Local texture synthesis: A static texture coding algorithm fully compatible with HEVC

Karam Naser, Vincent Ricordel, Patrick Le Callet

Local Texture Synthesis: A Static Texture Coding Algorithm Fully Compatible With HEVC

Karam Naser, Vincent Ricordel, Patrick Le Callet
LUNAM University, University of Nantes, IRCCyN UMR CNRS 6597
Polytech Nantes, Rue Christian Pauc BP 50609 44306 Nantes Cedex 3, France
karam.naser; vincent.ricordel; patrick.le-callet
@univ-nantes.fr

***Abstract*—Textures are one of the main characteristics of the visual scene. Perceptually, their details are less important than their semantic meaning. This property has been exploited in many texture coding (content based video coding) approaches by removing parts of the textures in the encoder and synthesizing them at the decoder side. Such an approach would necessarily need modification of the coding process and violating the standard.**

This paper introduces a novel algorithm for texture coding called Local Texture Synthesis (LTS), in which texture synthesis is employed in a full compatibility with HEVC standard. This implies that a basic HEVC decoder can be used to reconstruct the signal. LTS defines the necessary conditions to synthesize a patch and produces different synthesis of it. It tries then coding each of them, and finally chooses the one that minimizes the coding cost.

A prototype of this algorithm, based on Markov Random Fields, is given in this paper. This prototype provides up to 10% bitrate saving (using the same quantization parameter) while maintaining an equivalent visual quality.

Keywords*—*Texture Coding, Perceptual Optimization, Content-based Video Coding, Beyond HEVC

I. Introduction

Textures are well known component in image/video analysis as they usually cover wide areas in the visual signal. Nevertheless, it is often difficult to precisely define them. In this paper, we consider the textures as contiguous areas with homogeneous properties (color, orientation, motion, etc). They can range from simple textures (ex. DC patch) to more complicated ones such as grass, water, etc.

The human visual system is usually less attentive to textures than to the shapes or objects which they belong to. In terms of similarity, two textures can look visually indistinguishable even if there is a high point by point difference. For these reasons, texture synthesis algorithms can elegantly generate textures from original ones without a noticeable visual distortion.

Texture synthesis has evolved during the last two decades. There are several examples of a successful approaches such as [1] and [2]. These approaches can synthesize larger texture from a given example. Meanwhile, video coding technology is highly optimized in terms of rate and pixel-wise distortion. The latest MPEG encoder, known as HEVC [3], provides a significant bitrate reduction as compared to the previous MPEG encoders.

The demand on the video streaming and storage is still increasing. New immersive contents, such as Ultra-HD and High Dynamic Range (HDR) videos, open the door to new video coding schemes that can provide higher compression while maintaining an equivalent visual quality. One of the ways to reduce the bitrate is to exploit the knowledge about the human visual perception. There has been a big effort in the context of perceptual image/video coding [4]. The general idea of perceptual coding is to achieve a better coding efficiency in terms of rate-perceptual quality rather than the classical quality computed by PSNR.

The perceptual optimization of video coding can be implemented at various levels of the encoding/decoding process using different perceptual tools. Textures are interesting candidates for perceptual optimization as they can meet both perceptual and coding requirements. There are several approaches (ex. [5][6]) which can efficiently compress some textures by utilizing texture synthesis. Other approaches, such as [7][8][9], aim at enhancing the perceptual quality by embedding perceptual similarity metrics in the encoder cost function.

This paper presents a new algorithm, called Local Texture Synthesis (LTS), for static textures coding. This algorithm bridges the gap between texture synthesis and video coding in the sense that the synthesis is done during the encoding process such that the decoder can independently decode the bitstream. The objective of LTS is to achieve higher bitrate saving using texture synthesis without a need to modify the existing coding standard (HEVC decoder). The algorithm is generic regarding the encoder type and the texture synthesizer. For the purpose of experimental comparison, we present one instance of this algorithm using Markov Random Fields based texture synthesizer.

The rest of the paper is organized as follows. Sec. II provides the general description of LTS. In Sec. III, the details of using LTS in HEVC are described. The simulation and results are given in Sec. IV and the discussion and conclusion are given in Sec. V.

II. Local Texture Synthesis (LTS)

The basic idea behind LTS is: given a texture T, LTS tries to find the best synthesis of it, such that it minimizes its instantaneous coding cost. It can be understood as a re-assembling of the texture elements (known as texels) in a way that they become more encoder friendly, while satisfying some necessary constraints regarding the re-assembling process.

Let's take the example of a general block based encoder that processes blocks in a raster scan order. Considering a given frame (as the one shown in Fig. 1), for a given block (B), we define:

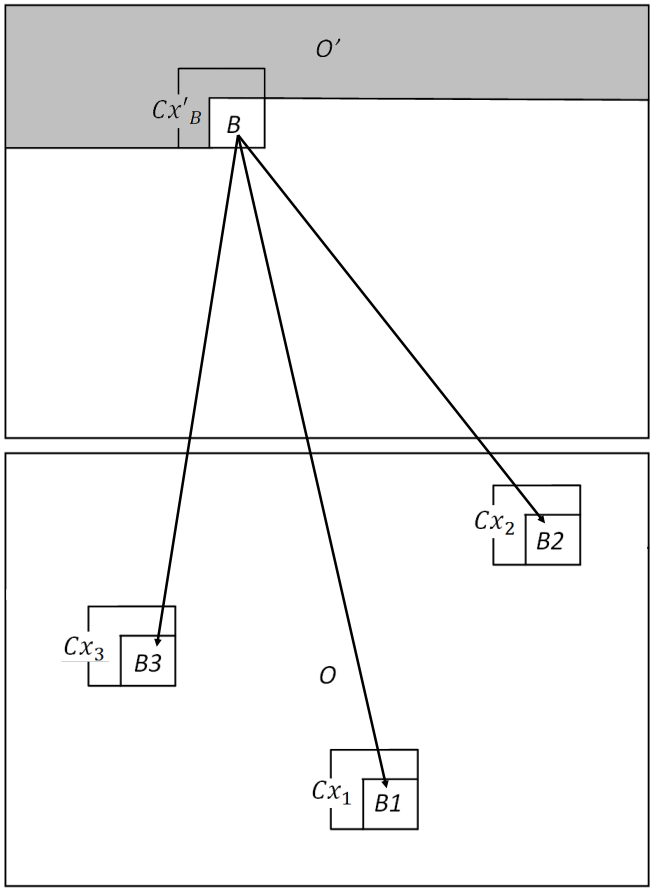

Fig. 1. LTS for general block based encoder.

1) **Original Signal** (O): It is the current frame to be encoded (uncompressed).
2) **Current Synthetic Signal** (O'): It is the synthesized part of the frame from O, whose blocks raster indexes are lower than the current block B index (also uncompressed).
3) **Block Context**: It is the set of pixels above and left from the block being processed. Thus, we can define two possible contexts, Cx and Cx' as its original context in O, or its synthetic context in O' (Fig. 1).

The algorithm is described in Fig 2: given the current frame to be encoded O, it first copies the contents of O into O'. Then, it works iteratively as follows:

For each block to be encoded (B) with its context (Cx'_B) available in O', the algorithm tries synthesizing all possible blocks to replace (B) and thus produces a set $\underline{B}$ of n synthesized blocks. A matching distance is used then to retain a subset $\underline{B}'$ from $\underline{B}$ (of size $m \leqslant n$) that well matches with the given context Cx'_B. Finally, the algorithm encodes each block B_i and selects the one that minimizes the coding cost.

For further understanding of LTS, the details are given in **Algorithm** 1.

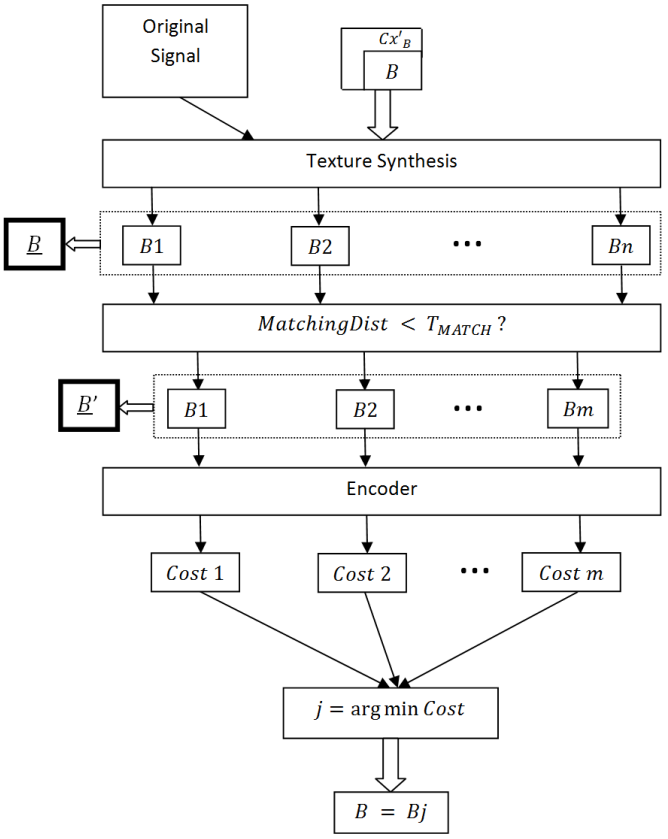

Fig. 2. Illustration of LTS.

Data: Current Frame O
Result: Compressed Frame O^*
Initialize O' with O
for *Every block* B *in* O **do**
  if *exist*(Cx'_B) **then**
    $\underline{B}$= *SynthesizeTexturePatches* (Cx'_B, O)
    for *All* B_j *in* $\underline{B}$ **do**
      if *MatchingDist* $(B_j, Cx'_B) < T_{MATCH}$ **then**
        $B = B_j$ in O'
        Encode(B)
        $Costs[j] = CodingCost(B)$
      end
    end
    $B = B_k$ in $O' \mid k = \underset{j}{\text{argmin}}\ Costs(j)$
  end
  B^* = Encode(B) in O^*
end

Algorithm 1: Local Texture Synthesis

III. Local Texture Synthesis in HEVC

LTS can be employed at different levels in HEVC. For instance, it can be used at the coding tree block (CTB), coding Block (CB) or prediction unit (PB) (cf. [3] for details about CTB, CB and PB). In this paper, LTS was employed at the CB

level. This implies that, for a given CB (with lumma block size ranges from 64x64 to 8x8), the texture synthesizer generates as many blocks as possible that match the context of the current block. The current CB is then replaced by each of these blocks and their coding costs are computed. The algorithm finally replaces this block by the one that minimizes the coding cost. In the following subsections, the details of each process are given.

A. Texture Synthesizer: SynthesizeTexturePatches (Cx'_B, O)

The Texture synthesizer in this work is based on Markov Random Fields (MRF). It generates many blocks as candidates to fill the current block. Referring back to Fig. 1, let the current block B in O, with context of Cx'_B. If there exists a set of blocks ($\underline{B}$) such that their contexts are the **same** as Cx'_B, then, according to MRF theory, for each $B_i \in \underline{B}$:

$$B_i = \underset{x}{\operatorname{argmax}}\ Pr(B = B_x \mid Cx'_B) : B_x \in O \qquad (1)$$

This implies that LTS fills B with one of the most probable blocks obtained from O.

In practice, it is very rare to have exactly the same contexts. For this reason, we define ($\underline{Cx}$) as a set of contexts which are the same as Cx'_B, if all of these contexts have the minimum Euclidean distance to Cx'_B.

B. Matching Distance: MatchingDist (B_j,Cx'_B)

For placing a certain block B_j in a given context Cx', the matching distance is defined as the Euclidean distance between the elements (up to a certain dimension) around the matching line in both horizontal and vertical dimensions.

C. Matching Threshold: T_{MATCH}

In order to judge whether matching distance is acceptable, we need to define a proper threshold. In this work, we provide a way to set this threshold depending on input signal in a dynamic manner. Given the set of same contexts $\underline{Cx}$, we find the minimum matching distance (MC_{min}). The threshold is then defined as:

$$T_{Match} = 1.1 \times MC_{Min} \qquad (2)$$

That is, when the matching distance is within 10% of the best matching distance, the algorithm tests the block for deciding the best coding cost.

D. Coding Cost: $CodingCost(B)$

The coding cost used here is the same one that is defined in the reference encoder, which is:

$$CodingCost(B) = D(B) + \lambda R(B) \qquad (3)$$

where $D(B)$ and $R(B)$ are the distortion and rate when encoding B, and λ is the Lagrangian multiplier adopted in HEVC reference software. As the original signal is not used by the encoder, the distortion $D(B)$ is measured with respect to the synthetic signal (O').

IV. Simulation and Results

In this paper, we used HM 16.2 [10] as a host encoder. The textures used in the simulation are obtained from Brodatz album [11]. In the following, we will discuss the results after encoding each of them.

A. Coding of the First Texture (T_1)

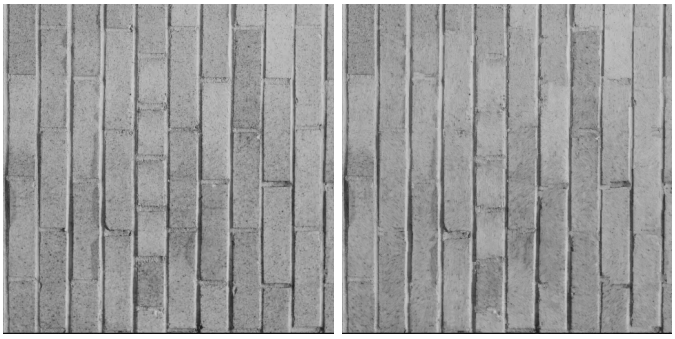

Fig. 3. Compressed T_1 with QP=27. Left: default encoder, right: LTS. Bitrate saving=9.756%.

This texture can be considered as a regular texture. LTS saves about 10% bitrate for the same quantization parameter (QP=27) as shown in Fig. 3. Visually, the quality of the synthesized texture is fairly well compared to the default HEVC encoder. One can notice that when using LTS, some bricks boundaries are eliminated. This is because coding a smooth area is generally less expensive than an edge. The encoder thus chooses to replace the block with an edge by a smoother one that satisfies the constraints given in Sec. III.

B. Coding of the Second Texture (T_2)

The second texture can be classified as a directional texture. Encoding it with LTS provides about 7% bitrate saving (Fig. 4). This texture is more difficult to encode than $T1$. Although that LTS re-arranges it, the texture still contains many discontinuities which consume high bitrate. The quality of the decoded texture is equivalent to the default one, but one can notice that many lines look comparatively disconnected.

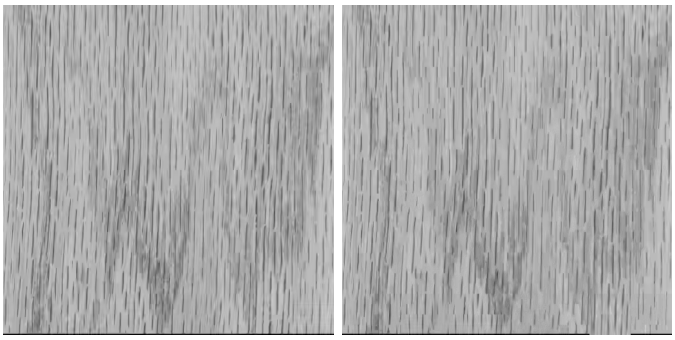

Fig. 4. Compressed T_2 with QP=37. Left: default encoder, right: LTS. Bitrate saving=6.996%.

C. Coding of the Third Texture (T_3)

This type of texture is an irregular one. LTS provides here the least bitrate saving (3%). Comparing the two compressed

textures in Fig. 5, we can notice many point by point difference, but overall, the two textures look very similar.

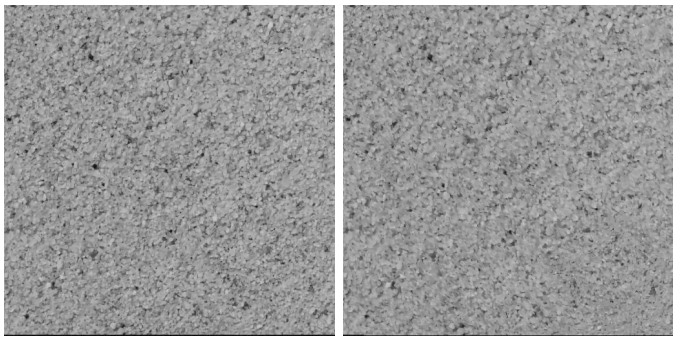

Fig. 5. Compressed T_3 with QP=32. Left: default encoder, right: LTS. Bitrate saving=3.106%.

D. Coding of the Fourth Texture (T_4)

This texture differs from the other three in the sense that its coarseness is degrading from left to right. This is an example of a non perfectly homogeneous texture. When such a texture is encoded with LTS, an oversimplification can occur (as shown in Fig. 6).

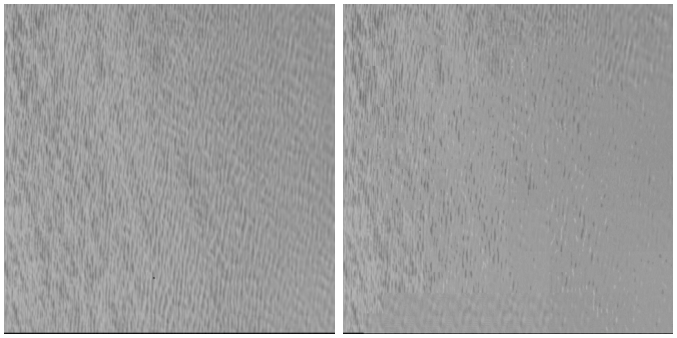

Fig. 6. Compressed T_4 with QP=27. Left: default encoder, right: LTS. Bitrate saving=4.493%.

V. Discussion and Conclusion

We have presented in this paper a new algorithm that allows *online* texture synthesis inside HEVC. This algorithm can be understood as a simplification of textures such that they are compressed more efficiently while keeping nearly a similar overall visual quality. A prototype of this algorithm, based on MRF was adopted, which provided a bitrate saving up to 10%.

The major advantage of LTS is that it is fully compatible with HEVC. This facilitate deploying the algorithm directly to the coding process without a need for modifying the HEVC standard.

LTS can be directly extended to dynamic textures coding by expanding the search range of contexts to the previous and next frames. This will provide higher freedom for the texture synthesizer and potentially increase the coding efficiency.

A potential improvement of LTS could be achieved by including a perceptual distance in order to assess the similarity between two contexts and/or to compute the matching distance. This would offer more reliable decisions and possibly increase the choice flexibility by selecting perceptually similar contexts rather than the ones that minimize the Euclidean distance.

VI. Acknowledgment

This work is supported by the PROVISION (PeRceptually Optimised VIdeo CompresSION) project, which is funded by the European Unions Seventh Framework Programme under grant agreement no 608231.

References

[1] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick, "Graphcut textures: image and video synthesis using graph cuts," in *ACM Transactions on Graphics (ToG)*, vol. 22, no. 3. ACM, 2003, pp. 277–286.

[2] J. Portilla and E. P. Simoncelli, "A parametric texture model based on joint statistics of complex wavelet coefficients," *International Journal of Computer Vision*, vol. 40, no. 1, pp. 49–70, 2000.

[3] G. J. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, no. 12, pp. 1649–1668, 2012.

[4] H. R. Wu, A. Reibman, W. Lin, F. Pereira, and S. Hemami, "Perceptual visual signal compression and transmission," *Proceedings of the IEEE*, vol. 101, no. 9, pp. 2025–2043, Sept 2013.

[5] J. Balle, A. Stojanovic, and J.-R. Ohm, "Models for static and dynamic texture synthesis in image and video compression," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 5, no. 7, pp. 1353–1365, 2011.

[6] F. Zhang and D. R. Bull, "A parametric framework for video compression using region-based texture models," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 5, no. 7, pp. 1378–1392, 2011.

[7] K. Naser, V. Ricordel, and P. Le Callet, "Experimenting texture similarity metric STSIM for intra prediction mode selection and block partitioning in HEVC," in *Digital Signal Processing (DSP), 2014 19th International Conference on*. IEEE, 2014, pp. 882–887.

[8] ——, "Performance analysis of texture similarity metrics in hevc intra prediction," in *Video Processing and Quality Metrics for Consumer Electronics (VPQM)*, 2015.

[9] ——, "Texture similarity metrics applied to hevc intra prediction," in *The third Sino-French Workshop on Information and Communication Technologies, SIFWICT 2015*, 2015.

[10] Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG, " High Efficiency Video Coding (HEVC) Test Model 16 (HM 16) Encoder Description, year = 2014," Tech. Rep.

[11] "USC-SIPI Dataset." [Online]. Available: http://sipi.usc.edu/database/