Modeling of Texture Movies for Video Coding

S. Valaeys†, G. Menegaz‡, J. Reichel† and F. Ziliani†

†VisioWave S.A.
ECUBLENS, Switzerland
E-mail: stephane.valaeys@a3.epfl.ch
julien.reichel/francesco.ziliani@visiowave.com

‡Audiovisual Communications Laboratory
EPFL, Switzerland
E-mail: gloria.menegaz@epfl.ch

Abstract— **We propose a novel model-based coding system for video. Model-based coding aims at improving compression gain by replacing the non-informative image elements with some perceptually equivalent models. Images enclosing large textured regions are ideal candidates. Texture movies are obtained by filming a static texture with a moving camera. The integration of the motion information within the generative texture process allows to replace the "real" texture with a "visually equivalent" synthetic one, while preserving the correct motion perception. Global motion estimation is used to determine the movement of the camera and to identify the overlapping region between two successive frames. Such an information is then exploited for the generation of the texture movies. The proposed method for synthesizing *2D+1* texture movies is able to emulate any piece-wise linear trajectory. Compression performances are very encouraging. Compared to MPEG-4 state-of-the-art video CODEC, the proposed method achieves on texture movies compression rates more than 10 times lower with sensibly better perceptual quality. Importantly, the current implementation is real-time on P3 processors.**

I. Introduction

Classical coding techniques for still images and videos are reaching their full potential. A new perspective must be taken to improve coding performances. In this paper, we propose a new *model-based* approach to coding, based on the redefinition of the notion of relevance. The so-called *model-based* approach to coding exploits the semantics of the scene for improving compression performances. For those regions that are not "informative", it is assumed that the relevance resides in the "visual appearance". The real information can thus be replaced by some "visually equivalent" synthetic representation generated by an adequate model. The practical use of such a model within a coding system adds some constrains to the modeling task. The model must be concise (*e.g.* require a relatively small number of parameters) to be competitive towards classical coding techniques. Furthermore, the generative process must be fast to enable real-time applications.

Images and videos containing large amounts of texture are appropriate candidates for model-based coding. Textures are difficult to code by classical schemes based on the exploitation of the spatial, temporal and statistical redundancy. Often they create a bottleneck in the coding chain. Moreover, they usually constitute a non informative part of the scene, so that their replacement by a visually equivalent pattern would not degrade the global perceived quality of the resulting image. Another factor that makes textures particularly appealing for model-based coding is that in the last decade a great research effort has been devoted the field of texture modeling, resulting in a wide number of possible solutions. The strict connections with the investigation of human vision makes it a powerful tool for the determination of the parameters, or perceptual primaries, which are extracted by the visual system and the way they are translated to higher level perceptual units. Besides the pioneering work of Heeger and Bergen [1], particularly relevant in this respect are the contributions of Portilla & Simoncelli [2] and Zhu [3]. These probabilistic texture models have grown on the insights coming from neural sciences. The main guidelines are in the assumption that the visual system responds to the statistics of the stimuli to which it is exposed and processes the visual information in order to maximize the "efficiency" of the representation [4]. The correct match between the derived bases functions with the receptive fields in primary visual cortex [5] have validated such works. An alternative approach aiming primarily at texture replication, as opposed to the identification of the perceptual features, has been followed by other authors. Some noteworthy examples are the solutions proposed by De Bonet [6], [7], Menegaz [8] Efros and Leung [9], Wei and Levoy [10], Xu et al [11] and Ashikhmin [12].

Conversely, the field of "dynamic" texture modeling is relatively under-investigated. Dynamic textures are usually meant as multi-dimensional stochastic processes exhibiting some stationarity over time [13]. Some examples are smoke, waves and foliage. This can be regarded as a generalization of the bi-dimensional case, where temporal evolution is a feature of the global stochastic process. Examples are the solutions proposed by Soatto [13], Bar-Joseph [14] and Szummer [15].

The novelty of our contribution is that we address the problem of modeling a different class of dynamic textures, for which the motion is not an intrinsic property of the considered process, but the result of a continuous change of the observation point of view. We aim at modeling the motion features as perceived by a moving observer. To make the distinction with respect of the 3D dynamic processes mentioned above, we call the considered class *2D+1 Texture Movies (TM)*. In this case, the key point is the preservation of the temporal correlation between subsequent images, or frames. More specifically, we consider here the case of a static texture - the grass - shot by a moving camera. This situation is indeed representative of the typical set of applications we are considering. The synthesis of 2D+1 textures

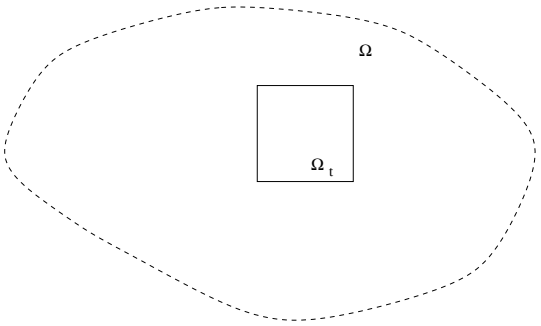

Fig. 1. Sublattice covered by the observation at time t.

is integrated in a wavelet-based coding system and combined with classical compression of the remaining parts of the images.

This paper is organized as follows. Sec. II proposes a simple theoretical presentation of the issue. Sec. III revisits the modeling technique for static textures proposed in [8] and details the specificity of the current implementation. Sec. IV describes the core of the proposed system, namely the movement-simulating algorithm. Sec. V provides some results for all the steps involved, i.e. static and dynamic texture synthesis as well as video coding performances, and Sec. VI derives conclusions.

II. Modeling of 2D+1 texture movies

Probabilistic static texture modeling aims at generating a new image from a sample texture, such that it is *sufficiently different* from the original yet appears to be generated by the *same underlying stochastic process*.

The goal of the proposed algorithm is to generalize such an idea to the generation of a progressively "growing" texture, where the direction and speed of growth is given *a-priori* by a predefined motion model. More specifically, we focus here on piece-wise linear trajectories. In this case, the main issue is the preservation of the perception of motion, namely the preservation of those visual features which determine the sensation of continuity in the texture flow.

It is worth pointing out that the trivial juxtaposition of temporally subsequent patches respectively sampled from successive frames is not a solution. The aliasing phenomena due to the sampling as well as the mismatch between the sampling grids associated to two successive frames would result in a discontinuity along the boundary. Importantly, such sampling artifacts are the origin of the failure of prediction-based coders when applied to texture sequences.

In what follows, we provide a simple formalization of the proposed solution.

Let Ω be the infinite lattice, and let Ω_t be the domain which is observed at time t, *e.g.* the spatial support associated with the observation at a given instant in time, as illustrated in fig. 1. Let then $I(\Omega_t)$ be the observation at time t and $\tilde{I}(\Omega_t)$ be the synthetic counterpart. Clearly:

$$\Omega_t \subset \Omega \quad \forall t \tag{1}$$

Accordingly, Ω_{t_1} and Ω_{t_2} denote the domains covered by the observations at times t_1 and t_2, respectively. The specificity of the proposed approach is that it provides a solution to the following problem:

Given two sub-lattices Ω_{t_1} and Ω_{t_2} such that:

$$\Omega_{t_1} \cap \Omega_{t_2} = \Omega_{\Delta t} \neq \emptyset, \tag{2}$$

generate a synthetic texture over Ω_{t_2} by *growing* it from the seed already present on $\Omega_{\Delta t}$ such that the impression of visual continuity is preserved.

If the two sets were disjoint, then the independent generation of the texture over the two domains would have been adequate. Conversely, where there is an overlap between the two domains, the independent generation of the texture would produce an apparent edge at the boundary or, equivalently, a flickering on the representation as a temporal sequence which destroys the continuity of the visual flow.

The key feature of the proposed model is the ability to synthesize a textures $\tilde{I}(\Omega_{t_2})$ over the domain Ω_{t_2} by growing the texture over $\overline{\Omega}_{\Delta t} = \Omega_{t_2} \backslash \Omega_{\Delta t}$ but keeping unchanged the texture already present over $\Omega_{\Delta t}$ and avoiding discontinuities along the boundary.

The previous discussion can be generalized to the case where the observations are themselves realizations of the stochastic process represented by the considered model for static textures. This is indeed the case when focusing on video coding. In such a framework, the idea is to transmit the model of the texture to reproduce, and use it to generate the texture all along the sequence. [1]

In this case, the following relation holds:

$$\tilde{I}(\Omega_{t+\Delta t}) \quad = \quad \tilde{I}(\Omega_t) \oplus \tilde{I}(\overline{\Omega}_{\Delta t}) \tag{3}$$

where $\tilde{I}(\Omega_{t+\Delta t})$ is the synthetic texture simulating the observation at time $t + \Delta t$, $\tilde{I}(\Omega_{\Delta t})$ is the texture seed, and where the operator $\oplus$ indicates the juxtaposition of the textures stated.

The spatial position of $\Omega_{t+\Delta t}$ can be easily recovered from the underlying motion model. Let $x, y \in \mathbb{R}$ be the spatial coordinates of the upper left corner of Ω_t and let h and w, with $h, w \in \mathbb{R}_*^+$, be the height, respectively the width, of the spatial domain Ω_t, assumed to be of rectangular shape. Given the estimated speed $\overrightarrow{v} = (v_x, v_y)$ at which the point of observation moves, it is straightforward to derive the position of the domain $\Omega_{t+\Delta t}$ concerned by the observation at time $t + \Delta t$ as the one whose upper left corner has coordinates:

$$x + \Delta x \quad = \quad x + v_x \cdot \Delta t \tag{4}$$

$$y + \Delta y \quad = \quad y + v_y \cdot \Delta t \tag{5}$$

Therefore, one can easily identify $\Omega_{\Delta t}$ and $\overline{\Omega}_{\Delta t}$. However, growing $\tilde{I}(\overline{\Omega}_{\Delta t})$ from the seed covering $\Omega_{\Delta t}$ is not trivial. We refer to Sec. IV for further details

[1] Of course this only holds when assuming that the texture is homogeneous. A simple generalization would be to assume that a model for an homogeneous texture would be suitable as long as the model parameters are contained within a predefined region of the feature space. We leave this subject for further investigation.

III. DWT-based Multiresolution Probabilistic Texture Modeling

Static texture synthesis is achieved using the Discrete Wavelet Transform-based Multiresolution Probabilistic Texture Modeling (DWT-MPTM) method proposed in [8]. The model consists in an implicit (non-parametric) representation and reproduction of intra- and inter-scale dependencies between DWT multiresolution coefficients. These dependencies are captured and regenerated using the multiresolution conditional sampling technique. This is a direct sampling from a distribution by a Parzen estimator. Since the theoretical link between the size and shape of the Parzen window and the quality of the perceptual features of the synthesized texture is still to be established (apart from the well-known impact of the window width), the size of the window is set empirically and is allowed to vary to ensure it that it englobes a "sufficient" amount of samples (we refer to the Appendix for further explanations).

Linking model parameters to perceptual features is a very difficult problem. Even though many statistical parameters have been proposed by different authors [1], [2], [3] as those relevant to texture in terms of "visual appearance", the way each of such parameters maps to perception and to higher visual cues is still unknown. Namely, the gap between low-level parameters and mid- to high-level visual features is still unsolved. The fact that non-parametric models, such as the DWT-MPTM, which are based on a decomposition too simple to be representative of the one performed by the visual system, provide results comparable in quality to other parametric state-of-the-art techniques suggests some redundancy in the more complex representations at least as far as perceptual features are concerned.

The algorithm that inspired the DWT-MPTM [6] generates a new texture image by shuffling coefficients of the original texture at different spatial resolutions. This shuffling is constrained in two ways:

- The local characteristics inside a frequency band must be preserved
- Coefficients in a frequency band depend on the corresponding ones at lower resolution

An *analysis pyramid* is built using a Laplacian pyramidal formulation [6]. A *conditioning pyramid* is built by applying a set of orientation selective filters to each level of the analysis pyramid. The conditioning pyramid describes the local properties of each frequency band. A *synthesis pyramid*, the counterpart of the analysis pyramid for the synthetic texture, is then filled with coefficients obtained by conditional sampling on the corresponding levels of the analysis pyramid.

The distinguishing feature of the approach proposed in [8] is that a single *hyper-pyramid*, the DWT hyper-pyramid, combines the functions of both the analysis and the conditioning. The algorithm is illustrated in fig. 2. The three detail subbands of the DWT are used as the local descriptors of the frequency bands. Accordingly, the synthesis pyramid is also a DWT hyper-pyramid.

In order to reproduce the characteristic structure of a texture, the statistical intra and interscale relationships observed in the original sample between coefficients must be preserved [2]. In that regard, we define the terms of *parent*, *tree* and *parent vector* as follows:

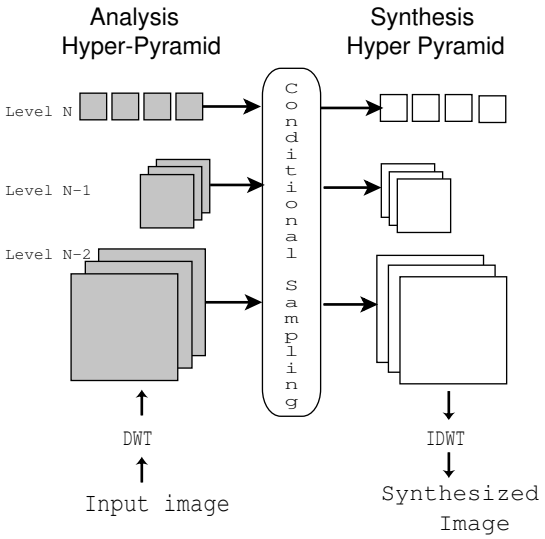

Fig. 2. Principles of the DWT MPTM. Three levels of decomposition for the DWT are represented.

- Given the total number N of levels in the pyramid, let $F_i^j(x,y)$ be the coefficient of coordinates (x,y) of the feature image F_i^j of level $0 < i < N$ and orientation $0 < j \leq 3$, $i, j \in \mathbb{N}$. Then, the *parent* of this coefficient is the sample $F_{(i+1)}^j(\lfloor \frac{x}{2}, \frac{y}{2} \rfloor)$. By extension, we say that the former coefficient is the *son* of the latter.
- Inspired by the wavelet quad-tree, we call *tree* any group of coefficients in a pyramid linked by a parent-son relationship throughout the levels.
- The parent vector of the coefficient $F_i^j(x,y)$ is the vector

$$\begin{aligned}\overrightarrow{V}_i^j(x,y) = \big[&F_{i+1}^1(\lfloor\frac{x}{2},\frac{y}{2}\rfloor), F_{i+1}^2(\lfloor\frac{x}{2},\frac{y}{2}\rfloor), F_{i+1}^3(\lfloor\frac{x}{2},\frac{y}{2}\rfloor), \ldots,\\ &F_N^1(\lfloor\frac{x}{2^{N-i}},\frac{y}{2^{N-i}}\rfloor), F_N^2(\lfloor\frac{x}{2^{N-i}},\frac{y}{2^{N-i}}\rfloor),\\ &F_N^3(\lfloor\frac{x}{2^{N-i}},\frac{y}{2^{N-i}}\rfloor)\big]\end{aligned} \tag{6}$$

where the notation is the same as above.

The parent vector holds all the information necessary for conditioning: local features of coarser levels as well as inter-level dependencies. Conditional sampling consists of two successive steps:

1. Build a candidate set $C_i^j(x,y)$ for the coefficient $F_i^j(x,y)$;
2. Randomly sample from $C_i^j(x,y)$

The candidate set $C_i^j(x,y)$ contains all coefficients $F_i^{j,a}(x',y')$ of the corresponding analysis pyramid subband which have "similar" parent vectors $\overrightarrow{S}_i^j(x',y')$:

$$C_i^j(x,y) = \{(x',y') : D(\overrightarrow{V}_i^j(x,y), \overrightarrow{S}_i^j(x',y') \leq \overrightarrow{T}_i\} \tag{7}$$

In (7), $\overrightarrow{T}_i$ is a threshold vector:

$$\overrightarrow{T}_i = [T^1_{i+1}, T^2_{i+1}, T^3_{i+1}, \ldots, T^1_N, T^2_N, T^3_N] \quad (8)$$

and D(·) represents the component-wise absolute difference operator (information on the selection of the threshold vector can be found in the Appendix). Accordingly, condition (8) can equivalently be written in the form:

$$|V^j_{i,k}(x,y) - S^j_{i,k}(x',y')| \leq T_{i,k} \quad (9)$$

where the subscript k indicates the k^{th} component of the respective vectors. Once the candidate set $C^j_i(x,y)$ is determined, one of its member is randomly chosen and its value is assigned to the coefficient $F^j_i(x,y)$ of the synthesis pyramid.

The synthesis process starts by filling the highest level of the synthesis hyper-pyramid, namely the three detail subbands of level N and the approximation, by copying the corresponding subbands of the analysis pyramid. Successive finer levels are then filled by conditional sampling.

A. Conditional Tiling in Transform Space

Results show that in many cases the structure of the texture is better preserved by limiting the sampling to the higher levels of the pyramid. In this particular implementation of the DWT-MPTM, conditional sampling is limited to the level $N-1$. Hence, subbands of the level N of the synthesis pyramid are copied from the analysis one, those of level $N-1$ are filled through conditional sampling while the remaining levels are filled by retaining the entire trees emerging from the last synthesized samples.

As explained in the Appendix, groups of coefficients are sampled at once to improve the efficiency of the algorithm while better preserving texture characteristics. Because they share the same parent vector, coefficients of coordinates $(x,y),(x+1,y),(x,y+1)$ and $(x+1,y+1)$, where x and y are even numbers, of all three subbands of level $N-1$ are sampled together.

The two ideas explained above, which are crucial to making the DWT-MPTM fast enough for real-time applications, influence the randomness of the algorithm. In fact, a careful analysis shows that implemented in such a way, the DWT-MPTM is reduced to what corresponds to a tiling of 2^N by 2^N blocks at image level. We speak of Conditional Tiling in Transform Space (CTTS). The advantages of CTTS over conventional tiling are twofold. Because it takes place in transform space, and because tiles are conditionally positioned next to one another, feature mismatches and block effects are greatly reduced. This enables the use of smaller tiles than would be possible at image level, which reduces the chance of apparent replica of the original sample.

B. Generating Pictures of Any Size

The guideline of the DWT-MPTM algorithm is to recreate parental relationships between pixels of different levels.

The generative process starts by copying the highest level of the analysis pyramid in the synthesis pyramid. However, when synthesizing a larger image, the original subbands are not enough to fill completely their synthetic counterpart. In [8] and [6], subbands are scaled or tiled to cover the desired area. Tiling is feasible only if the depth of the decomposition is sufficiently high to produce strongly homogeneous subbands. Ideally, these subbands should contain just one pixel [6]. Since in practical applications this is usually not the case, tiling would then generate blocking artifacts. For example, if a texture illuminant were not constant but fading from right to left, tiling would be visible and the resulting output image would be vitiated.

To solve this problem, one idea is to recursively fill empty coefficients at the border of those filled by values which are close to that of their filled neighbor. In doing so, we assume that those subbands have attained a certain level of smoothness. One advantage of this method is that the pool of coefficients somewhat like the filled neighbor is already computed for conditional sampling. This operation therefore requires no additional computation.

IV. Incorporation of the DTW-MPTM in a movement-simulating algorithm

As stated in Sec. II, the specificity of the proposed approach is to integrate the DWT-MPTM in a solution to the following problem:

Given two sub-lattices Ω_{t_1} and Ω_{t_2} such that:

$$\Omega_{t_1} \cap \Omega_{t_2} = \Omega_{\Delta t} \neq \emptyset,$$

generate a synthetic texture over Ω_{t_2} by *growing* it from the seed already present on $\Omega_{\Delta t}$ such that the impression of visual continuity is preserved.

We propose a method based on synthesizing a texture area larger than the video frame size, preserving the texture over $\Omega_{\Delta t}$ while generating a limited amount of new texture, only when necessary, to cover $\overline{\Omega}_{\Delta t}$ without creating apparent discontinuities.

It is worth pointing out that the straightforward solution of synthesizing each frame independently with the DWT-MPTM is not suitable because it creates a disjointed succession of rapid texture changes that fails to generate an impression of movement. One also quickly comes to the conclusion that a cut-and-paste approach at image level, in which the common section is correctly displaced and remaining empty parts of the frame are filled with newly synthesized patches of texture, creates unacceptable discontinuities.

Another trivial solution would be to synthesize a much larger texture area than the frame size and to select the covered domain to be part of the frame according to the camera movement. This method is however subject to application concerns. First, the required size of the synthetic texture should be known *a-priori*. Moreover, large amounts of texture could be produced without ever being needed.

A way to answer those concerns is to work in feature space. Although the DWT used for compression purposes is in general not shift-covariant, covariance properties hold for translations in transform space which correspond to translations at image level which can be broken down in

horizontal and vertical shifts of $k \cdot 2^N$ and $h \cdot 2^N$ pixels respectively, where $k, h \in \mathbb{Z}$ and N is the number of decomposition levels of the DWT. Working in feature space, we are consequently able to generate from a synthetic image S the following set:

$$\Gamma = \{S_{(k,h)}, k, h \in \mathbb{Z} | S_{(k,h)} = T_{(k,h)} I\} \tag{10}$$

where $T_{(k,h)}$ is the translation operator that applied to S produces a shift of $k \cdot 2^N$ and $h \cdot 2^N$ units in the horizontal and vertical directions, respectively.

As the translation from S to $S_{(k,l)}$ takes in fact place in feature space, the remaining empty parts of $S_{(k,l)}$ can be filled in feature space by applying the DWT-MPTM algorithm locally without creating discontinuities.

Considering the above, any random translation can be obtained by extending the size of S so as to add to it a border of 2^N pixels on all sides. Therefrom, simulating a random translation is a two-step process: obtaining the correct $S_{(k,h)}$; selecting the correct area of $S_{(k,h)}$ which is to make up the video frame. An example is shown in fig. 3. Let p and q, with $p, q \in \{0, 2 \cdot 2^N\}$ be the width in pixels of the border zone respectively in the horizontal and vertical direction of movement. To generate a horizontal, respectively vertical, movement of m, respectively n, pixels at image level, with $m \geq p$ corresponding to Δx and $n \geq q$ corresponding to Δy in Sec. II, the correct $S_{(k,h)}$ is chosen so that:

$$k = \min_{\tilde{k}} \{(p + \tilde{k} \cdot 2^N) \geq m\} \tag{11}$$

$$h = \min_{\tilde{h}} \{(q + \tilde{h} \cdot 2^N) \geq n\} \tag{12}$$

The window of visibility is then correctly positioned inside $S_{(k,h)}$, namely:

$$new\_p = p + k \cdot 2^N - m \tag{13}$$

$$new\_q = q + h \cdot 2^N - n \tag{14}$$

Proceeding in such a way, we avoid both having to know beforehand the amount of texture needed and generating large useless amounts of it.

Another aspect of the procedure is that, for obvious memory concerns, synthesized textures are not saved once they aren't needed anymore. This leads to the regeneration of texture in the case of movement in one direction then in the opposite one. However, because we deal with highly stochastic textures, subsequent changes are in our opinion very difficult to discern.

V. Results and Discussion

The correct way for evaluating the performances of any texture model, either static or dynamic, would be to design an *ad-hoc* subjective test characterizing it from a perceptual point of view. However, this task is beyond the scope of this contribution, so we leave it an open issue for future work. Instead, we did some informal subjective tests involving people of the laboratory who were not familiar with the subject, as well as some objective measures on both the motion information and the compression performances.

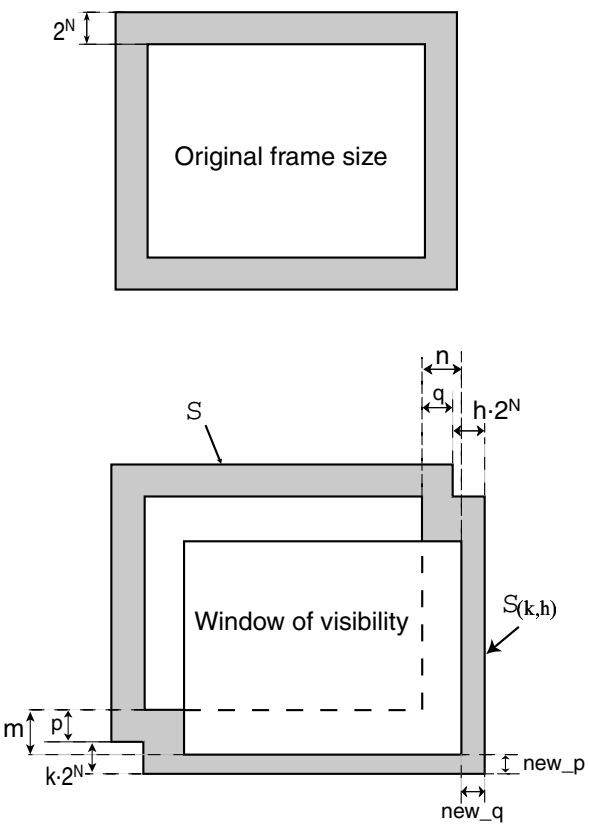

Fig. 3. Simulation of movement. The size of S is bigger than that of the original. When the added border is not enough to simulate the required movement, a shifted version of S is created. The window of visibility is then moved inside $S_{(k,h)}$ to reproduce the correct movement.

As the DWT-MPTM, which has been taken as the ground for our algorithm, has already been extensively tested [8], here we provide only a few examples of its performances. Fig. 4 shows the original (left column) and the synthesized (right column) images for two structured textures from the Brodatz album [16]. In general, results show that the method performs well on a wide variety of random and structured textures.

As mentioned throughout this paper, care has been devoted to attain a high level of efficiency. It is worth pointing out that entire frames of texture rarely have to be generated. In fact, for most frames, no new texture is synthesized at all. This is due to the added border, which can be interpreted as a buffer. When it is not enough, only a small percentage of the image size needs to be newly synthesized. Three cases must therefore be examined. If a scene change has been detected, an entire new frame of texture based on the sample sent must be generated. It takes around 45 milliseconds for a frame of 432×352 pixels to be generated from a 128×128 sample. When movement is such that a portion of new texture must be generated, the synthesis process takes about 15 milliseconds. Finally, the process is instantaneous (0.002 ms) for most frames where the win-

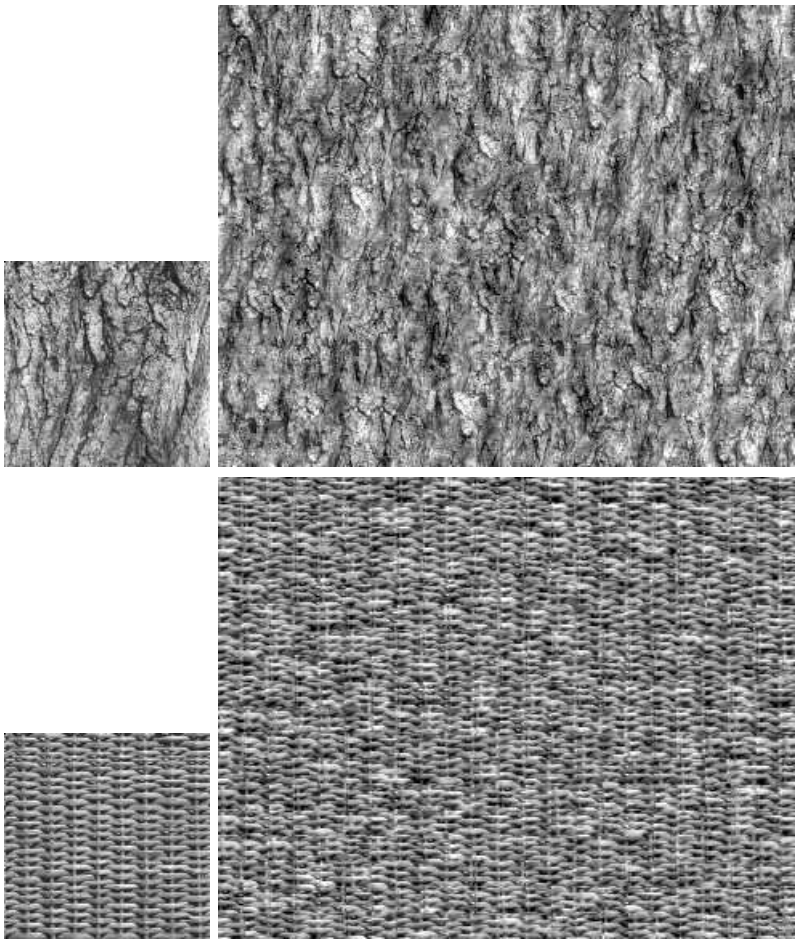

Fig. 4. Results for different types of texture.

dow of visibility must simply be adjusted.

Our texture movie synthesis model is particularly suitable for the integration within a coding system. First, it is entirely based on the DWT, which is the transformation exploited for coding, and classical motion estimation techniques [17]. Second, the implementation by the lifting scheme [18] has a number of features that are particularly adapted for the implementation on a device: it enables the in-place processing as well as the use of integer arithmetic [19], it simplifies the management of border conditions and reduces the complexity by up to a factor 4 [20]. Furthermore, the model is very concise and the synthesis algorithm is computationally efficient (single-step).

We therefore built a complete codec to measure the compression ability of our proposed method. The codec focuses on movies containing only texture with the intent of adding segmentation tools in the future to deal with video sequences containing large amounts of texture, as is often the case for sports sequences (football, rugby, ...). Let us point out right away that in regard to compression, synthesizing a larger picture than the one displayed is not a problem since no extra information is needed. Compression rates are consequently unaffected.

The first results concerning compression performances are promising[2]. The total rate corresponding to the proposed model-based system, *e.g.* needed to encode the model itself and the approximation subbands of the chromaticities for a 2D+1 grass texture, is of 104 kbits/s. It is worth pointing out that the decoding process works in real-time (40 frames per second) on Pentium III processors. By comparison, the minimum rate attainable by a state-of-the-art MPEG4 [21] encoder on the same sequence was more than ten times higher, namely 1408 kbits/s.

Even though the movement of the camera is reduced to simple translations, predictive coders like MPEG4 struggle with such texture sequences. Indeed, the analog texture is captured and sampled by the digital recording device. Because of the high frequency content of the texture, the conditions required for perfect reconstruction of the signal by the Nyquist theorem are not fulfilled and aliasing occurs. Moreover, the limited acquisition speed of the camera introduces additional frequency smoothing. Successive frames in the original movie are therefore not an exact translated version of one another. For the same reason, it is not possible to encode with a lossless method the extra amount of texture needed from one frame to another and "paste" it alongside the previous frame without creating discontinu-

[2]A wide set of synthesized as well as MPEG4 decoded sequences is available at `http://www.visiowave.com/gmbv2002/`.

ities. This leads us to believe that our codec fills a gap in the range of coding techniques.

The ability of the proposed algorithm to reproduce the correct temporal correlation between successive frames or, equivalently, the motion information present in the original sequence, has been tested by comparing the motion vector field estimated on the original sequence (fig. 5(a)) with that obtained by applying the motion estimation algorithm to the synthesized sequence (fig. 5(b)). As a counter-example, fig. 5(c) shows the motion vector field for the case where each frame was generated independently. The similarity between fig. 5(a) and (b), as opposed to the random distribution of the vectors in (c) proves that the proposed method is able to preserve the temporal link between the frames of the sequence and, consequently, to induce the same motion *perception* in the human observer. It is worth mentioning that the vectors on the right of fig. 5(b) simply translate the introduction of new texture that cannot be matched to previous blocks. This has been also verified by an informal test on some non-trained observers. All could correctly identify the situation and provided the same description of the dynamic scene for both the original and the synthesized test sequences.

Another very interesting feature of the synthesized sequence is that the perceptual quality of the frames is the same all over the sequence, for a given rate, namely, without an additional cost in terms or resources. This has a direct impact on the applications, enabling for example the pause with same quality on any image. Furthermore, frame-by-frame analysis shows that because of the movement and the limited capabilities of the acquisition device, many frames of the original sequence seem completely out of focus. These artifacts are no more present in the synthesized sequence, which also looks less noisy, jittery and blurry than the original (see fig. 6).

A different choice would be to exploit the degradation of the performances of the visual system to perceive high spatial frequency in presence of motion. This would further simplify the synthesis process at the expenses of the frame-wise resolution.

A major shortcoming of the present codec is its inability to render perspective. Because of this, most existing video texture sequences, like movies of football games for example, cannot be correctly processed. In the future, we hope to resolve this problem by filtering the texture according to its position in the frame. To complete the codec, other types of camera movement must also be added, like zooms and rotations. Rotations could be dealt with as a sum of infinitesimal translations.

VI. Conclusion

We propose a novel model-based coding system for video. The generalization of the DWT-MPTM modeling method for static textures to 2D+1 moving textures results in an algorithm able to synthesize any 2D+1 texture movie with any piece-wise linear trajectory. A representative texture patch is extracted from a frame of the original sequence and is used as model for synthesizing the other frames. The motion vector field is estimated at any frame and is used to constrain the generating process such that the correct temporal correlation between the images is preserved. The global model-based video coding system is highly computationally efficient. The current implementation is real-time on P3 processors. Furthermore, first results show that it outperforms the considered MPEG4 encoder in both coding gain and quality of the decoded information when evaluated either framewise or on the entire video. Among the issues deserving further investigation are the emulation of other camera functions like zooming and rotation and the integration of the synthesis procedure within an object-based coding framework.

VII. Appendix

A. Determining the Threshold Vector Values

The threshold values making up the vector $\overrightarrow{T}_i$ represent one of the most sensitive parameter of the algorithm. Roughly, large values do not hold enough discerning power making the resulting candidate set is too vast to be representative of the texture structure. Conversely, small values overconstrain the sampling generating some replica of the model.

The difficulty in finding the optimal threshold values is increased by the fact that they are texture-dependent. Our solution consists in fixing the minimum number C_{min} of candidates and progressively updating the threshold values until such a condition is met. The initial values $\overrightarrow{T}_{i,0}$ used for $\overrightarrow{T}_i$ are the standard deviations of the corresponding analysis subbands:

$$T^j_{i,0} = \sqrt{\frac{1}{n-1}\sum_x \sum_y (F^j_i(x,y))^2} \tag{15}$$

where n is the number of coefficients in the subband F^j_i. A wide range of values has been tested for both T^j_i and C_{min}. (between 0.2 and 1.5 times the standard deviation for the first, and 4 to 32 candidates for the second) but no sensible improvement has been observed.

B. Improving the Speed of the DWT-MPTM

One specificity of the current implementation is that the computational efficiency is sensibly improved with respect to [8]. This is due to the strategy followed for designing and implementing the multiscale conditional sampling.

First, the synthesis pyramid is filled *tree by tree* instead of level by level, *e.g.* by proceeding "vertically" instead of "horizontally" through the pyramid. This is based on the following property:

$$F^{j,a}_{i+1}(\lfloor \frac{x'}{2}, \frac{y'}{2} \rfloor) \in C^j_{i+1}(x,y) \rightarrow F^{j,a}_i(x',y') \in C^j_i(x,y) \tag{16}$$

namely, a coefficient can only belong to the candidate set $C^j_i(x,y)$ if its parent belongs to $C^j_{i+1}(\lfloor \frac{x}{2}, \frac{y}{2} \rfloor)$. However, this is a necessary but not sufficient condition. The candidates for $F^j_i(x,y)$ are the subset of descendants of the

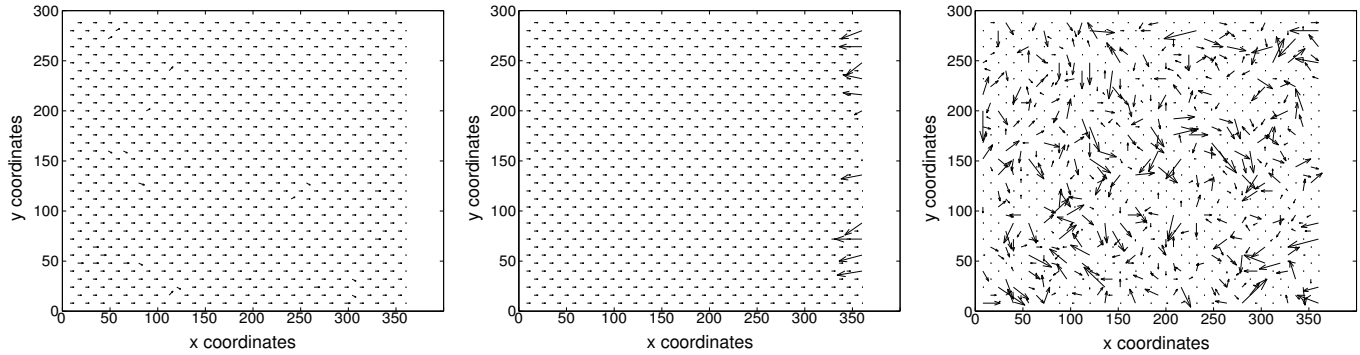

Fig. 5. Motion vector fields as estimated on (a) the original sequence (b) the synthesized sequence (c) the sequence obtained by synthesizing each frame independently.

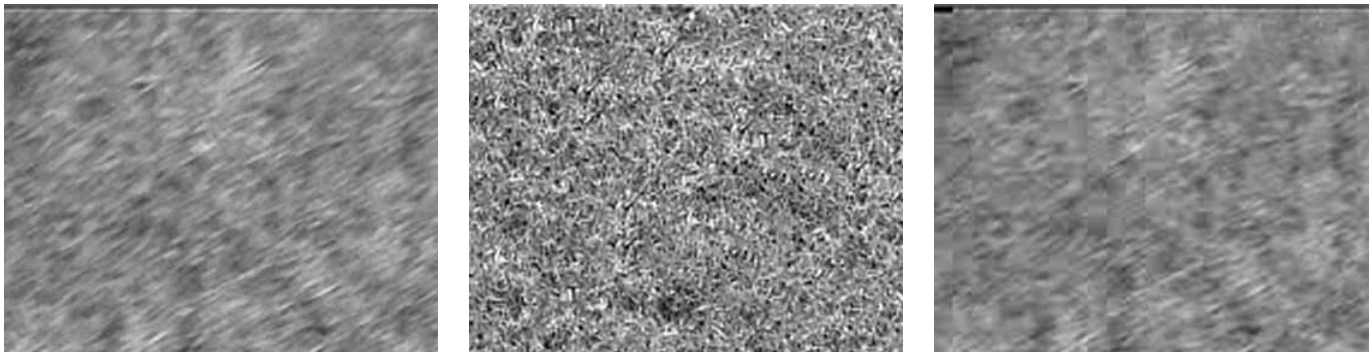

Fig. 6. Some frames of a movie are blurred because of the movement and the limitations of the acquisition device. (a) Original frame; (b) Corresponding synthetic frame; (c) Minimum rate MPEG4 decoded frame.

elements $C^j_{i+1}(x,y)$, $F^j_{i+1}(\frac{x'}{2},\frac{y'}{2})$ satisfying the additional condition:

$$|F^j_{i+1}(\lfloor\frac{x}{2},\frac{y}{2}\rfloor) - F^{j,a}_{i+1}(\lfloor\frac{x'}{2},\frac{y'}{2}\rfloor)| \leq T^j_{i+1} \quad \forall j = 1,\ldots,3 \tag{17}$$

Basically, the candidate set of the "just filled" sample brings all the *a-priori* information needed to build the candidate set for its descendants. The gain in efficiency is due to the (progressive) filling of all the positions within the current tree before, switching to other positions of the current level. Therefore, the synthesis tree is built branch by branch.

Second, the definition of the parent structure allows to sample *groups* of coefficients at once. More specifically: samples of coordinates (x,y) of their respective subbands in a common level share the same parent vector. Subsequently, if a coefficient of the analysis pyramid $F^{1,a}_i(x',y')$ belongs to the candidate set $C^1_i(x,y)$, then $F^{2,a}_i(x',y')$ and $F^{3,a}_i(x',y')$ belong to $C^2_i(x,y)$ and $C^3_i(x,y)$, respectively. We exploit this property by *linking* such coefficients for sampling the subbands of a common level. Instead of randomly choosing three different coefficients for $F^1_i(x,y)$, $F^2_i(x,y)$ and $F^3_i(x,y)$, the three coefficients $F^{1,a}_i(x',y')$, $F^{2,a}_i(x',y')$ and $F^{3,a}_i(x',y')$ are chosen at once and assigned to their respective position. Furthermore, because of the subsampling implied by the DWT, four pixels per subband share the same parent vector. If x and y are both even numbers, then the coefficients of coordinates $(x,\ y), (x,y+\ 1), (x+1,y)$ and $(x+1,y+1)$ of all detail subbands of a common level share the same parent vector. We also choose to link the sampling of those four coefficients. Accordingly, if $F^{j,a}_i(x',y')$ is chosen for $F^j_i(x,y)$, then the three other sons of the same father are assigned the following values:

$$F^j_i(x+1,y) = F^{j,a}_i(x'+1,y') \tag{18}$$
$$F^j_i(x,y+1) = F^{j,a}_i(x',y'+1) \tag{19}$$
$$F^j_i(x+1,y+1) = F^{j,a}_i(x'+1,y'+1) \tag{20}$$

For this to be correct, x, y, x' and y' must all be multiples of 2. If not, coefficients which do not share a common parent vector could be grouped.

In all, twelve coefficients are linked together. It is worth mentioning that besides the gain in complexity (a single candidate set is built for 12 coefficients, and the random function is called only once), the linking tends to strengthen the relationship between coefficients both within and across scales, improving the representation of the texture structure, for a given subband pyramid. Finally, to keep the complexity low, the size of both the original and synthetic images is constrained to be a power of

two.

C. Processing Boundary Samples

In our implementations of the DWT and IDWT, boundaries of the subbands are extended by symmetry. A virtual neighborhood is therefore created to enable convolution. While this does not have an impact for static texture synthesis, problems arise when movement is generated. When coefficients are shifted, some of them which lied at the border of their respective subbands are now surrounded by other pixels. Their neighborhoods have changed. Hence the result of convolution will change, which in turn will affect pixels at image level.

To minimize the artifacts, we use the lifting scheme implementation of the wavelet transform with the (5,3) filter, which has the particularity that each lifting step involves only the current and the previous or following samples. The size of the local neighborhood is then one. Consequently, only the border coefficient of each subband is concerned by changing neighborhoods. But since a coefficient at highest level of decomposition corresponds to 2^N pixels at image level, a border of 2^N pixels at image level is affected by changing values from one frame to another.

To avoid this problem we extend once again the size of the synthesized texture image, so that changing pixels are contained within a border of 2^N pixels which is never visible. Artifacts due to the border effect are always kept out of the actual video frame.

References

[1] D. J. Heeger and J. R. Bergen, "Pyramid-based texture analysis-synthesis," in *Proc. of SIGGRAPH '95*, 1995, pp. 229–238.

[2] J. Portilla and E. P. Simoncelli, "A parametric texture model based on joint statistics of wavelet coefficients," *International Journal of Computer Vision*, vol. 40, no. 1, pp. 49–71, December 2000.

[3] S. Zhu, Y. Wu, and M. Mumford, "Filters, random fields and maximum entropy (frame) - towards the unified theory for texture modeling," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 1996.

[4] E.P. Simoncelli and B.A. Olshausen, "Natural image statistics and neural representation," *Annual Review of Neuroscience*, vol. 24, pp. 1193–1215, 2001.

[5] B.A. Olshausen and D.J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, pp. 607–609, 1996.

[6] J. S. De Bonet, "Multiresolution sampling procedure for analysis and synthesis of texture image," in *Computer Graphics*. ACM SIGGRAPH, 1997, pp. 361–368.

[7] J. S. De Bonet and P. Viola, "A non-parametric multi-scale statistical model for natural images," *Advances in Neural Information Processing*, vol. 10, 1997.

[8] G. Menegaz, "Dwt-based non-parametric texture modeling," in *Proc. of the International Conference on Image Processing (ICIP)*, October 2001.

[9] A.A. Efros and T.K. Leung, "Texture synthesis by non-parametric sampling," in *Proc. of the International Conference on Computer Vision (ICCV)*, Speptember 1999.

[10] Li-Yi Wei and Marc Levoy, "Fast texture synthesis using tree-structured vector quantization," in *Proc. of SIGGRAPH 00*, 2000, pp. 479–488.

[11] B.N. Guo Y.Q. Xu and H.Y. Shum, "Chaos mosaic: Fast and memory efficient texture synthesis," Tech. Rep. MSR-TR-2000-32, Microsoft, April 2000.

[12] Micheal Ashikhmin, "Synthesizing natural textures," in *Symposium on Interactive 3D Graphics*, 2001.

[13] G. Doretto S. Soatto and Y. N. Wu, "Dynamic textures," in *Proc. of the International Conference on Computer Vision (ICCV)*, 2001.

[14] D. Lischinski Z. Bar-Joseph, R. El-Yaniv and M. Werman, "Texture mixing and texture movie synthesis using statistical learning," Taken from the website http://www.cs.huji.ac.il/ danix/texsyn/.

[15] M. Szummer and R.W. Picard, "Temporal texture modeling," in *Proc. of the International Conference on Image Processing (ICIP)*, Lausanne, Switzerland, 1996.

[16] P. Brodatz, *Textures : a photographic album for artists and designers*, Dover, New York, 1966.

[17] T. Aach, A. Kaup, and R. Mester, "Statistical model-based change detection in moving video," *IEEE Trans. on Signal Processing*, vol. 31, no. 2, pp. 165–180, Mar. 1993.

[18] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps," *J. Fourier Anal. Appl.*, vol. 4, no. 3, pp. 247–269, 1998.

[19] W. Sweldens A. R. Calderbank, I. Daubechies and B. L. Yeo, "Wavelet transforms that map integers to integers," *Appl. Comput. Harmon. Anal.*, vol. 5, no. 3, pp. 332–369, 1998.

[20] J. Reichel, *Complexity-Related Aspects Of Image Compression*, Ph.D. thesis, Swiss Federal Institute of Technology(EPFL), February 2001.

[21] Motion Picture Experts Group, "The MPEG home page," From the website http://mpeg.telecomitalialab.com, 2000.