IMPROVED H.264/AVC CODING USING TEXTURE ANALYSIS AND SYNTHESIS

Patrick Ndjiki-Nya, Bela Makai, Gabi Blättermann, Aljoscha Smolic, Heiko Schwarz, and Thomas Wiegand

Fraunhofer Institute for Communications Engineering – Heinrich Hertz Institute (HHI)
Image Processing Department
Einsteinufer 37, 10587 Berlin, Germany
{ndjiki/blaetter/smolic/hschwarz/wiegand}@hhi.de

ABSTRACT

We assume that the textures in a video scene can be classified into two categories: textures with unimportant subjective details and the remainder. We utilize this assumption for improved video coding using a texture analyzer and a texture synthesizer. The texture analyzer identifies the texture regions with unimportant subjective details and generates coarse masks as well as side information for the texture synthesizer at the decoder side. The texture synthesizer replaces the identified textures by inserting synthetic textures for the identified regions. The texture analyzer is based on MPEG-7 descriptors. Our approach has been integrated into an H.264/AVC codec. Bit-rate savings up to 19.4% are shown for a semi-automatic texture analyzer given similar subjective quality as the H.264/AVC codec without the presented approach.

1. INTRODUCTION

Many video scenes contain textures like water, grass, trees, clouds, sand, etc. These textures are difficult to code because of the large amount of shown detail. But, the exact reproduction of these textures can be considered as not important if they are shown with limited spatial accuracy and the original video is not known to the viewer . The viewer should be able to recognize the textures, which is often not the case when a pre-filter is utilized or these are blurred due to strong quantization. We exploit this idea for video coding using a texture analyzer at the encoder side and a texture synthesizer at the decoder side as shown in Figure 1.

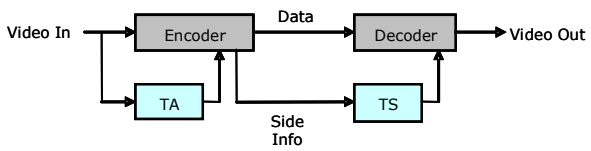

Figure 1 – Video coding using a texture analyzer (TA) and a texture synthesizer (TS).

The texture analyzer identifies detail-irrelevant texture regions, creates coarse masks corresponding to these regions and signals these masks as side information to the decoder in order to run the texture synthesizer. The texture synthesizer replaces the marked textures via inserting synthetic textures. The most important underlying assumption of the presented approach is that for the identified detail-irrelevant textures, known distortion criteria like mean squared error (MSE) are not suitable for efficient coding, since irrelevant detail may be reproduced.

In this paper, we show that it is often sufficient to represent detail-irrelevant textures using a similarity criterion such as an MPEG-7 texture descriptor [1],[2] as the coding distortion. The MPEG-7 similarity criteria lead to reproduced textures that show different details as the original textures. These detail differences are not visible to the viewer as long as the displayed spatial accuracy of the textures remains unchanged and are also much less disturbing as if they were coded at a bit-rate which is equivalent to the bit-rate of the side information of the texture synthesizer.

Analysis-synthesis-based codecs have already been introduced for object-based video coding applications, e.g. see [3]. The purpose of the analyzer and synthesizer modules in this case is usually the identification and appropriate synthesis of moving objects [3]. Such approaches can be seen as complementary to the one presented in this paper as the texture analyzer shown in Figure 1 tends to identify background textures.

A similar wavelet-based analysis-synthesis still image and video coding approach was introduced by Yoon and Adelson [4]. The algorithm presented is optimized for still images. Solutions regarding temporal consistency of synthesized texture regions are not explicitly presented.

The combination of multiple reference frames and affine motion-compensated prediction was introduced by Steinbach et al. [5]. In [5], a segmentation-free solution with more than two reference frames is presented. A suitable reference frame for motion compensation is selected using an MSE-based cost function as the

distortion criterion whereas in this work MPEG-7 similarity measures are employed.

Smolic et al. introduced an online sprite coding scheme [6] that is better suited for real-time applications than MPEG-4's static sprite coding [7]. A major drawback of the approach in [6] is the requirement for a very precise background segmentation.

In this paper, we utilize some of the ideas of [5] and [6] within our framework for video coding using analysis and synthesis.

The remainder of the paper is organized as follows. In Section 2 we introduce the texture analyzer, while in Section 3 the texture synthesizer is described. In Section 4 the system integration is presented. Finally, in Section 5 the experimental results are shown.

2. TEXTURE ANALYZER

The texture analyzer performs a split and merge segmentation of each frame of a given video sequence. The splitting step consists in analyzing a frame using a multi-resolution quadtree [8]. The latter encompasses several levels with the first level (level 0) being the original frame itself. In level 1, the original frame is split into 4 non-overlapping blocks, while it is split into 16 non-overlapping blocks in level 2, etc. The amount of blocks in level L is given by 2^{2L}.

2.1 Homogeneity Criteria

A block in level L is considered to have homogeneous content if its four sub-blocks in level L+1 have "similar" statistical properties. Inhomogeneous blocks are split further, while homogeneous blocks remain unchanged. The splitting stops, when the smallest allowed block size is reached, and the non-homogeneous areas of the considered frame are marked as not classified. The smallest allowed block size can be set according to a priori knowledge concerning the size of the structures in the given video sequence.

The segmentation mask obtained after the splitting step typically shows a clearly over-segmented frame. Thus post-processing of the former is required, which leads to the second step implemented by the texture analyzer - the merging step. For that, homogeneous blocks identified in the splitting step are compared pairwise and similar blocks are merged into a single cluster forming a homogeneous block itself. The merging stops if the obtained clusters are stable, i.e. if they are pairwise dissimilar. The final number of clusters is often considerably reduced by the merging step.

2.2 Similarity Estimation

The similarity assessment between two blocks is done based on MPEG-7 descriptors [1],[2]. We used the "Edge Histogram" (EH) texture descriptor and the SCalable Color (SCC) descriptor.

The EH descriptor represents the spatial distribution of four directional edges (one horizontal, one vertical, and two diagonal edges) and one non-directional edge for 16 local non-overlapping regions of a given image. The frequency of occurrence of each edge class is determined for each local region. This leads to an 80 (16x5) dimensional feature vector.

The SCC descriptor is basically a color histogram in the HSV color space. Note that the SCC descriptor can in principle be used in combination with other color spaces.

Two blocks are considered to be similar if the distance between the corresponding feature vectors lies below a given threshold:

$$\begin{aligned} d_{\ell_1}(\mathbf{SCC}_1, \mathbf{SCC}_2) &\leq T_{SCC} \\ d_{\ell_1}(\mathbf{EH}_1, \mathbf{EH}_2) &\leq T_{EH} \end{aligned} \qquad (1)$$

where d_{ℓ_1} represents the ℓ_1 metric, $\mathbf{SCC}_i$ / $\mathbf{EH}_i$ (i=1,2) are the feature vectors of the considered blocks, while T_{SCC} and T_{EH} are the similarity thresholds. The latter are thereby manually optimized for the key frames of the given sequence. The optimal threshold is then used for all frames of the video. The texture analyzer presented here can therefore be seen as a semi-automatic segmentation algorithm. A fully automatic analysis approach is currently under investigation.

2.3 Temporal Consistency

The splitting and merging steps segment each frame of a given sequence independently of the other frames of the same sequence. This yields inconsistent temporal texture identification. Thus a mapping of textures identified in a frame to textures identified in previous frames of the same sequence is required. However, in our approach it is important that the temporal consistency of identified textures is provided for a group-of-frames (GoF). A GoF encompasses two key frames (first and last frame of the GoF) and several partially synthesized frames between the key frames. Key frames are either I or P frames and coded using MSE as distortion criterion. This GoF-based video processing approach allows solving problems related to detail-irrelevant textures that become detail-relevant, e.g. when a zoom occurs and a detail is shown with much higher spatial accuracy than before. The system can then seamlessly switch between a synthesized and an MSE-accurate texture representation.

Temporal consistency of detected synthesizable textures is ensured by setting up a "texture catalogue". Each identified texture is mapped to one of the indexed textures if similar or added to the texture catalogue otherwise.

2.4 Warping of Segmented Areas

The reliability of the color- or texture-based identification of synthesizable parts of a GoF is increased by matching

the texture regions in the partially synthesized frames with the corresponding texture regions in the key frames. This mapping is achieved by warping the identified texture regions in the current frame towards the corresponding textures in the first or the last frame of the GoF. Warping is done using the planar perspective model as defined by the Parametric Motion Descriptor in MPEG-7 [1],[2]:

$$x' = [(a_1 + a_3 x + a_4 y) / (1 + a_7 x + a_8 y)] + x$$
$$y' = [(a_2 + a_5 x + a_6 y) / (1 + a_7 x + a_8 y)] + y \quad (2)$$

where (x', y') represent the warped coordinates of the original sample (x, y). $a_1,\ldots,a_8$ are the eight model parameters. The perspective motion model is suitable to describe arbitrary rigid object motion, if the camera operation is restricted to pure rotation and zoom. It is also suitable for rigid motion of planar objects with arbitrary camera operation. In practice these assumptions often hold approximately over the short period of a GoF as considered here. The parametric motion (parameters a_i) of each identified texture region in relation to the first and last frame of the GoF is estimated as described in [9].

Within the current system, texture synthesis can only be performed if corresponding regions can be found in the first or last frame of the GoF. Therefore a given identified texture region in the actual frame is warped towards the first frame of the GoF. The samples of the warped texture region that lie within the corresponding texture region of the first frame of the GoF are kept, while the others are labelled as not classified in the current frame. This leads to a reduced texture region in the current frame. The procedure is repeated using the last frame of the GoF as a reference.

3. TEXTURE SYNTHESIZER

Two texture synthesizers are presented in the following. The difference between the two synthesis approaches resides in the underlying texture-related assumptions.

The first texture synthesizer (texture synthesizer I) warps the texture from the first or the last frame of the considered GoF towards each synthesizable texture region identified by the texture analyzer as illustrated in Figure 2. A motion parameter set and a control parameter are required by texture synthesizer I for each synthesizable texture region identified by the texture analyzer. The control parameter determines whether the current texture region is to be synthesized using the first or the last frame of the considered GoF. The reference frame that leads to the best synthesis result is thereby used. That is, the motion parameters $a_1,\ldots,a_8$ that lead to the smallest difference signal between synthesized and original texture region are kept. This synthesizer works remarkably well for rigid objects with the assumptions implied in the motion model.

The second texture synthesizer (texture synthesizer II) enables the representation of local motion activity in a given texture region, which makes it suitable for the synthesis of non-rigid textures. Texture synthesizer II models textures based on Markov Random Field methods [10] regarding a given texture as a realization of a local random process.

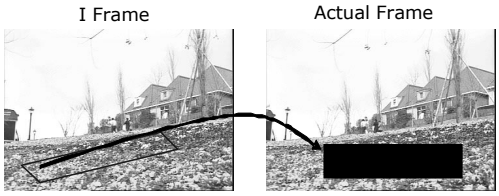

Figure 2 – Texture synthesizer I, warping of texture from reference frame towards region to be filled

That is, each texture sample is predictable from a small set of spatially neighboring samples and is independent of the rest of the texture. Figure 3 depicts the principle of texture synthesizer II. A given texture region is warped from the first and / or the last frame of the considered GoF towards the current frame given a control parameter word (2 bits, i.e. only left reference or only right reference or both references available) and two / one motion parameter set(s) sent by the encoder (cp. Section 2.4). Note that the samples of the warped texture region are not directly inserted into the corresponding missing texture region in the current frame.

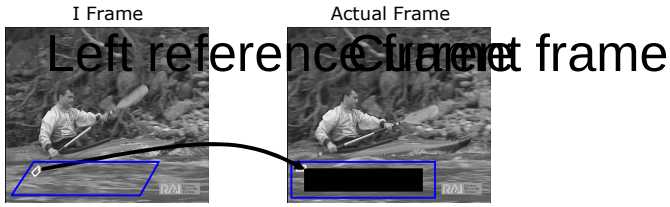

Figure 3 – Texture synthesizer II, warping of texture from reference frame towards region to be filled

The causal neighborhood of each sample of the missing texture is compared to the neighborhoods of the warped samples within a restricted area (typically 3x3 samples) in order to capture local motion characteristics of the given texture. The warped sample with the most similar neighborhood (MSE) is finally copied to the location of the corresponding sample of the missing texture.

4. SYSTEM INTEGRATION

We have incorporated the texture analyzer and synthesizer into an H.264/AVC codec. In our implementation I and P frames are key frames and coded using MSE. B frames are candidates for a possible partial texture synthesis. When a B frame contains identified synthesizable texture regions, the corresponding segmentation mask, the corresponding motion parameters as well as the corresponding control flags are transmitted as side information.

While decoding, all macroblocks belonging to a synthesizable texture region are handled as skipped macroblocks [11]. After all macroblocks of a frame are completely decoded, the texture synthesizer is called, and all reconstructed YUV samples of macroblocks belonging to a synthesizable texture region are replaced.

5. EXPERIMENTAL RESULTS

In our experiment we have integrated the texture analyzer and texture synthesizer into an H.264/AVC video codec. We then analyzed and synthesized three well known test sequences, Flowergarden, Concrete and Canoe. The sequences Flowergarden and Concrete contain rigid textures useful to demonstrate that an approximate representation of some textures can be done without subjectively noticeable loss of quality. The Canoe sequence contains water, which is a good representative of the non-rigid texture class.

The following set-up was used for the H.264/AVC codec (Joint Model 2.1): Three B frames, one reference frame for each P frame, CABAC (entropy coding method), rate distortion optimization, 30Hz progressive video at CIF resolution. The quantization parameter QP was set to 16, 20, 24, 28 and 32.

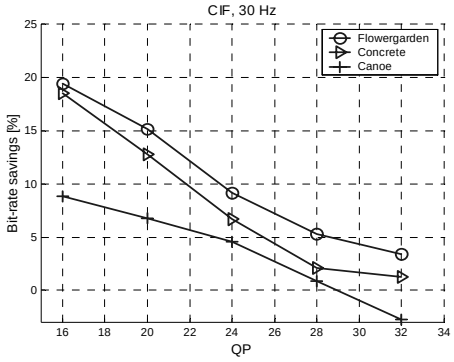

Figure 4 – Bit-rate savings w.r.t. quantization accuracy

Figure 4 depicts the bit-rate savings obtained for each of the test sequences. It can be seen that the highest savings were measured for the highest quantization accuracy considered (QP=16). Substantial bit-rate savings of 19.4% (Flowergarden), 18.5% (Concrete) and 8.8% (Canoe) were measured at this QP value. The bit-rate savings decrease with the quantization accuracy due to the fact that the volume of the side information remains constant over the different QP settings. The visual quality at the selected QP settings was in all cases comparable to the quality of the decoded sequences using the standard codec. Sequences for subjective evaluation can be down-loaded from http://bs.hhi.de/~ndjiki/SE.htm.

6. CONCLUSIONS AND FUTURE WORK

We have presented a video coding approach based on texture analysis and synthesis. We classify a given video scene into detail-relevant and detail-irrelevant texture regions. Detail-irrelevant texture regions are detected by a texture analyzer and regenerated by a texture synthesizer.

We tested our system by integrating our modules into an H.264/AVC codec. Bit-rate savings up to 19.4% are shown for a semi-automatic texture analyzer given similar subjective quality as the standard H264/AVC codec.

The interaction between texture analyzer and synthesizer is subject to future work. Especially a more precise analysis of the synthesizable texture regions is required to avoid synthesizing texture regions with low texturization and thus little bit-rate to code them using MSE.

7. REFERENCES

[1] ISO/IEC JTC1/SC29/WG11/N4358, „Text of ISO/IEC 15938-3/FDIS Information technology – Multimedia content description interface – Part 3 Visual“, Sydney, Australia, July 2001.

[2] ISO/IEC JTC1/SC29/WG11/N4362, „MPEG-7 Visual Part of eXperimentation Model Version 11.0“, Sydney, Australia, July 2001.

[3] M. Wollborn, „Prototype Prediction for Colour Update in Object-Based Analysis-Synthesis Coding”, IEEE Transactions on Circuits and Systems for Video Technology, Special Issue on Very Low Bit Rate Video Coding, Vol. 4, No. 3, pp. 236-245, June 1994.

[4] S.-Y. Yoon and E. H. Adelson, „Subband texturesynthesis for image coding“, Proceedings of SPIE on Human Vision and Electronic Imaging III, Vol. 3299, pp. 489-497, San Jose, CA, USA, January 1998.

[5] E. Steinbach , T. Wiegand, and B. Girod, „Using Multiple Global Motion Models for Improved Block-Based Video Coding“, Proc. ICIP1999, IEEE International Conference on Image Processing, Vol. 2, pp. 56-60, Kobe, Japan, October 1999.

[6] A. Smolic, T. Sikora and J.-R. Ohm, „Long-Term Global Motion Estimation and its Application for Sprite Coding, Content Description and Segmentation“, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 9, No. 8, pp. 1227-1242, December 1999.

[7] ISO/IEC JTC1/SC29/WG11/N3515, „MPEG-4 Video VM Version 17.0“, Beijing, China, July 2000.

[8] J. Malki et al., „Region Queries without Segmentation for Image Retrieval by Content”, VISUAL'99, pp.115-22, 1999.

[9] A. Smolic and J.-R. Ohm, „Robust Global Motion Estimation Using a Simplified M-Estimator Approach“, Proc. ICIP2000, IEEE International Conference on Image Processing, Vancouver, Canada, September 2000.

[10] L.-Y. Wei and M. Levoy, „Fast Texture Synthesis using Tree-structured Vector Quantization“, Proc. of SIGGRAPH 2000, Conference on Computer Graphics and Interactive Techniques, New Orleans, Louisana, USA, July 2000.

[11] T. Wiegand (Ed.) „Editor’s Proposed Draft Text Modifications for Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC), Awaji draft“, Awaji, Japan, January 2003.