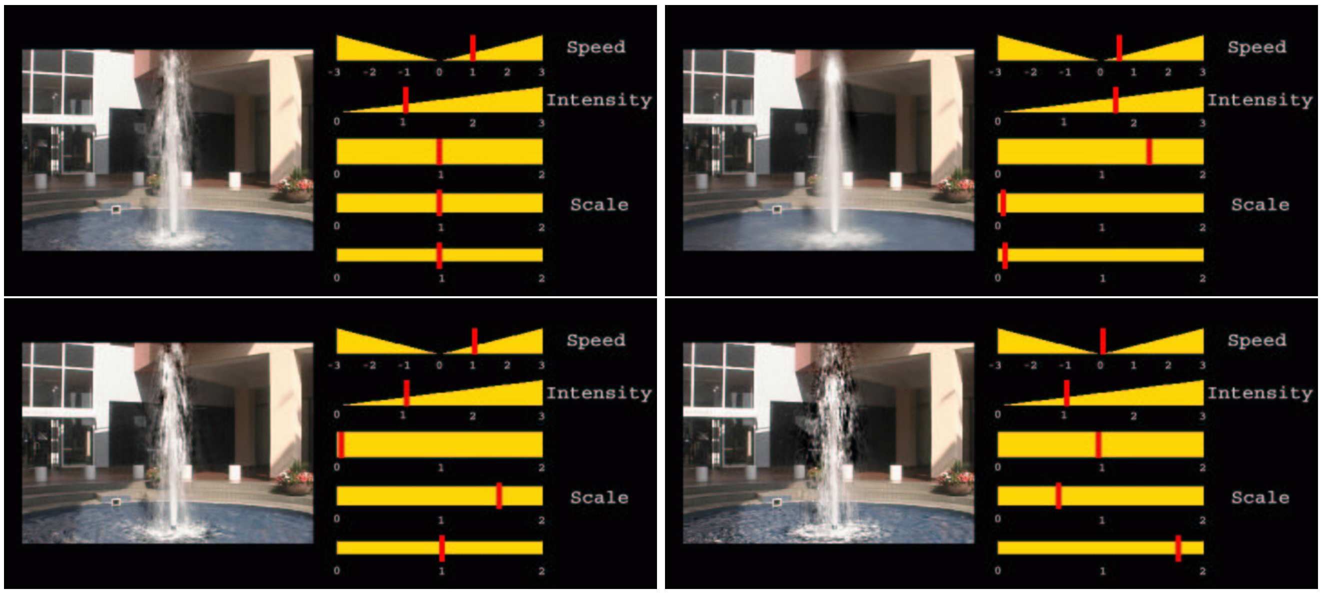

Figure 4: *Fountain [COLOR]. Different choices of intensity and scale parameters produce changes that appear to correspond to different nozzles, from a "spray-like" fountain (top right), to a "frothy" fountain (bottom left), to a "spurty" fountain (bottom right).*

`doretto/projects/dynamic-textures.html`, since paper is not a suitable medium to display dynamic images. This section can be used as a guide to walk through the movie. Notice that the movie has been compressed using MPEG, and therefore on certain displays it may appear spatially "blocky." This has nothing to do with the algorithm presented in this paper, and is merely a consequence of the MPEG encoding. Blockiness is not present in the uncompressed version of the movie, and a

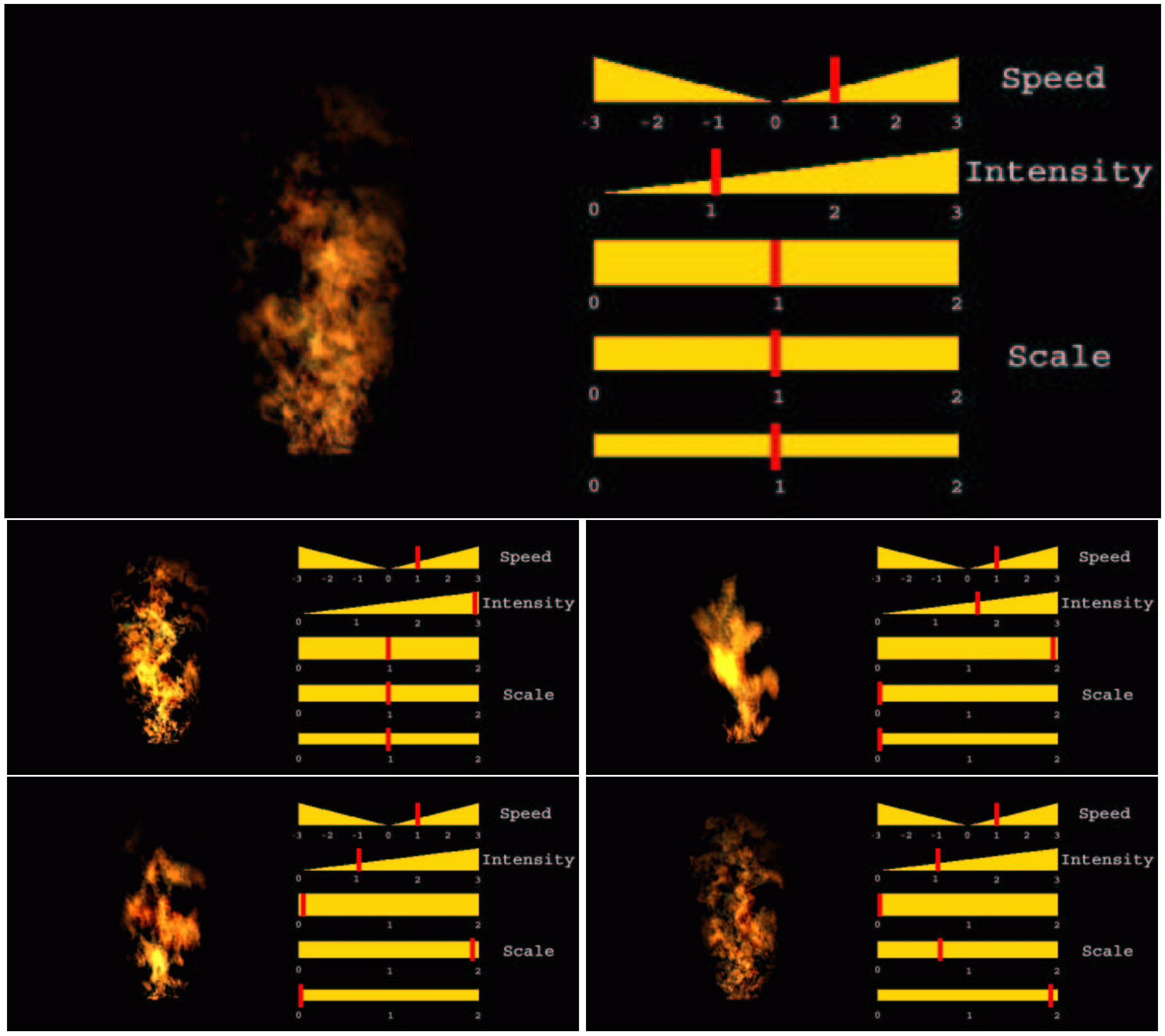

Figure 5: *Fire [COLOR]. Different choices of parameters produce changes in the apparent nature of the flame (different combustion characteristics). Non-realistic effects can also be achieved by altering the dynamics of each color component independently.*

high-quality display is recommended.

The first part of the movie ("Dynamic Texture Synthesis") is for background purposes and illustrates the results of [15], where a short input clip is used to learn a model $\hat{A}, \hat{Q}, \hat{C}$, which is then used to synthesize a longer version of the original sequence by just sampling a random Gaussian vector $\hat{v}(t)$ at each instant of time.

The second part of the movie ("Editing Dynamic Textures") describes the novel content of this paper, where the parameters Ω, V, α, and $w_1, \dots, w_n$ are used to generate modified (and yet admissible) parameters $\tilde{A}, \tilde{Q}, \tilde{C}$ to modify the simulation. In our experiments we have used some B/W movies (taken from the MIT Temporal Texture database[10]), and some color movies of 320×220 pixels. In all the experiments, the state dimension n has been set to 50.

Figures 2–5 show on the left a depiction of the scene, and on the right the corresponding values of the parameters Ω (speed), that ranges from 0 to 3 or from -3 and 0 depending on the value of V, α (intensity) from 0 to 3. The weights $w_1, \dots, w_n$ (visual components at different scales) have been divided in three groups $w_1 = \cdots = w_5$ (coarse scale), $w_6 = \cdots = w_{15}$ (medium scale), $w_{16} = \cdots = w_{50}$ (fine scale); their values ranges from 0 to 2.

Figure 2 (“Smoke”) shows, from top to bottom and from left to right, a frame from the synthetic movie obtained with neutral values for the parameters (i.e. $\tilde{A} = \hat{A}, \tilde{C} = \hat{C}$ etc.), a frame where the intensity of the driving input has been increased, which results in an apparently more “turbulent” smoke, a frame where the coarse frequency component has been amplified, which results in a thinner “hazy” smoke, a frame where the intermediate frequency has been amplified, and one where the high frequency has been amplified, resulting in a grainy, “patchy” smoke. In addition, the movie shows changes in speed where the smoke is sped up, then slowed down to a stop and reversed.

Figure 3 (“Ocean”) shows, from top to bottom and from left to right, a frame from the “neutral” synthetic movie, a frame where the intensity and the coarse and fine scales have been amplified, which results in a “rougher” sea movement with larger waves. The bottom left image shows what we call the “lake effect,” where the waves appear more gentle and smooth. Finally, increasing the intensity and the fine scale, while decreasing the coarse and middle scale results in a “rain effect” (bottom right), like rain pouring on a pond.

Figure 4 (“Fountain”) shows how playing with the intensity and scale parameters results in interesting effects that appear to be the results of changing the nozzle of the fountain, from a “spurty” fountain, to a “spray-like” fountain. Also, the fountain can be slowed down and brought to a complete stop.

Finally, Figure 5 (“Fire”) shows the effects of altering a dynamic texture of a flame, including changing the spatial scales, speed, direction etc.

4.1 Limitations and Extensions

This paper presents a way of modifying the simulation of a video-based model. Although we have shown that a wide variety of different dynamic textures can be obtained, ours remains an IMB model, and therefore the editing power is far from that of PHB models. Nevertheless, IMB models are conceptually and computationally simple, and we believe that being able to edit them is important.

This paper presents results for the simplest form of dynamic textures, that is *linear* ones. There are a number of directions in which these results can be extended. First, *non-linear* dynamic textures can be modeled, where the state evolves according to $x(t+1) = f(x(t), v(t))$ and the output is obtained by a non-linear map $y(t) = h(x(t), w(t))$. Second, *non-Gaussian* driving distributions can be explored. Furthermore, one needs not be restricted to dynamic textures in two dimensions: the output $y(t)$ can be a (vectorized version of) a tensor of any rank. Therefore, one can consider *three-dimensional dynamic textures*, for instance hair or cloth. In addition, the spatial component of the dynamic texture needs not live in a linear space. One can consider direct *synthesis over manifolds*, for instance surfaces represented in triangulated or implicit form.

Concerning *color*, in Section 4 we have shown results where all the channels were altered in the same way. For greater realism, one should consider the color vector as evolving on a sphere (or another

[10]`ftp://whitechapel.media.mit.edu/pub/szummer/temporal-texture`

constrained color space representation), whereas for greater creative power one may consider editing the dynamics of each color channel independently.

The algorithms we have described rely on modifying the *global* dynamics of the image. This, in our opinion, severely limits their applicability. Ultimately, one should explore integrating spatio-temporal *segmentation* with our techniques, in order to alter different portions or "layers" of the scene in different ways, and in order to overlay various visual phenomena onto existing scenes.

5 Conclusions

We have presented a method to edit the temporal statistics of a sequence of second-order stationary images, which we call "dynamic textures." To the best of our knowledge, work in this area is novel. Our technique consists in modifying the parameters of a linear Gaussian model, which is learned from an input sequence according to a technique proposed by [15], so as to maintain stability and minimality. In particular, we have described ways to edit the spatial frequency content of the sequence, modify or reverse the speed of the simulation and change the intensity of the driving noise.

While the techniques we describe will never allow the editor to achieve the flexibility of physics-based models, the hope is that, when coupled with spatial editing techniques such as warping, segmentation and mapping, they will add to the repertoire of tools available to game designers and special effect editors.

Acknowledgment

We acknowledge the support of NSF and Intel.

References

[1] Z Bar-Joseph, R El-Yaniv, D Lischinski, and M Werman. Texture mixing and texture movie synthesis using statistical learning. In *IEEE Transactions on Visualization and Computer Graphics*. To appear.

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

[3] Alessandro Bissacco, Alessandro Chiuso, Yi Ma, and Stefano Soatto. Recognition of human gaits. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Kauai, HA, December 2001.

[4] Jeremy S De Bonet. Multiresolution sampling procedure for analysis and synthesis of texture images. In *Proceedings of SIGGRAPH 1997*, Computer Graphics Proceedings, Annual Conference Series, pages 361–368. ACM, ACM Press / ACM SIGGRAPH, 1997.

[5] A. W. Fitzgibbon. Stochastic rigidity: image registration for nowhere-static scenes. In *Proceedings of IEEE International Conference on Computer Vision*, volume 1, pages 662–669, Vancouver, BC, Canada, July 2001.

[6] 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.

[7] David J Heeger and James R Bergen. Pyramid-based texture analysis/synthesis. In *Proceedings of SIGGRAPH 1995*, Computer Graphics Proceedings, Annual Conference Series, pages 229–238. ACM, ACM Press / ACM SIGGRAPH, August 1995.

[8] 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.

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

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

[11] A Papoulis. *Probability, Random Variables and Stochastic Processes*. McGraw-Hill, 1991.

[12] Jovan Popović, Steven M. Seitz, Michael Erdmann, Zoran Popović, and Andrew Witkin. Interactive manipulation of rigid body simulations. In Kurt Akeley, editor, *Proceedings of SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 209–218. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, July 2000. ISBN 1-58113-208-5.

[13] 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.

[14] Arno Schodl, Richard Szeliski, David H Salesin, and Irfan Essa. Video textures. In *Proceedings of SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series. ACM, ACM Press / ACM SIGGRAPH, January 2000.

[15] S. Soatto, G. Doretto, and Y. Wu. Dynamic textures. In *Proceedings of IEEE International Conference on Computer Vision*, volume 2, pages 439–446, Vancouver, BC, Canada, July 2001.

[16] J Stam and E Fiume. Depicting fire and other gaseous phenomena using diffusion processes. In *Proceedings of SIGGRAPH 1995*, Computer Graphics Proceedings, Annual Conference Series, pages 129–136. ACM, ACM Press / ACM SIGGRAPH, August 1995.

[17] Martin Szummer and Rosalin W Picard. Temporal texture modeling. In *Proceedings of IEEE International Conference on Image Processing*, volume 3, pages 823–826, Lausanne, Switzerland, 1996.

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

[19] Li Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series. ACM, ACM Press / ACM SIGGRAPH, 2000.