Higher Order SVD Analysis for Dynamic Texture Synthesis

Roberto Costantini, *Member, IEEE*, Luciano Sbaiz, *Member, IEEE*, and Sabine Süsstrunk, *Member, IEEE*

***Abstract*—Videos representing flames, water, smoke, etc., are often defined as *dynamic textures*: "textures" because they are characterized by the redundant repetition of a pattern and "dynamic" because this repetition is also in time and not only in space. Dynamic textures have been modeled as linear dynamic systems by unfolding the video frames into column vectors and describing their trajectory as time evolves. After the projection of the vectors onto a lower dimensional space by a singular value decomposition (SVD), the trajectory is modeled using system identification techniques. Synthesis is obtained by driving the system with random noise. In this paper, we show that the standard SVD can be replaced by a higher order SVD (HOSVD), originally known as Tucker decomposition. HOSVD decomposes the dynamic texture as a multidimensional signal (tensor) without unfolding the video frames on column vectors. This is a more natural and flexible decomposition, since it permits us to perform dimension reduction in the spatial, temporal, and chromatic domain, while standard SVD allows for temporal reduction only. We show that for a comparable synthesis quality, the HOSVD approach requires, on average, five times less parameters than the standard SVD approach. The analysis part is more expensive, but the synthesis has the same cost as existing algorithms. Our technique is, thus, well suited to dynamic texture synthesis on devices limited by memory and computational power, such as PDAs or mobile phones.**

***Index Terms*—Dynamic textures, tensors, texture synthesis, singular value decomposition (SVD).**

I. Introduction

DYNAMIC textures synthesis is the process of creating artificial textures. This can be achieved starting either from a description (model) of a physical phenomenon or from existing video sequences.

The first approach is called *physics-based* and leads to a description of the dynamic texture that usually requires few parameters. This approach has been extensively adopted for the reproduction of synthetic flames or fire, since they are often used in gaming applications or digital movies [2], [3]. Even though parameter tuning is not always straightforward, the synthesis results are impressive, but computationally extremely expensive. This limits the use of this type of model to cases where synthesis can be done offline, such as during editing in the movie making process.

The second approach is called *image-based*, as it does not aim at modeling the physics underlying the natural process, but at replicating existing videos. This can be done in two ways. In the first, synthesis is done by extracting different clips from the original video and patching them together to obtain a longer video, ensuring that the temporal joints are not noticeable and that the dynamic appearance is maintained [4], [5]. This type of synthesis is called *nonparametric* or *patch-based*, since it is not based on a model and reduces the synthesis to a collage of patches. It has the advantage of ensuring high visual quality because the synthetic video is composed of the original video frames, marginally modified by morphing operations only along clips discontinuities. However, the entire synthetic video has to be created in one step and stored in memory, thus not allowing for on-the-fly synthesis. In addition, this technique is not flexible, since it permits to modify the appearance of single frames, but not the texture dynamics.

In the second, a *parametric* image-based approach is used to build a model of dynamic textures. The dynamic texture is analyzed and model parameters are computed. The visual quality of the synthesis is generally less good than for patch-based techniques, but the parametric approach is more flexible, more compact in terms of memory occupation, and usually permits on-the-fly synthesis. Moreover, it can also be used for other applications, such as segmentation [6], recognition [7], and editing [8].

TABLE I
BRIEF SUMMARY OF THE DIFFERENT APPROACHES FOR DYNAMIC TEXTURE ANALYSIS/SYNTHESIS. LOW, AVERAGE, AND HIGH VALUES INDICATE A RELATIVE SCALE

	Physics-based	Image-based	
		Patch	Parametric
Analysis Cost	HIGH	LOW	LOW
Synthesis Cost	HIGH	LOW	LOW
Model Size	LOW	HIGH	LOW
Specificity	HIGH	LOW	LOW
Flexibility	HIGH	LOW	HIGH

Table I summarizes the different methods for analysis/synthesis of dynamic textures, highlighting their advantages and drawbacks. The model size is the number of coefficients of the model. The term "specificity" indicates if a given approach is specific to a certain type of dynamic texture, such as fire, water, or smoke, or can be used for all kinds of dynamic textures. The term "flexibility" indicates if the characteristics of the generated texture can easily be changed during the synthesis.

The physics-based approaches have high flexibility, but also high specificity, since a model for fire cannot be used for the generation of water or smoke, for instance. They have high flexibility since the visual appearance of the synthetic texture can be modified by tuning the model parameters. They have, however, high synthesis and analysis cost. Image-based approaches

Manuscript received March 27, 2007; revised August 24, 2007. This work was supported by the Swiss National Science Foundation (SNSF) under Grant 21-067012.01. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Michael Elad.

The authors are with the School of Computer and Communication Sciences, Ecole Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland.

Digital Object Identifier 10.1109/TIP.2007.910956

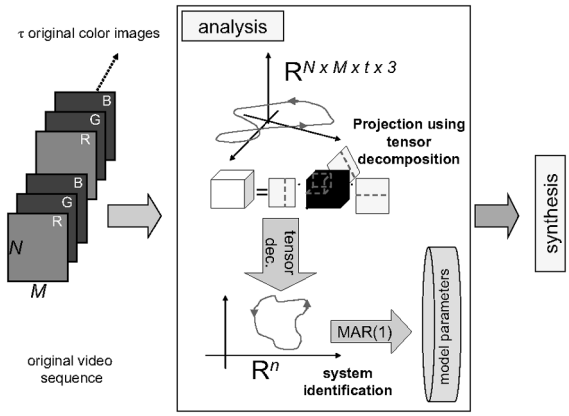


Fig. 1. Schematic representation of the tensor-based linear model approach for analysis and synthesis. The input video is analyzed directly as a multidimensional signal using a tensor decomposition technique. This is followed by a MAR(1) fitting, where the system dynamic is estimated. This gives as an output the parameters of the linear model that are used for synthesis.

are less specific, since the same model can be used to represent different dynamic textures by just changing the parameters. Low-memory occupation and synthesis cost favor the use of parametric models in situations where synthesis has to be done very fast, such as real-time applications.

Recently, the parametric model proposed by Doretto *et al.* [1], [9] was shown to be a valid approach for analysis/synthesis of dynamic textures. Each video frame is unfolded into a column vector and constitutes a point that follows a trajectory as time evolves. The analysis consists in finding an appropriate space to describe this trajectory and in identifying the trajectory using methods of dynamical system theory. The first part is done by using a singular value decomposition (SVD) to perform dimension reduction to a lower dimensional space. The point trajectory is then described using a multivariate auto-regressive (MAR) process of order 1. Dynamic textures are, thus, modeled using a linear dynamic system and synthesis is obtained by driving this system with white noise. In this model, the SVD exploits the temporal correlation between the video frames but the unfolding operations prevent the possibility of exploiting spatial and chromatic correlations.

We use the parametric approach of [1] but perform the dynamic texture analysis with a higher order SVD, which permits to simultaneously decompose the temporal, spatial, and chromatic components of the video sequence. This approach was proposed by the authors in [10] and here it is described in detail. Our scheme is depicted in Fig. 1. SVD in the analysis is substituted by HOSVD, while the MAR(1) model fitting is the same as in [1].

HOSVD is an extension of the SVD to higher order dimensions. It is not an optimal tensor decomposition in the sense of least squares data fitting and has not the truncation property of the SVD, where truncating the first n singular values permits to find the best n-rank approximation of a given matrix. Despite this, the approximation obtained is not far from the optimal one and can be computed much faster [11]. In fact, the computation of HOSVD does not require iterative alternating least squares algorithms, but needs standard SVD computation only.

The major advantage of the HOSVD is the ability of simultaneously considering the spatial, temporal, and chromatic correlations. This allows for a better data modeling than a standard SVD, since dimension reduction can be performed not only in the time dimension but also separately for spatial and chromatic content. Differences in vertical and horizontal spatial frequency content can be accounted for.

The separate analysis of each signal component allows adapting the signal "compression" given by the dimension reduction to the characteristics of each dynamic texture. For comparable visual synthesis quality, we, thus, obtain a number of model coefficient that is on average five times smaller than those obtained using standard SVD, as shown in Section IV. Creating more compact models is also addressed in [12], where dynamic texture shape and visual appearance are jointly addressed, thus enabling the modeling of complex video sequences containing sharp edges. Their and our approach are both characterized by a more computationally expensive analysis, but also a fast synthesis. In our case, synthesis can be done in real-time. This makes our technique very appropriate for applications with memory constraints, such as mobile devices.

We believe that HOSVD is a very promising technique for other video analysis and approximation applications. Recently, it has been successfully used in image based texture rendering [13], face superresolution [14], and in face analysis and recognition [15].

The article is structured as follows. In Section II, we present the definitions and properties of HOSVD, highlighting its difference with respect to standard SVD. In Section III, we describe the synthesis model based on HOSVD analysis and show how it is used for synthesis. Section IV describes the setup used to evaluate the algorithm performance. Section V presents the comparisons between the performance of our method and those obtained from two different methods having similar computational cost: standard SVD [1] and FFT [16]. Finally, Section VI concludes the article.

II. Higher Order SVD

Tensor decomposition was studied in psychometric data analysis during the 1960s, when data sets having more than two dimensions (generally called "three-way data sets") became widely used [17]. A fundamental achievement was brought by Tucker (1963), who proposed to decompose a 3-D signal using directly a 3-D principal component analysis (PCA) instead of unfolding the data on one dimension and using the standard SVD. This three-way PCA [18] is also known as Tucker3 decomposition. In the 1980s, such multidimensional techniques were also applied to chemometrics analysis.

The signal processing community only recently showed interest in the Tucker3 decomposition. The work of Lathauwer *et al.* (2000) [11], [19] proved that this decomposition is a multilinear generalization of the SVD to multidimensional data. Studying its properties with a notation more familiar to the signal processing community, the authors highlighted its properties concerning the rank, oriented energy, and best reduced-rank approximation. As the decomposition can have higher dimensions than 3, they called it higher order SVD (HOSVD). In the following, we consider the notation of [11] and define the HOSVD decomposition.

A. Definitions and Properties

We denote tensors as calligraphic letters ($\mathcal{A}, \mathcal{B}$, etc.), matrices as bold capital letters ($\mathbf{A}, \mathbf{B}$, etc.), vectors as bold small letters ($\mathbf{a}, \mathbf{b}$, etc.), and scalar as small letters (a, b, etc.).

Tensors are generalization of matrices of *orders* higher than 2; a tensor $\mathcal{A} \in \mathbb{R}^{(I_1 \times I_2 \times \cdots \times I_p)}$ has order p and $I_1, I_2, \ldots, I_p$ are integer numbers indicating the number of elements for each dimension. For example, a grayscale video sequence can be considered a tensor of order 3, with $I_1 = N, I_2 = M$, and $I_3 = \tau$, if it is composed by τ video frames of dimension $N \times M$ pixels.

By unfolding its elements along dimension h, a matrix unfolding $\mathbf{A}_{(h)}$ is obtained, whose columns are called h-mode vectors. The h-rank of tensor $\mathcal{A}$ is the rank of the h-mode matrix unfolding $\mathbf{A}_{(h)}$.

The product of a tensor $\mathcal{A} \in \mathbb{R}^{(I_1 \times I_2 \times \cdots \times I_q \times \cdots \times I_p)}$ and a matrix $\mathbf{B} \in \mathbb{R}^{J_q \times I_q}$ is denoted by $\mathcal{A} \times_q \mathbf{B}$ and is a tensor $\mathcal{C} \in \mathbb{R}^{(I_1, I_2, \ldots, I_{q-1}, J_q, I_{q+1}, \ldots, I_p)}$, where

$$c_{i_1 i_1 \ldots i_{q-1} j_q i_{q+1} \ldots i_p} = \sum_{i_q} a_{i_1 i_1 \ldots i_{q-1} i_q i_{q+1} \ldots i_p} u_{j_q i_q}.$$

Note that this is the product between $\mathbf{B}$ and the matrix unfolding $\mathbf{A}_{(q)}$ along the q dimension: $\mathbf{C}_{(q)} = \mathbf{B}\mathbf{A}_{(q)}$.

In tensors, the matrix unfoldings can have different ranks and the notion of tensor rank does not coincide with the notion of h-rank in the 2-D case. A rank 1 tensor is a tensor given by the outer product of p vectors $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_p$

$$a_{i_1 i_1 \ldots i_p} = (\mathbf{u}_1)_{i_1} (\mathbf{u}_2)_{i_2} \ldots (\mathbf{u}_p)_{i_p} \tag{1}$$

for all values of the indices, where $(\mathbf{u}_j)_{i_i}$ is the ith component of vector $\mathbf{u}_j$. The rank of an arbitrary p-order tensor $\mathcal{A}$, denoted as $\mathrm{rank}(\mathcal{A})$, is the minimal number of rank-1 tensors that yield $\mathcal{A}$ in a linear combination. Thus, the rank of a tensor can be different form its h-rank, even when all h-ranks are equal, and, in general, is difficult to determine in a direct way [11].

The standard SVD is schematically depicted in Fig. 2(a). In the top half, it is formulated according to standard notation, i.e., as the matrix product between a left matrix $\mathbf{U}$, a diagonal matrix $\mathbf{S}$, and a right matrix $\mathbf{V}^H$. Since bidimensional matrices are a particular case of tensors of order 2, this product can also be expressed using tensor notation. This is shown in the bottom half of Fig. 2(a).

The HOSVD introduced in [11] is an extension of the three-way Tucker decomposition to higher orders. A p-order tensor $\mathcal{A}$ is decomposed as

$$\mathcal{A} = \mathcal{S} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \cdots \times_p \mathbf{U}^{(p)} \tag{2}$$

where $\mathbf{U}_{(1)}, \mathbf{U}_{(2)}, \ldots, \mathbf{U}_{(p)}$ are orthogonal matrices that contain the orthonormal vectors spanning the column space of the matrix unfolding $\mathbf{A}_{(i)}$ with $i = 1, 2, \ldots, p$. An example in the case $p = 3$ is depicted schematically in Fig. 2(b). $\mathcal{S}$ is defined as the *core* tensor. It corresponds to the generalization of the matrix $\mathbf{S}$ in the standard SVD. Generally, $\mathcal{S}$ is a full tensor, not a diagonal matrix as $\mathbf{S}$.

The HOSVD is computed in two steps according to the following algorithm.

1) For $i = 1, 2, \ldots, p$, compute the unfolding matrix $\mathbf{A}_{(i)}$ from $\mathcal{A}$ and compute its standard SVD: $\mathbf{A}_{(i)} = \mathbf{U}\mathbf{S}\mathbf{V}^H$; the orthogonal matrix $\mathbf{U}^{(i)}$ is defined as $\mathbf{U}^{(i)} = \mathbf{U}$, i.e., as the left matrix of the SVD.
2) Compute the core tensor using the inversion formula

$$\mathcal{S} = \mathcal{A} \times_1 \mathbf{U}^{(1)^H} \times_2 \mathbf{U}^{(2)^H} \cdots \times_p \mathbf{U}^{(p)^H} \tag{3}$$

where the symbol ^{H} denote the Hermitian matrix transpose operator.

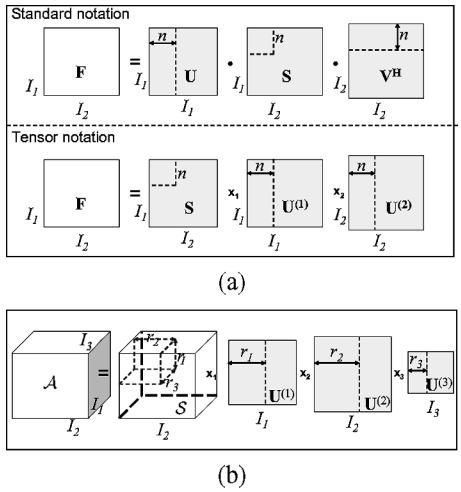


Fig. 2. Standard and multiway SVD. (a) Standard SVD of a matrix $\mathbf{F}$ and its components $\mathbf{U}, \mathbf{V}$ (unitary matrices), and $\mathbf{S}$ (diagonal matrix); we depict the decomposition using both matrix and tensor notation; the best n-rank approximation of $\mathbf{F}$ is obtained taking the first n rows/columns of matrices $\mathbf{U}$ and $\mathbf{V}$, and a $n \times n$ diagonal matrix from $\mathbf{S}$, as indicated. (b) Multiway SVD: the 3-D input signal is decomposed into a product of a tensor core $\mathcal{S}$ (a full matrix) and three unitary matrices $\mathbf{U}^{(1)}, \mathbf{U}^{(2)}$, and $\mathbf{U}^{(3)}$. An approximation of $\mathcal{A}$ is obtained by taking the first r_1, r_2, and r_3 components as shown; this will not give the best (r_1, r_2, r_3)-rank approximation of $\mathcal{A}$, but the computation is much faster and the difference in approximation is generally small.

Standard SVD, illustrated in Fig. 2(a), is used as a dimension reduction technique. If $\mathbf{F}$ is a matrix that is decomposed using standard SVD, the product of the first n columns of the left and n rows of the right matrices with the first n elements of the diagonal matrix $\mathbf{S}$ produces the best n-rank approximation matrix for $\mathbf{F}$. This property does not extend to orders greater than 2. In tensors where $ps > 2$, simple truncation of the first $r_1 < I_1, r_2 < I_2, \ldots, r_p s < I_p$ columns of the matrices $U^{(i)}$ [see Fig. 2(b)] does not produce the best rank-$(r_1, r_2, \ldots, r_p)$ approximation of $\mathcal{A}$. The computation of the best rank approximation requires an iterative alternated least-square (ALS) algorithm and is quite time consuming [19]. However, the approximation obtained from simple truncation has been proven to be in most cases quite similar to the optimal approximation [11].

III. HOSVD-Based Analysis and Synthesis

A dynamic texture can be considered as a multidimensional signal. In the case of a grayscale image video, it can be represented with a 3-D tensor by assigning spatial information to the first two dimensions and time to the third. In a color video sequence, chromatic components add another dimension. The input signal then becomes 4-D.

The analysis is done by first decomposing the input signal using the HOSVD and then by considering the orthogonal matrix derived from the decomposition along the time dimension.

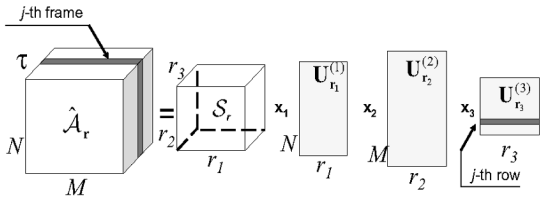


Fig. 3. Schematic representation of the product between truncated matrices $\mathbf{U}_{r_i}^{(i)}$ and tensor $\mathcal{S}_{\mathbf{r}}$, resulting in the approximated tensor $\hat{\mathcal{A}}_{\mathbf{r}}$. Here, we consider a 3-D tensor representing a grayscale video sequence. I_1 and I_2 are the spatial dimension and I_3 the temporal dimension. The jth video frame is obtained by considering the product of the tensor $\mathcal{S}_{\mathbf{r}}$, the matrices $\mathbf{U}_{r1}^{(1)}$ and $\mathbf{U}_{r2}^{(2)}$, and the jth row of matrix $\mathbf{U}_{r3}^{(3)}$. The latter matrix corresponds to the decomposition of the input tensor along the time axis and is used in the analysis part to find the MAR(1) model of dynamics.

This matrix contains the dynamics of the video sequences, since its columns, ordered along the time axis, correspond to the weights that control the appearance of the dynamic texture as time evolves.

Let $\mathcal{A}$ be a $(I_1 \times I_2 \times \cdots \times I_p)p$-order tensor and let (t) be the index corresponding to the temporal dimension. For example, a grayscale video sequence composed by τ video frames of size $N \times M$ has $p = 3s,\ I_1 = N,\ I_2 = M,\ I_3 = \tau$, and $(t) = (3)$. For a color video sequence $p = 4$ and the fourth dimension is $I_4 = 3$, corresponding to the number of color channels. Let k be the discrete time index $k = 1, 2, \ldots, \tau$.

A. Analysis

First Step: We decompose the tensor $\mathcal{A}$ using the HOSVD as $\mathcal{A} = \mathcal{S} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \ldots \times_t \mathbf{U}^{(t)} \times_{t+1} \ldots \times_p \mathbf{U}^{(p)}$. We have specifically highlighted the contribution given by the orthogonal matrix $\mathbf{U}^{(t)}$, as it corresponds to the decomposition of the input signal unfolded along the time axis. Before the decomposition, we subtract the temporal pixel average $\mathcal{M}$, in order to have a zero-mean component in the time axis

$$(\mathcal{M})_{i_1 \ldots i_{t-1} i_{t-1} \ldots i_p} = \frac{1}{I_t} \sum_{k=1}^{I_t} (\mathcal{A})_{i_1 \ldots i_{t-1}\, k\, i_{t+1} \ldots i_p}. \quad (4)$$

Dimension reduction is applied by taking the first $r_1, r_2, \ldots, r_p$ components of the orthogonal matrices and of the core tensor $\mathcal{S}$, as indicated schematically in Fig. 2(b) in the case of a 3-D tensor. We denote with a subindex the matrices and tensor thus obtained: $\mathbf{U}_{r_1}^{(1)}, \mathbf{U}_{r_2}^{(2)}, \ldots, \mathbf{U}_{r_p}^{(p)}$ and $\mathcal{A}_{\mathbf{r}}$, where $\mathbf{r} = (r_1, r_2, \ldots, r_p)$ and $\mathbf{U}_{r_i}^{(i)} = [\mathbf{u}_1^{(i)}, \mathbf{u}_2^{(i)}, \ldots, \mathbf{u}_{r_i}^{(i)}]$, with $i = 1, \ldots, p$.

Second Step: We denote the matrix $\mathbf{U}_{r_t}^{(t)} \in \mathbb{R}^{I_t \times r_t}$ as $\mathbf{X}$ and $\mathbf{x}_j$ as its jth row. This simplifies the notation. The rows of this matrix contain the information about the dynamics of the input video sequence. This is shown in Fig. 3, where for simplicity we report the example of the decomposition of a 3-D tensor that represents a grayscale video sequence. The sequence has τ frames composed by $N \times M$ grayscale images. When a dimension reduction is done, the input data is approximated by $\hat{\mathcal{A}}_{\mathbf{r}} = \mathcal{S}_{\mathbf{r}} \times_1 \mathbf{U}_{r_1}^{(1)} \cdots \times_t \mathbf{X} \times_{t+1} \ldots \mathbf{U}_{r_p}^{(p)}$, where $\hat{\mathcal{A}}_{\mathbf{r}}$ denotes an approximation of $\mathcal{A}_{\mathbf{r}}$. In Fig. 3, we note that the jth frame of the input video sequence is obtained by multiplying the core tensor with the orthogonal matrices $\mathbf{U}_{r_1}^{(1)}, \mathbf{U}_{r_2}^{(2)}$, and with the jth row of matrix $U_{r_3}^{(3)}$.

The dynamic is represented using a multivariate auto-regressive model of order 1. This means that we are looking for a $r_t \times r_t$ matrix $\mathbf{H}$ such that

$$\mathbf{x}_{j+1} = \mathbf{H}\mathbf{x}_j + \mathbf{e}_{j+1} \quad (5)$$

where $\mathbf{e}_{j+1}$ is a residual error. To find the matrix $\mathbf{H}$, we use a classical least-squares technique as proposed in [1]. We call $\mathbf{P_1}$ the matrix including the first $\tau - 1$ rows of $\mathbf{X}$ and $\mathbf{P_2}$ the matrix including the last $\tau - 1$ rows of $\mathbf{X}$: $\mathbf{P_1} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{\tau-1}]$ and $\mathbf{P_2} = [\mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_\tau]$. Matrix $\mathbf{H}$ is computed as

$$\mathbf{H} = \mathbf{P_2}\mathbf{P_1}^T(\mathbf{P_1}\mathbf{P_1}^T)^{-1}. \quad (6)$$

The residual part is modeled in order to have $\mathbf{e}_{j+1} = \mathbf{G}\mathbf{v}_{j+1}$, where $\mathbf{v} \in \mathbb{R}^{n_v}$ is a Gaussian random vector $\mathcal{N}(0, I_{n_v} \times I_{n_v})$. $\mathbf{G}$ is obtained by estimating the input noise covariance matrix $\mathbf{Q}$ from the residual matrix $\mathbf{E} = [\mathbf{e}_2, \mathbf{e}_3, \ldots, \mathbf{e}_\tau]$ and imposing that

$$\mathbf{G}\mathbf{G}^T = \mathbf{Q}. \quad (7)$$

Since this procedure is the same as in [1], we refer to it for further details.

B. Synthesis

The model parameters are as follows:

- matrices $\mathbf{U}_{r_i}^{(i)}$ with $i = 1, 2, \ldots, t-1, t+1, \ldots, p$, obtained from the first r_i left singular vectors of each of the p unfolding matrices of tensor $\mathcal{A}$ (we recall that the index (t) is the one associated to the time dimension);
- tensor $\mathcal{S}_{\mathbf{r}}$, obtained from (3) applied to the left singular truncated matrices $\mathbf{U}_{r_i}^{(i)}$;
- matrices $\mathbf{H}$ and $\mathbf{G}$, obtained from (6) and (7), respectively;
- tensor $\mathcal{M}$, obtained from (4).

The dynamic texture is represented by the following system of linear equations

$$\begin{cases} \mathbf{x}_{k+1} = \mathbf{H}\mathbf{x}_k + \mathbf{G}\mathbf{v}_k \\ \mathcal{Z}_k = \mathcal{S}_{\mathbf{r}} \times_1 U_{r_1}^{(1)} \cdots \times_t \mathbf{x}_{k+1} \times_{t+1} \cdots \times_p U_{r_p}^{(p)} + \mathcal{M}, \end{cases} \quad (8)$$

where $\mathcal{Z}_k$ indicates the tensor generated at time k when the system is driven by random noise $\mathbf{v}$ and corresponds to a synthetic video frame. The dimension of this tensor is $(I_1 \times I_2 \times \ldots I_{t-1} \times 1 \times I_{t+1} \times \cdots \times I_P)$. The term $\mathcal{M}$ indicates the temporal average subtracted before the decomposition.

IV. Performance Evaluation Setup

We evaluate the performance of the HOSVD-based linear model of (8) with respect to algorithms that perform synthesis with a similar computational cost. We have chosen two algorithms for comparison: the algorithm of Doretto *et al.* [1], where the analysis is done using standard SVD, and the algorithm of Abraham *et al.* [16], which uses the same analysis of Doretto's in the Fourier Transform domain. This is done to exploit the spatial correlation among pixels, since each frequency component

of a frame is a function of all its pixels. Moreover, a threshold mechanism allows to select only the most significant FFT coefficients, thus resulting in a model having generally less coefficients. The drawback is that the synthesis requires an inverse FFT operation for each frame.

We use three dynamic texture videos for the comparisons. The videos are "Flame," "Grass," and "Pond." Figs. 6–8 show some video samples. The video sequence "Flame" is characterized by a strongly varying dynamic since the sequence alternately shows samples of small and large amplitude and by low to medium spatial frequency content. The sequence "Grass" is characterized by low-frequency temporal content and high-frequency spatial content along the horizontal axis, given by the repetition of the grass pattern. The last sequence shows low-frequency spatial and temporal content. Following the Claerbout *et al.* principle on reproducible research [20], this article is completely reproducible; thus, all the videos, test results, and code are available online [21].

A. RGB to YC_bC_r Conversion

The native color encoding of a texture video is RGB. For the experiments, we have chosen to consider also a different color encoding: YC_bC_r in 4:2:0 format (denoted simply as YC_bC_r from now), i.e., having the chrominance channels downsampled both horizontally and vertically by a factor of two. This is an encoding commonly used for videos that permits to obtain a smaller size dynamic texture representation, as shown in [22].

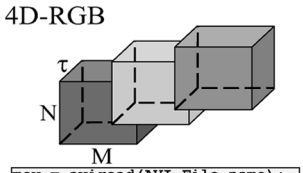
We have, thus, three algorithms for each color encoding. The first algorithm is Doretto's basic algorithm [1]. Applied to RGB and YC_bC_r images it is denoted respectively as 2-D RGB and 2-D YC_bC_r.

The second is the FFT-based algorithm of Abraham [16]. Since the original implementation considers grayscale images only, we extended it to color by applying the algorithm as is to the three color channels separately, both in the case of RGB and YC_bC_r color encoding. This avoids major modifications. The resulting implementations are denoted as FFT-RGB and FFT-YC_bC_r, respectively.

The third is the HOSVD-based algorithm we propose. Here, we have chosen to use the data representation that is depicted schematically in Fig. 4. In the first representation (left), data is organized in a 4-D tensor according to horizontal, vertical, temporal, and chromatic dimensions, respectively. We show the three color components separately to indicate a 4-D tensor. In the second representation (right), YC_bC_r color encoding is used to compact the color image frames into a single plane that contains luminance and chrominance channels. Chrominance is downsampled by a factor of two both horizontally and vertically. Fig. 4 reports also the MATLAB code used to create this data structure, where we use the MATLAB Tensor Classes developed by Kolda *et al.* [23]. The algorithm derived from the first data representation is called 4-D RGB, the latter 3-D YC_bC_r. Table II summarizes the naming convention used to refer to the algorithms.

B. Prediction Error

In order to evaluate the algorithms performance, we consider the error between original and synthetic frames. Two factors




```
mov = aviread(AVI_File_name);
tau = length(mov);
im = getfield(mov,{1,1},…
     'cdata');

[N M C] = size(im);
% Creating the 4D-RGB matrix
for t = 1:tau
 im = getfield(mov,{1,i},…
          'cdata'));
 Data(:,:,t,1) = im(:,:,1);
 Data(:,:,t,2) = im(:,:,2);
 Data(:,:,t,3) = im(:,:,3);
end
```

```
% Creating the 3D-YCbCr matrix
for t = 1:tau
 im = getfield(mov,{1,i},…
        'cdata'));
 Ycbcr = rgb2ycbcr(im));

 Lum = Ycbcr(:,:,1);
 Cb = imresize(Ycbcr(:,:,2),0.5);
 Cr = imresize(Ycbcr(:,:,3),0.5);
 Data(1:N,1:M,i) = Lum;
 Data(N+1:3*N/2,1:M/2,i) = Cb;
 Data(N+1:3*N/2,M/2+1:M,i) = Cr;
end
```

Fig. 4. Data organization. Left: Data is organized in a 4-D tensor where the first two dimensions correspond to space, the third dimension corresponds to time, and the fourth dimension is color. Right: Data is organized in a 3-D tensor where the first two dimensions correspond to spatial and color information, while the third corresponds to time. Bottom: MATLAB code used to obtain such representation is given.

TABLE II
NAMING CONVENTION OF THE ALGORITHMS USED FOR THE COMPARISON TESTS

	RGB	YC_bC_r
HOSVD-Based	4D-RGB	3D-YC_bC_r
Doretto's	2D-RGB (as in [1])	2D-YC_bC_r (as in [22])
FFT-Based	FFT-RGB (as in [16])	FFT-YC_bC_r (modified [16])

contribute to this error. The first is the error intrinsic to the low-rank approximation given by the choice of the number of singular values. The second is the error introduced when modeling the texture dynamic with an MAR(1) model. The latter is properly called *prediction error*.

In the case of the FFT-based algorithm and also when using YC_bC_r color encoding, there are two additional errors. The FFT algorithm introduces an error when the FFT coefficients are truncated, while for YC_bC_r, the error is given by the color upsampling and converting operations. Here, we call *prediction error* the total error between an original frame and its prediction obtained by the models in final RGB color encoding that is used for display purposes. Mathematically, this error is defined as follows:

$$\frac{1}{\tau-1}\sum_{j=2}^{\tau} 10\log_{10}\frac{255^2}{\mathrm{MSE}\left(\mathbf{I}_j^{\mathrm{RGB}}-\hat{\mathbf{I}}_j^{\mathrm{RGB}}\right)} \tag{9}$$

where $\mathbf{I}_j^{\mathrm{RGB}}$ and $\hat{\mathbf{I}}_j^{\mathrm{RGB}}$ are the original and predicted video frames in RGB video format, respectively. The one step image prediction and the corresponding frame synthesis are computed as follows:

$$\begin{cases}\hat{\mathbf{x}}_j = \mathbf{H}\mathbf{x}_{j-1}\\ \hat{\mathcal{Z}}_j = \mathcal{S}\times_1 U^{(1)}\cdots\times_t \hat{\mathbf{x}}_j\times_{t+1}\cdots\times_n U^{(n)}+\mathcal{M}\end{cases} \tag{10}$$

where $\mathbf{x}_{j-1}$ is the $(j-1)$th column of matrix $\mathbf{X}$ obtained in the analysis step, and frame $\hat{\mathbf{I}}_j^{\mathrm{RGB}}$ is obtained from $\hat{\mathcal{Z}}_j$ by an appropriate color conversion (when needed).

The parameter n_v introduced in Section III-A does not play a role here, since the term $\mathbf{G}\mathbf{v}_j$ of (8) is not considered when the one step prediction is computed.

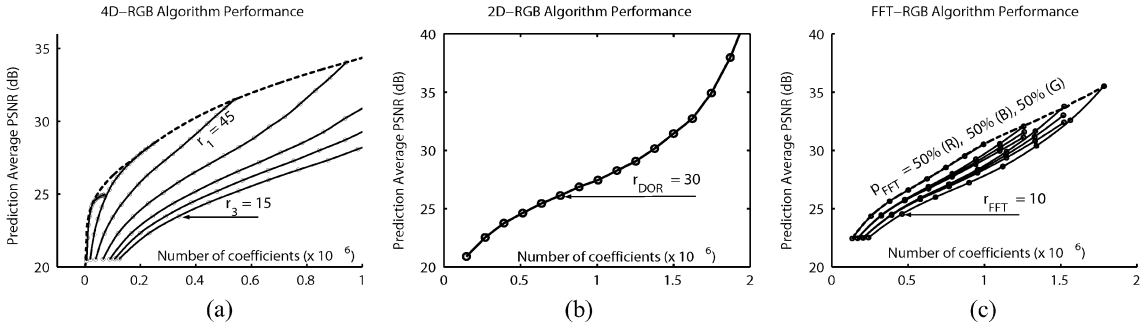


Fig. 5. Example of the performance (prediction error versus model size) obtained by the proposed tensor-based algorithm and by the two other algorithms used for comparisons using the dynamic texture "Flame." (a) Tensor-based performance using 4-D RGB data representation: we have varied r_1 (spatial) and r_3 (temporal) components, while $r_4 = 3$. The dots indicate the performance obtained at different values of r_3, while the lines are computed at constant r_1; the dashed line indicates the optimal performance and is obtained by interpolating the data at disposal from simulation. (b) Doretto's algorithm: Dots indicate different values of r_{DOR}. (c) FFT-based algorithm: Dots indicate different value of r_{FFT}, while the lines correspond to different combination of the FFT coefficient percentage for R, G, and B, respectively. The best performance is indicated by the dashed line (in this case it corresponds to $p_{\mathrm{FFT}} = 50\%$ for red, green, and blue channels).

TABLE III

NUMBER OF MODEL COEFFICIENTS AND COST OF THE SYNTHESIS. THE "DECOMPOSITION COEFFICIENTS" ARE THOSE OBTAINED BY THE DECOMPOSITION OF THE INPUT SIGNAL USING THE HOSVD, SVD, AND FFT, RESPECTIVELY; THE "DYNAMICS COEFFICIENTS" ARE THE COEFFICIENTS CORRESPONDING TO THE SYSTEM DYNAMICS, I.E., THE NUMBER OF COEFFICIENT OF MATRICES $\mathbf{H}$ AND $\mathbf{G}$ OF (8). THE COST OF THE SYNTHESIS IS INDICATED IN NUMBER OF MULTIPLICATION OPERATIONS; ADDITIONS ARE NOT TAKEN INTO ACCOUNT, SINCE THEY CAN BE INCORPORATED IN MULTIPLICATION. THE TERM CC INDICATES THAT A COLOR CONVERSION IS ALSO NEED. WE DO NOT INDICATE THE CONTRIBUTION OF THE TEMPORAL AVERAGE $\mathcal{M}$, SINCE IT IS NEGLIGIBLE. IT CORRESPONDS TO 3 MN FOR RGB AND TO 1.5 MN FOR YC_bC_r MODELS, RESPECTIVELY

	Decomposition Coefficients	Dynamic Coefficients	Synthesis Cost (main contribution)
4D-RGB	$Nr_1 + Mr_2 + 3r_4 + r_1r_2r_3r_4$	$r_3(r_3 + n_v)$	$3MNr_3$
3D-YC_bC_r	$1.5Nr_1 + Mr_2 + 3r_4 + r_1r_2r_3r_4$	$r_3(r_3 + n_v)$	$1.5MNr_3 + CC$
2D-RGB	$3NMr_{DOR}$	$r_{DOR}(r_{DOR} + n_v)$	$3MNr_{DOR}$
2D-YC_bC_r	$1.5NMr_{DOR}$	$r_{DOR}(r_{DOR} + n_v)$	$1.5MNr_{DOR}$ + CC
FFT-RGB	$3NMr_{FFT}p_{FFT}/100$	$r_{FFT}(r_{FFT} + n_v)$	$3NMr_{FFT}p_{FFT}/100$ + IFFT
FFT-YC_bC_r	$1.5NMr_{FFT}p_{FFT}/100$	$r_{FFT}(r_{FFT} + n_v)$	$1.5NMr_{FFT}p_{FFT}/100$ + IFFT + CC

C. Comparison Setup

The parameters of the HOSVD-based model are the number of singular values retained from the tensor decomposition. Since this decomposition considers spatial (horizontal and vertical), temporal, and chromatic components separately, this results in a number of four parameters in the case of the 4-D RGB data representation and three parameters in the case of 3-D YC_bC_r. We indicate as r_1, r_2, r_3, and r_4 the parameters associated to vertical, horizontal, temporal, and color dimensions, respectively.

This gives enough freedom to optimize the decomposition according to the characteristics of the video sequence. However, to keep the number of parameters comparable to the other methods and to limit the number of simulations, we imposed that $r_1 = r_2$. Naturally, this penalizes the algorithm performance when applied to sequences having a clear spatial orientation, such as the "Grass" video sequence, which is characterized by strong horizontal frequency and low vertical frequency. Using $r_2 > r_1$ would have increased the quality of the predicted frames, since the horizontal content would have been better approximated. Moreover, in the case of 4-D RGB, we fixed $r_4 = 3$, thus not compressing the color information.

The parameters r_1 varied from 5 to N with an interval of $5, r_2$ from 5 to M. Since N and M are in general different, we allow r_1 differ from r_2 when one or the other reaches its maximum value. In all other cases, the restriction $r_1 = r_2$ is valid. The temporal parameter r_3 varied from 5 to $\tau - 5$, with an interval of 5, where τ indicates the temporal length of the video sequence.

Doretto's algorithm performance varies according to one single parameter. This is the number of singular values that are retained from the 2-D SVD in the analysis. We call this parameter r_{DOR}: it is an integer ranging from 1 to τ. When testing the algorithm, we varied r_{DOR} from 5 to$\tau - 5$ with an interval of 5.

The FFT-based algorithm performance can be changed by tuning two parameters. As in the previous method, the first parameter is the number of singular temporal values retained, indicated as r_{FFT}. The second parameter is the number of most significant FFT coefficients, indicated as a percentage value p_{FFT}. During the experiment we have varied r_{FFT} from 5 to $\tau - 5$ with an interval of 5, and we have considered 5 different values for p_{FFT}: 90%, 80%, 70%, 60%, and 50%.

Fig. 5 shows an example of the algorithms' performance obtained for the video sequence "Flame." The dots indicate simulation results, while the continuous lines indicate their interpolation. The dashed line corresponds to the best performance obtained by the algorithm for this video sequence.

D. Model Size and Synthesis Cost

The model size is the number of model coefficients and results from two contributions: the coefficients of the SVD or HOSVD decomposition and those related to the system dynamics. In Table III, we report the sizes of the models and the synthesis cost (main contribution). Since r_3, r_{DOR}, and r_{FFT} usually have similar values, the synthesis cost is comparable for the all algorithms.

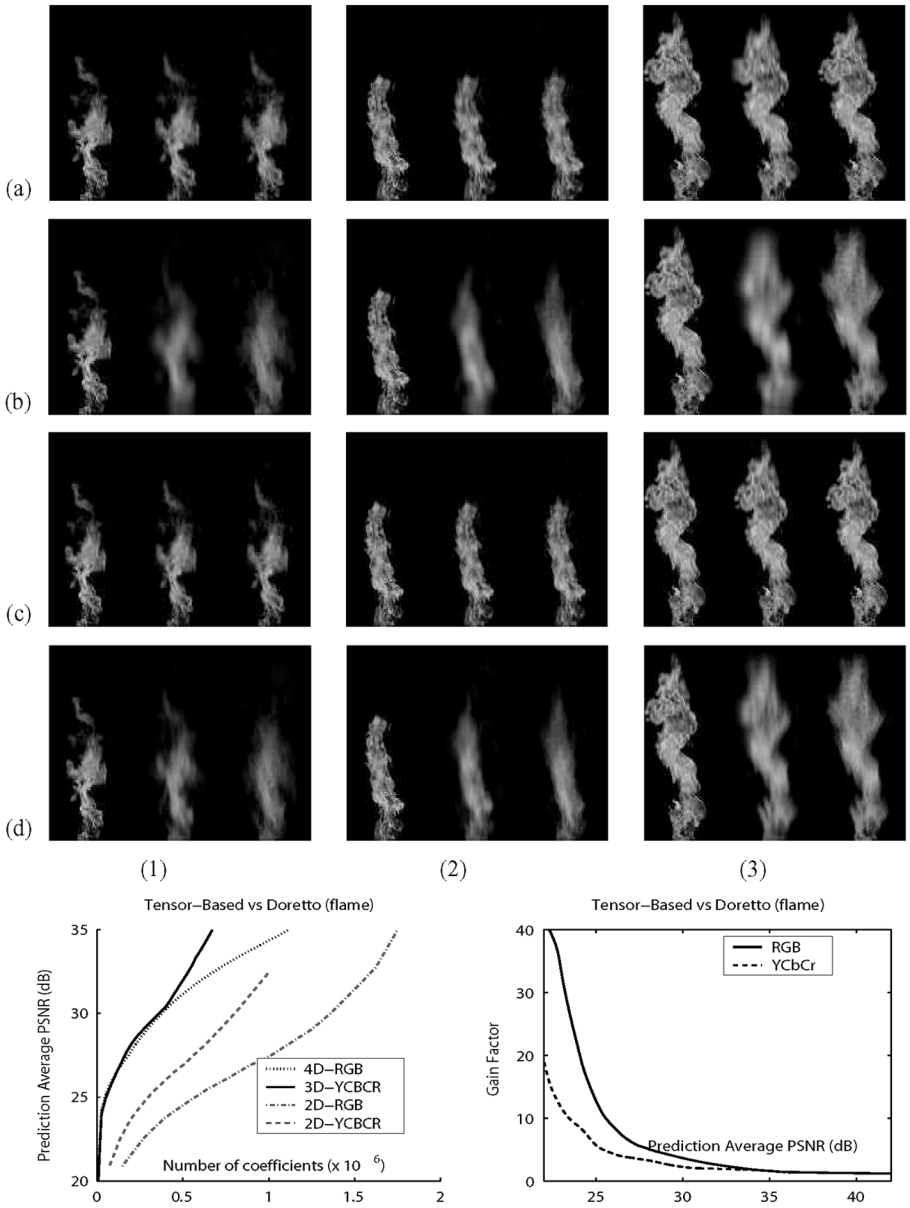


Fig. 6. Performance comparison between the tensor-based and Doretto's model using the test video "Flame." Images (a)–(d) report the result of the synthesis; each image shows three flames; left: the original flame; center: synthetic flame obtained using the tensor-based mode; right: the synthetic flame obtained using Doretto's model. Images (a) and (b) have been obtained ensuring a prediction error of 30 and 22.5 dB, respectively, by using the tensor-based model on a 4-D RGB data representation and the standard 2-D RGB Doretto's model. Images (c) and (d) have been obtained ensuring a prediction error of 30 and 22.5 dB, respectively, by using the tensor-based model on a 3-D YC_bC_r data representation and the modified Doretto's algorithm (indicated in the paper as 2-D YC_bC_r). Images in columns (1)–(3) have been obtained from the second, 12th, and 25th frames of the video sequence "Flame." The last row reports the comparison between models' performance. The results have been obtained using both data representations. The parameters of the tensor-based decomposition are chosen to ensure the best performance; at left, we report the prediction error versus the model size, while, at right, we report the gain in terms of model coefficients, defined as the ratio between Doretto's and tensor-based model's coefficients.

E. Model-Order Selection

In Soatto and Doretto's model, dimension reduction is controlled by only one parameter. Using HOSVD for the analysis, three or four parameters can be varied. Parameters r_1 and r_2 correspond to the spatial content of the video for vertical and horizontal dimensions, respectively. They can be chosen according to the spatial content of each video frame. In the case of the "Grass" sequence, for instance, the presence of vertical grass blades in each frame motivates using a higher value for r_2, since the horizontal frequency content is more dominant compared to the vertical one. For other sequences, such as for "Pond," r_1 and r_2 can be set to similar values, since there is not a clear prevalence of high-frequency components in either direction.

The parameter r_3 models the temporal content of the video and corresponds to the parameter n of Doretto's model. It can be chosen according to the same rationale, i.e., according to the temporal frequency content of the video.

In the 4-D RGB representation, r_4 describes the chromatic content. Dimension reduction can also be imposed to the color information. This is only appropriate, however, when the color content of the video is not varying rapidly, for example with

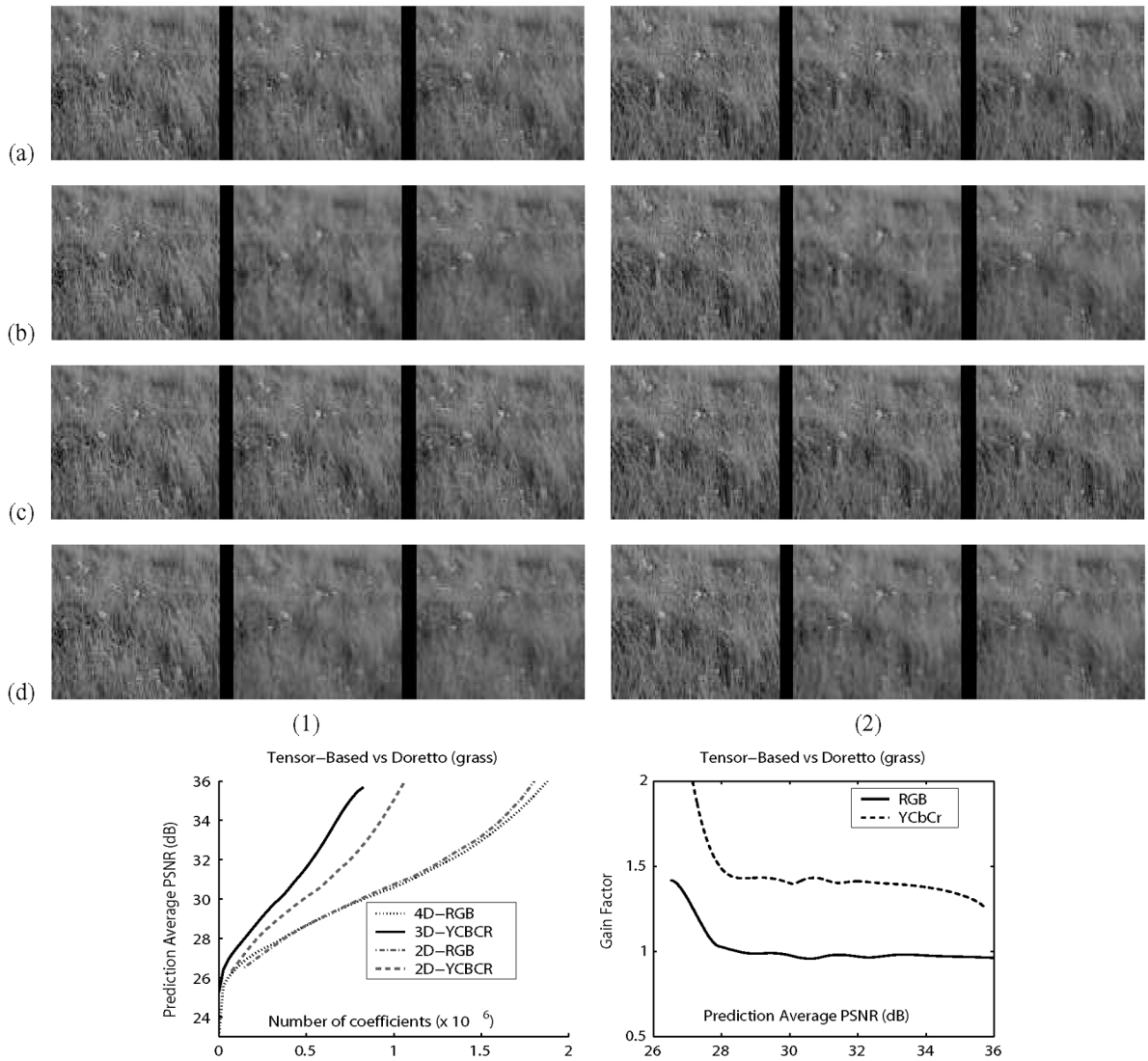


Fig. 7. Performance comparison between the tensor-based and Doretto's model using the test video "Grass." Images (a)–(d) report the result of the synthesis; each image shows three grass fields; left: the original one; center: result of synthesis using the tensor-based mode; right: result of synthesis using Doretto's model. Images (a) and (b) have been obtained ensuring a prediction error of 30 and 22.5 dB, respectively, by using the tensor-based model on a 4-D RGB data representation and the standard 2-D RGB Doretto's model. Images (c) and (d) have been obtained ensuring a prediction error of 30 and 26 dB, respectively, by using the tensor-based model on a 3-D YC_bC_r data representation and the modified Doretto's algorithm (indicated in the paper as 2-D YC_bC_r). Images in columns (1)–(3) have been obtained from the second and 16th frames of the video sequence "Grass." The last row reports the comparison between models' performance. The results have been obtained using both data representations. The parameters of the tensor-based decomposition are chosen to ensure the best performance; at left, we report the prediction error versus the model size, while, at right, we report the gain in terms of model coefficients, defined as the ratio between Doretto's and tensor-based model's coefficients.

the "Flame" video, where color changes only slightly with time and/or space. In this case, we can assign a smaller value, such as $r_4 = 2$, i.e., using only two basis function for the chromatic content of the video.

V. Results

The results obtained are shown in Figs. 6–9. Each Figure is composed by two parts. In the first part (top), images of the original and synthetic videos are depicted, obtained at low (22.5 dB) and at a high (30 dB) visual quality. In the second part (bottom), two plots show the comparison between the HOSVD-based and a reference algorithm. The plot at the left reports the prediction error and the number of model coefficients. The plot at the right shows the ratio between the number of coefficients of the HOSVD-based model and that of a reference model that ensures the same prediction error. In the case of the HOSVD and FFT-based models, the performance shown in the figures corresponds to the best performance, i.e., the one indicated in Fig. 5 by a dashed line (for the "Flame" sequence).

With the exception of the video sequence "Grass," the HOSVD-based model needs far less coefficients compared to the other models. For a PSNR average value associated to a good visual quality of the synthetic video (approximatively from 25 to 30 dB), the gain reaches a value varying from 5 to 10. In other words, 5 to 10 times less coefficients are needed when using the HOSVD-based model, still ensuring the same prediction quality. To view the synthesized videos, please refer to our web-page at [21]. Indeed, the overall visual quality of the

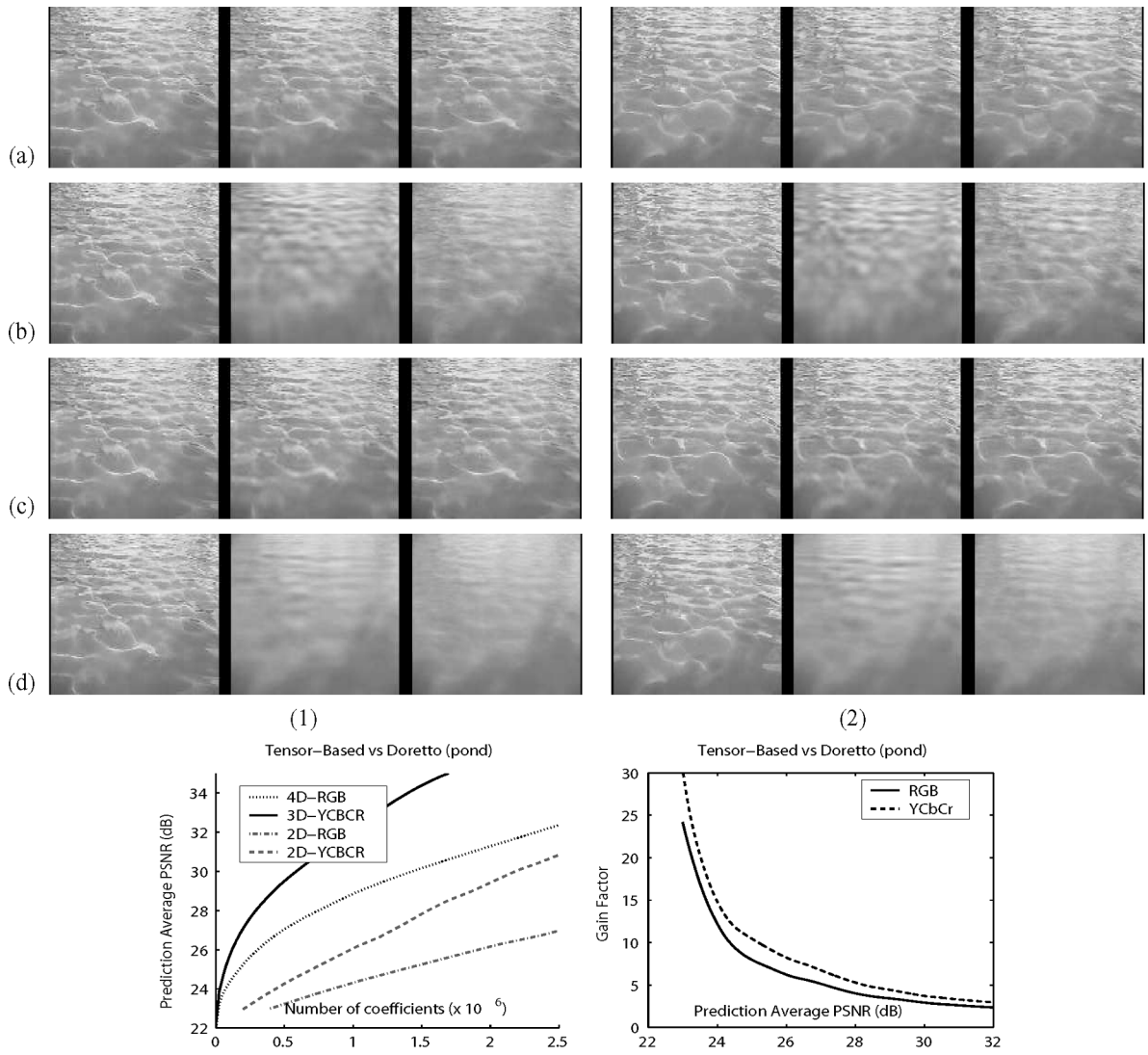


Fig. 8. Performance comparison between the tensor-based and Doretto's model using the test video "Pond." Images from (a)–(d) report the result of the synthesis; each image contains three parts; left: the original; center: synthesis obtained using the tensor-based mode; right: synthesis obtained using Doretto's model. Images (a) and (b) have been obtained ensuring a prediction error of 31 and 24 dB, respectively, by using the tensor-based model on a 4-D RGB data representation and the standard 2-D RGB Doretto's model. Images (c) and (d) have been obtained ensuring a prediction error of 30 and 22.5 dB, respectively, by using the tensor-based model on a 3-D YC_bC_r data representation and the modified Doretto's algorithm (indicated in the paper as 2-D YC_bC_r). Images in columns (1)–(3) have been obtained from the second and the 11th frames of the video sequence "Pond." The last row reports the comparison between models' performance. The results have been obtained using both data representations. The parameters of the tensor-based decomposition are chosen to ensure the best performance; at left, we report the prediction error versus the model size, while, at right, we report the gain in terms of model coefficients, defined as the ratio between Doretto's and tensor-based model's coefficients.

synthetic videos is comparable. The synthetic videos obtained at 22.5 dB show a low-quality synthesis for individual frames, but the video dynamic is comparable to the one obtained at higher PSNR. In other words, the spatial approximation is visually worse than the temporal one.

The gain curves show the comparisons between models using the same color encoding. We can notice that, in general, YC_bC_r color encoding offers a better performance than RGB. This is true for both Doretto's method and the FFT-based method. The video sequence "Grass" is different since, as pointed out before, it is characterized by high spatial frequency content. A large amount of spatial singular values is needed to represent it efficiently, thus resulting in a large model size.

VI. Conclusion

We propose to decompose the multidimensional signal that represents a dynamic texture by using a tensor decomposition technique. As opposed to techniques that unfold the multidimensional signal on a 2-D matrix, our method analyzes data in their original dimensions. This decomposition, only recently used for applications in image and video processing, permits to better exploit the spatial, temporal, and chromatic correlation between the pixels of the video sequence, leading to an important decrease in model size.

Compared to algorithms where the unfolding operations are performed in 2-D or where the spatial information is exploited

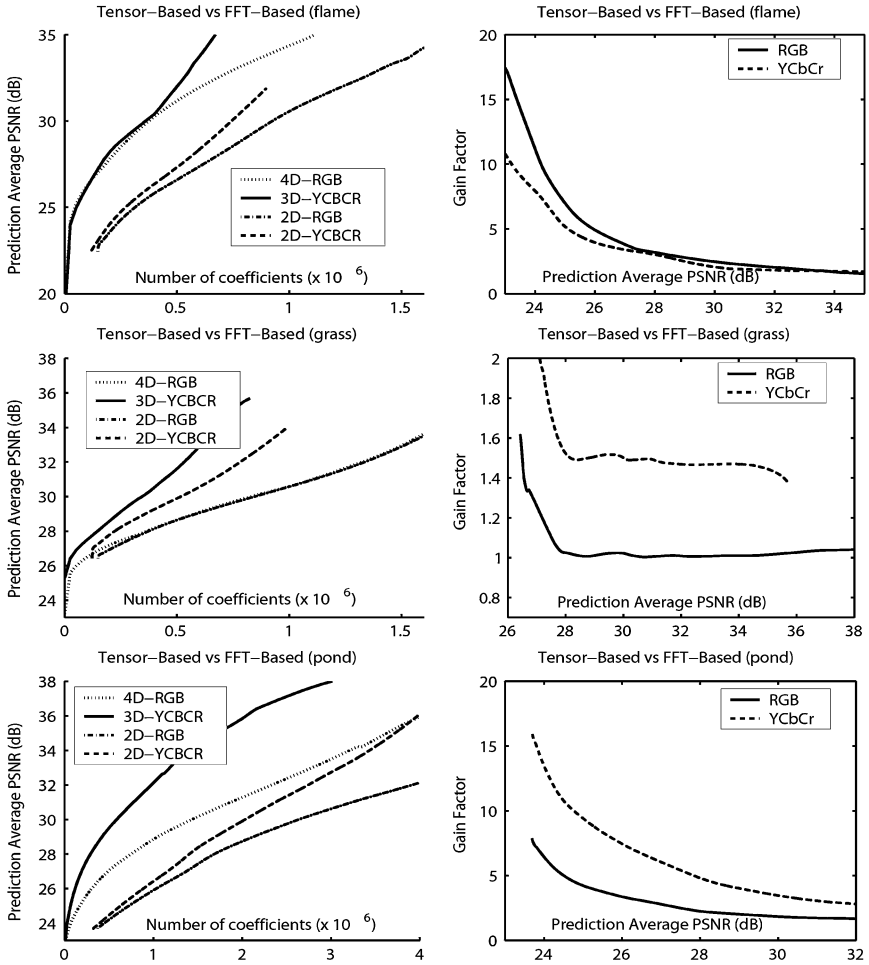


Fig. 9. Performance comparison between tensor and FFT-based's models obtained using the test video sequences at disposal. We report for each video sequence two evaluations: at left, the prediction error versus the model size, both obtained using the tensor-based algorithm on 4–D RGB and 3-D YC_bC_r data representation and the FFT-based algorithms on RGB and YC_bC_r color encoding (indicated as FFT-RGB and FFT-YC_bC_r in the paper); at right, the gain in terms of model coefficients between the two models, i.e., the the ratio between FFT's and tensor-based model's coefficients. For each video sequence, the parameters of the two models have been chosen to ensure the best individual performance.

by considering the analysis in the Fourier domain, our method results in models with on average five times less coefficients, still ensuring the same visual quality.

Despite being a suboptimal solution for the tensor decomposition, the HOSVD ensures close-to-optimal energy compaction and approximation error. The suboptimality derives from the fact that the HOSVD is computed directly from the SVD, without using expensive iterative algorithms, such as done for the optimal solution. This is an advantage, since the analysis can be done faster and with less computational power.

The few model parameters permit to perform synthesis in real-time. Moreover, the small memory occupancy favors the use of the HOSVD based model in architectures characterized by constraints in memory and computational power complexity, such as PDAs or mobile phones.

ACKNOWLEDGMENT

The authors would like to thank C. Carmeli and J. Vandewalle for fruitful discussions on the use of tensor decomposition.

REFERENCES

[1] G. Doretto, A. Chiuso, Y. Wu, and S. Soatto, "Dynamic textures," *Int. J. Comput. Vis.*, vol. 51, no. 2, pp. 91–109, 2003.

[2] J. Stam and E. Fiume, "Depiction of fire and other Gaseous phenomena using diffusion processes," in *Proc. ACM SIGGRAPH*, 1995, pp. 129–136.

[3] V. Pegoraro and S. Parker, "Physically-based realistic fire rendering," in *Proc. Eurographics Workshop on Natural Phenomena*, 2006, pp. 51–59.

[4] A. Schödl, R. Szeliski, D. Salesin, and I. Essa, "Video textures," in *Proc. ACM SIGGRAPH*, 2000, pp. 489–98.

[5] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick, "Graphcut textures: Image and video synthesis using graph cuts," in *Proc. SIGGRAPH*, 2003, pp. 277–286.

[6] G. Doretto, D. Cremers, P. Favaro, and S. Soatto, "Dynamic texture segmentation," in *Proc. IEEE Int. Conf. Image Processing*, 2003, pp. 1236–1242.

[7] P. Saisan, G. Doretto, Y. N. Wu, and S. Soatto, "Dynamic texture recognition," in *Proc. IEEE CVPR*, 2001, vol. 2, pp. 58–63.

[8] G. Doretto and S. Soatto, "Editable dynamic textures," in *Proc. IEEE CVPR*, 2003, vol. 2, pp. 137–142.

[9] S. Soatto, G. Doretto, and Y. N. Wu, "Dynamic textures," in *Proc. IEEE ICCV*, 2001, vol. 2, pp. 439–46.

[10] R. Costantini, L. Sbaiz, and S. Susstrunk, "Dynamic texture analysis and synthesis using tensor decomposition," *Lecture Notes Comput. Sci.*, vol. 4292, pp. 1161–1170, 2006.

[11] L. De Lathauwer, B. De Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM J. Matrix Anal. Appl.*, vol. 21, no. 43, pp. 1253–1278, 2000.

[12] G. Doretto and S. Soatto, "Dynamic shape and appearance models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 12, pp. 2006–2019, Dec. 2006.
[13] M. A. O. Vasilescu and D. Terzopoulos, "TensorTextures: Multilinear image-based rendering," in *Proc. ACM SIGGRAPH*, 2004, pp. 336–342.
[14] W. Liu, D. Lin, and X. Tang, "Hallucinating faces: TensorPatch superresolution and coupled residue compensation," in *Proc. IEEE CVPR*, 2005, vol. 2, pp. 478–84.
[15] M. A. O. Vasilescu and D. Terzopoulos, "Multilinear analysis of image ensembles: TensorFaces," *Lecture Notes Comput. Sci.*, vol. 2350, pp. 447–60.
[16] B. Abraham, O. I. Camps, and M. Sznaier, "Dynamic texture with fourier descriptors," in *Proc. 4th Int. Workshop Texture Analysis and Synthesis*, 2005, pp. 53–58.
[17] R. Henrion, "N-way principal component analysis. Theory, algorithm and applications," *Chemometr. Intell. Lab. Syst.*, vol. 25, pp. 1–23, 1994.
[18] L. Tucker, *Implication of Factor Analysis of Three-Way Matrices for Measurement of Change*. Madison, MI: Univ. Wisconsin Press, 1963.
[19] L. De Lathauwer, B. De Moor, and J. Vandewalle, "On the best rank- 1 and rank-$(R_1, R_2, \ldots, R_N)$ approximation of higher-order tensors," *SIAM J. Matrix Anal. Appl.*, vol. 21, no. 43, pp. 1324–1342, 2000.
[20] M. Schwab, N. Karrenbach, and J. Claerbout, "Making scientific computations reproducible," *Comput. Sci. Eng.*, vol. 2, pp. 61–67, 2000.
[21] R. Costantini. [Online]. Available: http://lcavwww.epfl.ch/reproducible- research/CostantiniIP07/index.html
[22] R. Costantini, L. Sbaiz, and S. Süsstrunk, "Dynamic texture synthesis: Compact models based on luminance-chrominance color representation," in *Proc. IEEE Int. Conf. Image Processing*, 2006, pp. 2085–2088.
[23] B. W. Bader and T. G. Kolda, "MATLAB tensor classes for fast algorithm prototyping," *ACM Trans. Math. Softw.*, vol. 32, no. 4, 2006.

Roberto Costantini (S'03–M'07) received the "Laurea in Ingegneria Elettronica" M.S. degree from the Dipartimento di Elettrotecnica, Elettronica ed Informatica, University of Trieste, Italy, in 2000, for work on low-complexity compression of video for telesurveillance applications, conducted in collaboration with the Institut de Microtechnique (IMT) de Neuchâtel, Switzerland, and the Ph.D. degree on static and dynamic texture modeling from the Images and Visual Representation Group (IVRG), Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, in 2007.

He is a Research Associate at the IVRG, School of Information and Communication Sciences (I&C), EPFL. Between 2000 and 2002, he was an Assistant Researcher at IMT and then moved to IVRG-EPFL. His main research areas are computer vision, digital photography, stochastic processes, time series analysis, and dynamic texture synthesis. In 2004, he worked on machine vision algorithms at Logitech Labs, Fremont, CA, where he pursued the development of webcam video effects.

Luciano Sbaiz (M'98) received the "Laurea in Ingegneria" degree in electronic engineering and the Ph.D. degree from the University of Padova, Padova, Italy, in 1993 and 1998, respectively.

Between 1998 and 1999, he was Postdoctoral Researcher at the Audiovisual Communications Laboratory at Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, where he conducted research on the application of computer vision techniques to the creation of video special effects. In 1999, he joined Dartfish, Ltd., Fribourg, Switzerland, as Project Manager. Within the company, he developed video special effects for television broadcasting and sport analysis. In 2004, he took a position as Senior Researcher at the Audiovisual Communications Laboratory, EPFL, where he conducts research on signal processing. His activities are in the field of image and audio processing, superresolution techniques, and acoustics.

Sabine Süsstrunk (M'02) received the B.S. degree in scientific photography from the Swiss Federal Institute of Technology (ETHZ), Zurich, Switzerland, the M.S. degree in graphic arts publishing from the Rochester Institute of Technology, Rochester, NY, and the Ph.D. degree from the School of Computing Sciences at the University, East Anglia, Norwich, U.K.

She is a Professor of images and visual representation at the School of Information and Communication Sciences (I&C), Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland. Her main research areas are digital photography, color image processing, image quality metrics, and digital archiving. From 2003 to 2004, she was a Visiting Scholar at Hewlett-Packard Laboratories, Palo Alto, CA. Prior to the EPFL, she was a Principle Imaging Researcher at Corbis Corporation, Seattle, WA, and an Assistant Professor with the School of Photographic Arts and Sciences, Rochester Institute of Technology.

Dr. Süsstrunk is Associate Editor of the IEEE TRANSACTIONS ON IMAGE PROCESSING and a member of IS&T, OSA, and ACM.