Texture refinement framework for improved video coding

Fabien Racapé, Marie Babel, Olivier Déforges, Dominique Thoreau, Jérôme Viéron, Edouard Francois

TEXTURE REFINEMENT FRAMEWORK FOR IMPROVED VIDEO CODING

F. Racapé[ab], M. Babel[a], O. Déforges[a], D. Thoreau[b], J. Viéron[b], E. François[b]

[a]Institute of Electronics and telecommunication of Rennes (IETR), Rennes, France
[b]Thomson R&D, Compression lab., Cesson-Sévigné, France

ABSTRACT

H.264/AVC standard offers an efficient way of reducing the noticeable artefacts of former video coding schemes, but it can be perfectible for the coding of detailed texture areas. This paper presents a conceptual coding framework, utilizing visual perception redundancy, which aims at improving both bit-rate and quality on textured areas. The approach is generic and can be integrated into usual coding scheme. The proposed scheme is divided into three steps: a first algorithm analyses texture regions, with an eye to build a dictionary of the most representative texture sub-regions (RTS). The encoder preserves then them at a higher quality than the rest of the picture, in order to enable a refinement algorithm to finally spread the preserved information over textured areas. In this paper, we present a first solution to validate the framework, detailing then the encoder side in order to define a simple method for dictionary building. The proposed H.264/AVC compliant scheme creates a dictionary of macroblocks

Keywords: Video coding, texture synthesis

1. INTRODUCTION

Textured areas are ubiquitous in video pictures. The most common are irregular textures: the patterns are changed with fixed or dynamic statistical properties, so they are not easily, spatially or temporally, predictable. Because of their low predictability, these highly detailed regions require a significant coding cost in usual coding schemes such as H.264/AVC[1] , although they are generally not located in regions of interest. Incidentally, the limitation of their cost by classical encoding algorithms leads to a severe quantization of high frequencies, and creates obvious flattening artefacts.

This paper presents a generic coding framework in which some of the most representative patches of textured regions are coded at a higher quality than the rest of the frame, and then spread their information at decoder side. This quality distribution, taking in account perceptual redundancy, aims at both improving texture quality and saving bit-rate.

An interesting framework[2] proposes to skip some macroblocks at the encoder side, since they are expected to be re-synthesised using their neighbours texture at the decoder side. The choice of macroblocks to be skipped is based on a low-gradient criteria and is independent from the "re-synthesis" algorithm which is graph-cut based.[3] Even though our approach also tends to retrieve texture at decoder side, it differs since in the sense that, instead of skipping, we preserve a low quality version of textured areas that are furtherly refined. This last idea has been considered in,[4] in which the texture synthesis approach[5] is implemented, using a selected texture exemplars and taking the low quality data as initialisation. However, the latter approach has not been linked with a solution at encoder side to select the texture exemplars to be used at decoder side.

In the current framework, we propose a generic scheme in which the selection algorithm is tightly linked to the refinement algorithm in order to improve the overall performance. The current scheme has been studied for a given refinement algorithm operating in the DCT domain but is not only limited to. It is also directly compliant with texture synthesis algorithms.[6,7]

[a]{fabien.racape, marie.babel, olivier.deforges}@insa-rennes.fr
[b]{dominique.thoreau, jerome.vieron, edouard.francois}@thomson.net

Our three steps algorithm analyzes first the input reference video in order to detect the most relevant sub-regions, in terms of texture. These Representative Textured Sub-regions (RTS) are then sent to the encoder, which encodes them at a higher quality than the rest of textured areas. Their information is finally widespread to the rest of lower quality regions, using the refinement algorithm.

The remainder of this paper is organized as follows: in section 2 is introduced the global framework. The proposed scheme is then detailed in section 3. Section 4 presents experimental results obtained in an H.264 based codec.

2. FRAMEWORK

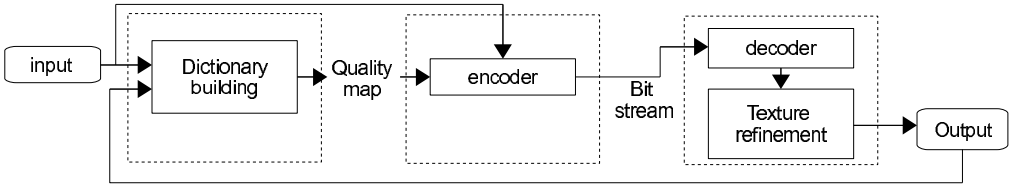


Figure 1. Global synopsis of the algorithm.

The proposed approach aims at encoding textured regions in a new way, in order to save bit-rate without damaging texture quality. Figure 1 presents the conceptual framework, whose three steps are described below.

2.1 Dictionary building

The evaluation of the relevancy of a textured region consists in measuring the level of information it will be able to spread when encoded at a higher quality. Formally, let Π denote the set of the set of the regions in the current frame. Creating a dictionary of RTS is done by searching the subset $\Omega \subset \Pi$, which maximizes

$$\max_{\Pi} \left(\sum_{i \in \Omega} \Delta Q(r(i)) + \sum_{j \notin \Omega} \Delta Q(r(j)/\Omega) \right) \quad (1)$$

Where Q is a quality criteria and $r(i)$ the texture region i.

It results in a quality map, presented in figure 1, made up of RTS, which will be set as input for the encoding step.

2.2 Encoding

The encoding is then applied, driven by the previously defined quality map. Thus, RTS are encoded at a higher quality than the rest of textured part of frames, so that to provide a better distribution of the texture information in the encoded stream. A side information, giving RTS locations for texture refinement step, is also encoded.

2.3 Refining texture

After decoding the previous bit-stream, a texture refining step is applied. It aims at wide spreading RTS quality signal to the rest of textured parts of the frame.

Figure 1 also shows a closed-loop system in order to iteratively improve the selection of RTS. Moreover, such a loop is necessary to avoid a "drift" effect.

3. PROPOSED SCHEME

The section describes a possible instance of the proposed framework.

3.1 Solving the functional

Given a metric of distortion $D[r_1, r_2]$ between two regions r_1 and r_2, and $m(.)$ the refinement operator, equation 1 becomes

$$\max_{\Pi} \big(f(i) + g(j)\big) \tag{2}$$

with

$$\begin{cases} f(i) = \sum\limits_{i \in \Omega} \Big(D\big[r_{Q_0}(i), r_{ref}(i)\big] - D\big[r_{Q_1}(i), r_{ref}(i)\big] \Big) \\ g(j) = \sum\limits_{j \notin \Omega} \Big(D\big[m(r_{Q_1}(j), r_{Q_0}(i_{best}(j))), r_{ref}(j)\big] - D\big[r_{Q_1}(j), r_{ref}(j)\big] \Big) \end{cases} \tag{3}$$

where $f(i)$ represents the quality improvement at RTS locations and $g(j)$ the resulting gain of the refined texture areas. $i_{best}(j)$ denotes the position of the best match for j in Ω, Q_1 and Q_0 the quality criteria of RTS and other regions respectively.

Next section describes the texture refinement algorithm, required for RTS searching.

3.2 Texture refinement

Solving equation 3 requires to know how to use the information of each hypothetical RTS in order to improve the others. Many techniques are directly able to address this problem. One studied technique consists in refining high frequencies, removed by severe quantization, in DCT domain.

The refinement step does not directly use the whole RTS but work with sub-blocks. For each incoming block to be refined, the idea is to consider every sub-block positions in RTS, in order to localize the one, which leads to the best refinement. Hence, for each incoming axa blocks, with $a < r_{size}$, an analysis step computes DCT coefficients for each block and applies

$$\begin{cases} \forall n \in [0, \beta[, & \Lambda_{merged}(n) = \Lambda_{QP_0}(n) \\ \forall n \in [\beta, a^2 - 1], & \Lambda_{merged}(n) = \Lambda_{QP_0}(n) + \alpha\big(\Lambda_{QP_1}(n) - \Lambda_{QP_0}(n)\big) \end{cases} \tag{4}$$

where $\Lambda(n)$ denotes the DCT coefficient at the position n in zigzag order. Two degrees of freedom appear: $\alpha \in]0; 1]$ is the weight of the added information, and $\beta \in [0; a^2 - 1]$ denotes the first DCT coefficient to merge in zigzag order.

Other techniques can also be considered as refinement step: guided texture synthesis[7] , for instance, using dictionary patches as exemplars and low quality textured areas as guide, or pixel-based approaches[5,6] using low quality as iteration 0.

The next section describes the RTS searching step.

3.3 RTS Research

Texture refinement requires some phasing to select the best candidate to proceed each incoming block to refine. As described above, we choose to work with blocks smaller than RTS. Figure 2 shows an example of 3 possible blocks inside a square RTS.

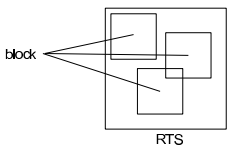


Figure 2. Example of RTS sub-blocks.

We present here two methods for a block-based research: an exhaustive and a cross-correlation based solutions.

3.3.1 Exhaustive research

Figure 3 shows the proposed process, which gets three blocks as input: the block b_{RTS} inside the current potential RTS, with quality Q_1, the current block b_{cur} outside the RTS in the Q_0 frame and its colocalized b_{ori} in the original frame for comparison. The refinement operation is applied between the current block and blocks inside the hypothetical RTS. This exhaustive scheme uses Sum of Square Differences (SSD) as distortion metric D. Two SSD are computed: D_1 between b_{cur} and b_{ori}, and D_2 between the resulting refined block b_{ref} and b_{ori}. $E = D_2 - D_1$ enables then to evaluate the energy supplied by the current block of the hypothetical RTS.

When every b_{RTS} has been processed, we keep

$$b_{best}/D(b_{best}) = \max_{b_{RTS} \in RTS} \big(D(b_{RTS})\big). \tag{5}$$

$$D_{RTS} = \sum_{i \in frame \setminus RTS} D_{best}(i) \tag{6}$$

finally gives the energy in 2 provided by the potential RTS to compare with the others.

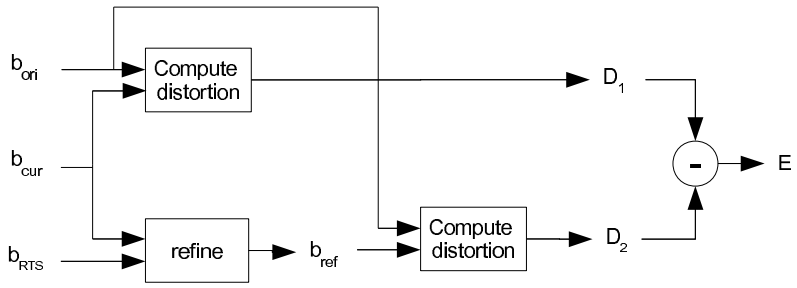


Figure 3. Energy computation for best match searching.

3.3.2 Phase correlation

This section aims at proposing an alternate and less complex solution to the exhaustive research described above. Based on cross-correlation phasing, this new solution can be also used at both encoder and decoder side, since original versions of blocks are not required. We compute cross-correlation as follows:

$$C(r,c) = \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} I(k,l) D(r+k, c+l) \tag{7}$$

where C is a two-dimensions array, I and D are blocks belonging to current image and dictionary respectively. The coordinates r and c where C is maximum correspond to peaks of correlation. We experimentally compute the cross-correlation in the Fourier transform domain. First, the algorithm applies a Blackman windowing,

$$R(r,c) = Bl(r,c).I(r,c) \tag{8}$$

where

$$\begin{cases} Bl(r,c) = b(r).b(c) \\ b(n) = 0.42 - 0.5cos(2\pi n) + 0.08cos(4\pi n)), \end{cases} \tag{9}$$

to both I and D, in order to avoid aliasing. R is the resulting windowed image of I. Cross-correlation C is then defined as

$$C = TF^{-1}(TF(I).TF(D)^*) \tag{10}$$

where $TF(D)$ represents the discrete Fourier Transform of D and $TF(D)^* = TF(D)$ since D is a real signal. The main peak of correlation and its relative vector are then used for the refinement operation.

At decoder side, peaks of correlation P can be compared to the autocorrelation maximum $A_b(0,0)$ of current block b. Hence, a threshold λ on peak values,

$$\lambda = \alpha A_b(0,0) \tag{11}$$

can be used to evaluate whether the incoming block has the same type of texture. So the refinement step can be applied. α represents a kind of confidence threshold.

3.4 Sorting atoms of the dictionary

After having computed the energy provided by each hypothetical RTS, the algorithm has to sort them in order to build a dictionary with the most representative sub-regions.

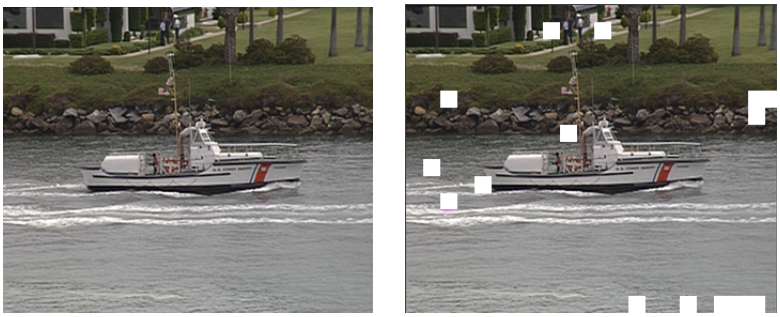
The idea is to build iteratively the dictionary, RTS by RTS. After adding a new RTS, all the related parts of the picture (whose best match belongs to it) will not be considered during next steps. Then, the first "atom" (RTS) is directly computed from 12 with, for now, $Card(\Omega_1) = 1$. And for all the parts of the picture related to Ω_1, the following applies

$$\forall j \notin \Omega_1 / i_{best}(j) \in \Omega_1, D\big[m(r_{Q_1}(j), r_{Q_0}(i_{best}(j))), r_{ref}(j)\big] = 0. \tag{12}$$

The resulting table is used for the next iteration, searching for the second "atom", which also consists in maximizing 1, with $Card(\Omega_2) = 2$. The number of RTS in the dictionary can be either fixed or computed from a distortion threshold.

4. EXPERIMENTAL ANALYSIS

4.1 Experimental context



Original video | Preserved blocks

AVC decoded video | Processed video

Figure 4. Images cooresponding to different steps of the framework.

The described framework has been implemented in an H.264/AVC compliant scheme. Previous sub-regions (RTS) correspond now to 16x16 macroblocks (MB). A MB-based mapping option, enabling to encode MBs at different quantization parameters (QP), has been integrated in the JM encoder[8] .

Although the framework is suitable for inter-frame coding, experiments have been so far carried out frame by frame. So, we only use I frames when encoding sequences with H.264/AVC. Dictionary building is achieved by generating two versions of the video. One high quality Q_1 and one low quality version Q_0, using respectively QP_1 and QP_0. Then, the RTS are extracted, following section 3.

In the following, the two sequences *Coastguard* CIF and *Macleans* CIF are used for tests.

Figure 4 illustrates the different steps of the framework with the selected MBs in white, leading to the decoded frame presented on the third image. The refinement step, according to previous study, enhances it, giving the potential result, presented on the fourth picture.

4.2 Preliminary remark

H.264/AVC standard offers to set a varying parameter QP for some MBs in a same frame. However, encoding a frame at QP_0 with some MBs at QP_1 affects other MBs, i.e. those localized in the causal future, which can be predicted from QP_1 MBs, as shown on figure 5. Note that the difference image has been weighted to lay emphasis on the differences between left and middle frames.

Frame encoded with QP=35 | Some MBs at QP=25 | Differences

Figure 5. Differences between a frame encoded with QP=35 and the same in which some MBs are encoded at QP=25.

Such an encoding scheme, without texture refinement step at decoder side, gives poorer results than H.264 scheme with an only QP. Indeed, only future MBs can be predicted from dictionary MBs, it is intuitively less efficient.

For example, considering the *Coastguard* sequence for quantification parameters: $QP_0 \in [20, 45]$ one can observe an average[9] increase of 6.54% of bit-rate. Thus, we choose to reduce coding efficiency on some MBs in order to both save bit-rate and improve the overall texture quality.

4.3 Potential of the proposed scheme

Gains	4x4 blocks	8x8 blocks
ΔPSNR	2.46dB	0.85dB
Δrate	-17.62%	-6.62%

Table 1. Resulting potential gains using 4x4 and 8x8 blocks.

This section aims at showing the maximum obtainable gains of the algorithm, studying the encoder side, i.e. with decisions based on a distortion with the original frame. Note that it represents the maximum gains for the proposed refinement operator described in section 3. Indeed, the global framework is still under study: in particular, we attempt to implement refinement process based on texture synthesis.

Following figures and tables present the results, with $Card(\Omega) = 15$. Block sizes of 4x4 and 8x8 have been tested because of their practical integration with H.264. Table 1 shows that best results are provided by a refinement operation applied to 4x4 blocks, enabling a better location of the best match by "phasing" inside the dictionary MBs (RTS). Indeed, 4x4 blocks enable to save 17.62% bit-rate whereas 8x8 blocks save 6.62%.

Data	Gains	4 lowest rates	4 highest rates	mean
Coastguard	ΔPSNR	2.71dB	2.83dB	2.77dB
	Δbit-rate	-42.17%	-30.45%	-36.31%
Macleans	ΔPSNR	2.25dB	1.80dB	2.02dB
	Δbit-rate	-34.45%	-20.92%	-27.68%

Table 2. Resulting potential gains depending on ΔQP.

Table 2 presents the potential gains of the proposed scheme with $\Delta QP = QP_0 - QP_1 = 10$. Column 3 presents the average gain on the 4 lowest QP_0, which is 42.17% for bit-rate saving between processed *Coastguard* sequence and an H264/AVC reference. One can see the potential of the proposed scheme. In the following, we

Gains	$\Delta QP = 5$	$\Delta QP = 10$	$\Delta QP = 15$
ΔPSNR	1.95dB	1.59dB	0.97dB
Δrate	-27.43%	-23.18%	14.34%

Table 3. Resulting potential gains depending on ΔQP .

study the impact of varying the parameter ΔQP on the results. Table 3 presents the potential gains, depending on ΔQP, on *Coastguard* sequence. The best results are obtained considering $\Delta QP = 5$ with, for instance 27.43% bit-rate saving. Indeed, a high ΔQP involves more degraded texture to refine.

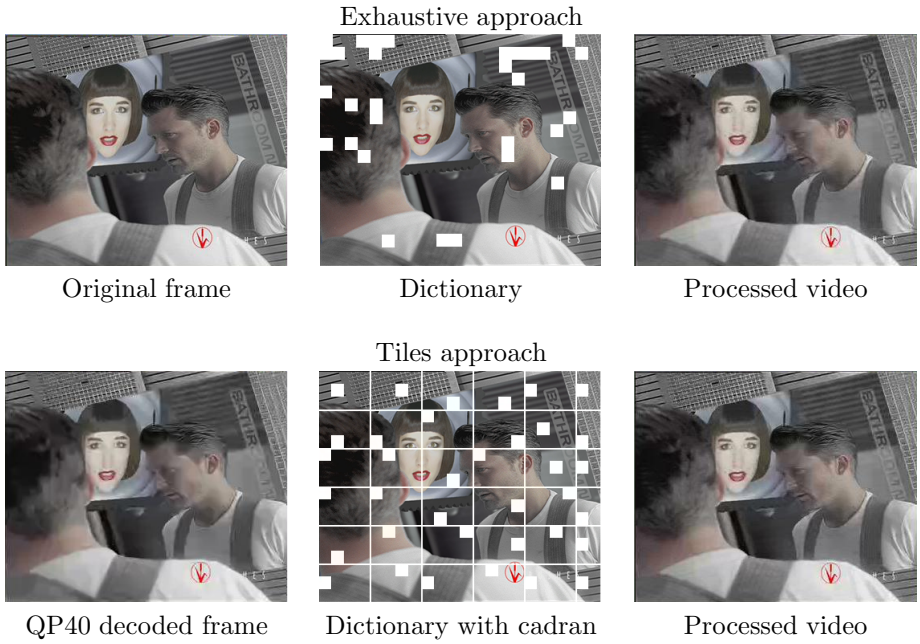


Figure 6. Dictionary and resulting frames. Line 1 presents the Exhaustive scheme and line 2 presents the tile-based scheme (tiles separated by white lines).

4.4 Tiles

We decide here to apply a coarse segmentation step by cutting the frames into tiles, aiming at both doing a first spatial selection of RTS and reducing the computation cost. Figure 6 shows an example of different selections of dictionary MBs and respective resulting frames obtained on the *Macleans* sequence with $QP_0 = 40$ and $\Delta QP = 10$. One can see that the dictionary contains a lot of spatially close RTS in case of exhaustive research, whereas they are widespread with tiles solution, enabling more robust choices at decoder side. Figure 8 shows the results obtained, using 3 $\Delta QP = \{5, 10, 15\}$. Figure 7 compares the results for the Exhaustive and Tiles

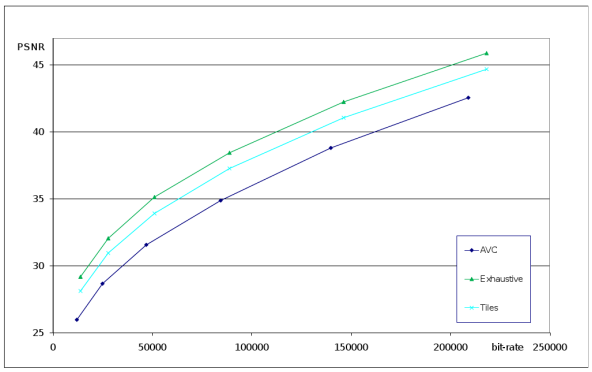


Figure 7. Curves bit-rate/distortion on *Coasguard* for the Exhaustive, Tiles frameworks and AVC.

framework. As expected, Tiles approach gives lower results than the exhaustive one. However, this solution is an interesting compromise to symplify the scheme, since it reduces computation cost by an average factor 35.8.

4.5 Use of cross-correlation

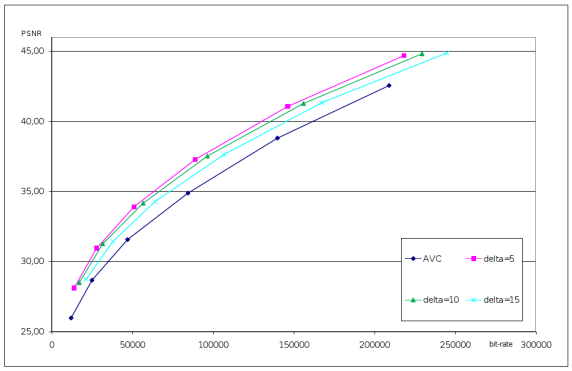


Figure 8. Curves bit-rate/distortion on *Coasguard* for the Tiles frameworks and AVC depending on ΔQP.

This section describes the use of cross-correlation tool, described in section 3, which aims at making the framework symmetric between encoder and decoder.

Table 4 gives the distortions for the algorithm with phase correlation. As expected, gains are lower than exhaustive research. However, this algorithm uses only encoded signal for comparison, whereas previous solutions requires the original versions of current blocks.

Gains	4 lowest rates	4 highest rates	mean
ΔPSNR	0.37dB	0.31dB	0.34dB
Δbit-rate	-4.61%	-4.13%	-4.37%

Table 4. Resulting gains of cross-correlation based algorithm on *Coasguard* sequence with $QP_0 = 40$ and $\Delta QP = 10$.

5. CONCLUSION AND FUTURE WORK

We described here a new video coding framework, aiming at exploiting texture perceptual redundancy, by favouring representative regions, in order to spread their energy to the rest of textured areas. A first approach has been validated, using a H.264/AVC codec. We have presented the potential of the scheme, in terms of bit-rate saving and texture quality improving. Different tools, such as Tiles and cross-correlations, have been then considered in the dictionary building process, aiming at reducing complexity and providing more reliable solutions for both sides of the framework. Further research work is currently under study, in order to improve the proposed scheme. In particular, an ongoing research direction is to experiment with pixel-based texture synthesis methods, guided by the low quality transmitted layer.

REFERENCES

[1] "Draft-T Recommandation and final Draft International-Standard of joint Video Specification (ITU-T Rec. H.264 — ISO/IEC 14496-10 AVC)," (2003).

[2] Zhu, C., Sun, X., Wu, F., and Li, H., "Video coding with spatio-temporal texture synthesis," in [*Proceedings of IEEE International Conference on Multimedia and Expo*], 112–115 (2007).

[3] Kwatra, V., Schdl, A., Essa, I., Turk, G., and Bobick, A., "Graphcut textures: image and video texture synthesis using graph cuts," in [*Proceedings of ACM SIGGRAPH*], 277–286 (2003).

[4] Oh, B. T., Su, Y., Segall, A., and J, K. C.-C., "Synthesis-based texture coding for video compression with side information," in [*Proceedings of IEEE International Conference on Image Processing*], 1628–1632 (2008).

[5] Kwatra, V., Aaron, I. E., and Kwatra, B. N., "Texture Optimization for Example-based Synthesis," in [*Proceedings of ACM SIGGRAPH*], 795 – 802 (2005).

[6] Wei, L.-Y. and Levoy, M., "Fast texture synthesis using tree-structured Vector Quantization," in [*Proceedings of ACM SIGGRAPH*], 479–488 (2000).

[7] Ashikhmin, M., "Synthesizing natural textures," in [*Proceedings of ACM Symposium on Interactive 3D Graphics*], 217–226 (2001).

[8] "JM reference software version 14.0 (current version: 15.1)," (2007).

[9] Bjontegaard, G., "Calculation of average PSNR differences between RD-curves," in [*ITU - Telecommunications Standardization Sector : VCEG-M33*], (2001).