VIDEO CODING USING TEXTURE ANALYSIS AND SYNTHESIS

Patrick Ndjiki-Nya, Bela Makai, 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/makai/smolic/hschwarz/wiegand}@hhi.de

Abstract

A new approach to video coding is presented, where video scenes are classified into textures with subjectively relevant and irrelevant details. We apply this idea to improve video coding by using a texture analyzer and a texture synthesizer. The analyzer identifies the texture regions with no important subjective details and generates coarse masks as well as side information for the synthesizer at the decoder side. The synthesizer replaces the detail-irrelevant textures by inserting synthetic textures into the identified regions. Texture analyzer and synthesizer are based on MPEG-7 descriptors. The 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 grass, trees, sand, water, clouds, etc. These textures are difficult to code because of the large amount of visible detail. However, the exact reproduction of these textures can be considered as not important if they are shown with a limited spatial accuracy – the details are often irrelevant. Moreover, since the viewer typically does not know the original video, it is very unlikely that the exact reproduction of details is important. The viewer should just be able to recognize the textures, which is often not the case when for instance 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.

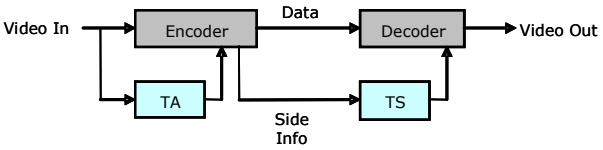

Figure 1 – Video coding using a texture analyzer (TA) and a texture synthesizer (TS)

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 to run the texture synthesizer. The texture synthesizer replaces the textures identified by the masks 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.

One important problem associated with the presented approach occurs, when a detail-irrelevant texture turns into a detail-relevant texture, e.g. when a zoom occurs and a detail is shown with much higher spatial accuracy then before. For that, a technique is needed to seamlessly switch between synthesized and MSE-accurate texture representation. The approach presented in this paper also addresses this issue.

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 consis-

tency 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 the Lagrangian cost function as the distortion criterion. The chosen cost function is MSE-based 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 static sprite coding in MPEG-4 [7]. In MPEG-4 static sprite coding, the background of a video scene is transmitted at the beginning of the sequence. A major drawback of the approach in [6] is the requirement for a very precise background segmentation, i.e. foreground objects have to be separated very accurately.

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

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^L.

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 and the related samples are marked as not classified when the smallest allowed block size is reached. This 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.

Figure 2 shows the segmentation masks of a frame after the splitting (left frame) and after the merging step (right frame). Regions labelled as not classified are marked by a black border, while classified regions are marked by a non-black border. It can be seen that the number of homogeneous clusters is substantially reduced after the merging step.

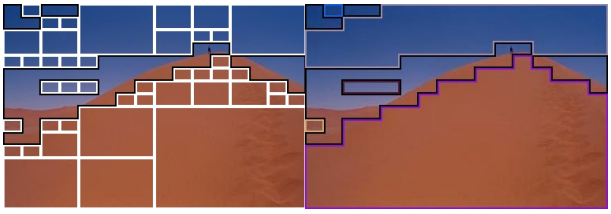

Figure 2 – Segmented image after the splitting step (left) and after the merging step (right)

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 colour histogram in the HSV colour space. The resolution (number of colours or bins) of the SCC descriptor can be varied from 16 to 256 colours. The number of possible colours is thereby doubled for each resolution step. We use the highest resolution step for best possible segmentation results given the SCC descriptor. Note that the SCC descriptor can in principle be used in combination with other colour spaces.

Two blocks are considered to be similar if the distance between the corresponding feature vectors lies below a given threshold. The latter is implemented as a proportion of the maximum possible distance that is dependent on the selected metric (l_1, l_2) and the used descriptor (SCC, EH). A threshold of zero indicates that two feature vectors must show a 100 % match to be seen as similar while a threshold of one means a 0 % match is sufficient to be seen as similar. The similarity threshold is manually selected and fixed for a given sequence.

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 approach enables seamless switching between between synthesized and MSE-accurate texture representation after each key frame.

Temporal consistency of the synthesizable parts of a GoF is ensured by setting up a "texture catalogue", which contains information about the textures present in the given sequence. The texture catalogue is initialized with the feature vectors of the textures identified in the first frame of the sequence. In case no texture is identified in the starting frame, the catalogue is initialized with the textures of the first frame where at least one texture is found. The textures identified in the following frames are first compared to the indexed texture(s) and mapped to one of them if similar. The former are added to the texture catalogue otherwise.

2.4 Warping of Segmented Areas

The reliability of the colour- or texture-based identification of synthesizable parts of a GoF is increased by matching the texture regions into the partially synthesized frames with the corresponding texture regions in the key frames. This mapping is achieved by warping of 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]:

$$\begin{aligned} x' &= [(a_1 + a_3 x + a_4 y) / (1 + a_7 x + a_8 y)] + x \\ y' &= [(a_2 + a_5 x + a_6 y) / (1 + a_7 x + a_8 y)] + y \end{aligned} \quad (1)$$

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 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].

The warping 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. We have found that this approach yields good but sometimes too conservative results in that the resulting regions are often too small.

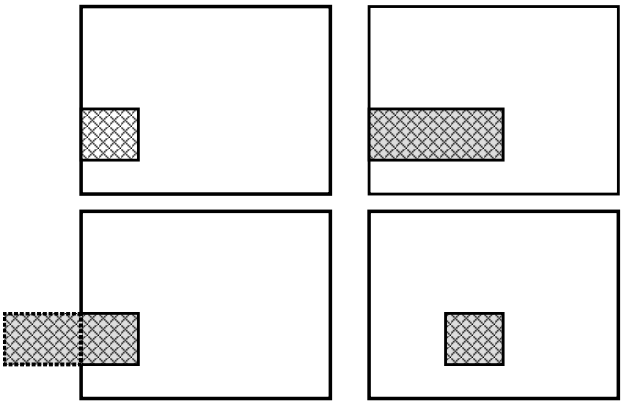

Figure 3 – Example for segmentation mask generation using warping. Top left: segmentation mask of the key frame with one synthesizable (hatched) texture region. Top right: segmentation mask of current frame to synthesize with one synthesizable (hatched) texture region of the same class as hatched region in the key frame. Bottom left: segmentation mask of key frame with warped synthesizable texture region of current frame to synthesize. Bottom right: Resulting segmentation mask of current frame to synthesize after motion compensation.

Figure 3 depicts an example of how motion compensation of segmentation masks is done. The hatched texture regions are of the same class and represent a synthesizable texture area. It is assumed that the synthesizable texture region is subject to translational motion. Only the intersection of the warped texture region in the current frame and the corresponding texture region in the reference frame is used for synthesis. It can be seen that part of the current texture region lies outside the key frame. Figure 3 (bottom right) depicts the segmentation mask resulting from the motion compensation.

Note that parameters like the smallest allowed block size (Section 2.2) and the similarity thresholds (Section 2.3) are optimized manually. The texture analyzer presented above is therefore a semi-automatic segmentation algorithm.

3. Texture Synthesizer

For the texture synthesizer, we assume rigid objects. We further assume that the frame-to-frame displacement of the objects can be described using the perspective motion model. Synthesis for textures like water that are not rigid are currently under investigation.

The texture synthesizer 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 4. A motion parameter set and a control parameter are required by the texture synthesizer for each synthesizable texture region identified by the texture analyzer [9]. The control parameter indicates whether the current texture region is to be synthesized using the first or the last frame of the considered GoF. The key frame that leads to the best synthesis result is used. That is, the motion parameters $a_1,\ldots,a_8$ that lead to the smallest MSE between synthesized and original texture region are kept.

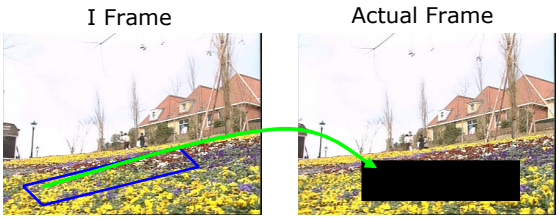

Figure 4 – Texture synthesizer filling texture region identified by texture analyzer using left reference frame

4. System Integration

We have incorporated the texture analyzer and synthesizer into the reference software (JM 2.1) of the H.264/AVC project [10]. In this implementation, I and P frames are always conventionally coded; only B frames are candidates for a possible texture synthesis. In case a B frame contains identified synthesizable texture regions, the corresponding segmentation mask, the corresponding motion parameters as well as the corresponding control flags have to be transmitted (Sections 2.1 and 2.2).

For macroblocks that are marked by the control parameter, the texture synthesizer is used at the decoder and all reconstructed samples of macroblocks belonging to a synthesizable texture region are generated.

5. Experimental Results

We have conducted tests on the two well known test sequences Flowergarden and Concrete. Both of them contain textures useful to demonstrate that an approximate representation of some textures can be done without subjectively noticeable loss of quality.

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

Figure 5 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) and 18.5 % (Concrete) were measured at this QP value using semi-automatically generated masks. 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. It can be seen that for the highest QP value considered (QP=32) the bit-rate savings are still of 3.41% (Flowergarden) and 1.26% (Concrete).

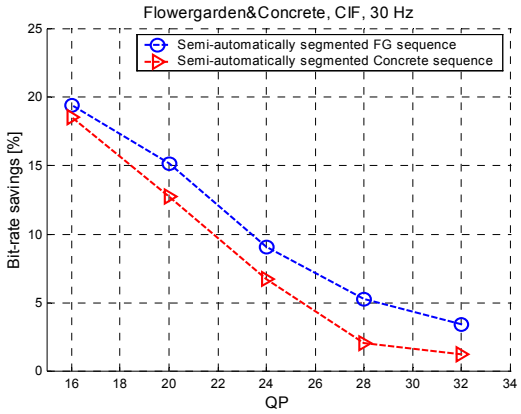

Figure 5 – Bit-rate savings vs. quantization accuracy

The visual quality at the selected QP values 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.

Figure 6 depicts the result obtained for the sixth frame of the "Flowergarden" sequence. It can be seen that the difference signal is nearly zero in the sky texture region, while it is quite significant in the flower texture region. However, there is hardly any difference visible when comparing the synthesized and the original sequences. This shows that PSNR may not be a suitable measure for this part of the video. We are not aware of suitable alternative quality measures matching the human subjective perception of the video. Thus no video quality evaluations based on objective measures are presented in this paper. Note that due to some misclassifications in the sky texture region (some leaves of the tree assigned to the sky texture class), some details are missing in the partially synthesized frame. As can be seen in bottom of Figure 6 this artefact is not noticeable if the original frame is not available at the decoder as is the case in transmission systems.

Figure 6 – Coding result for Flowergarden. Top left : Original frame (#6). Bottom left: Frame with synthesized texture regions. Top right: Difference signal (gain factor 3). Bottom right: Conservative motion compensated segmentation mask (color black corresponds to detail-relevant texture region).

Conclusions and future work

We have presented a new 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 reproduced by a texture synthesizer.

We have tested our idea 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 further 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. For these textures, the gain is rather small. Moreover, incorporating more motion features into the analyzer might help to improve consistency of identified textures for synthesis.

We are also investigating synthesis for textures like water that contain local motion and therefore require different synthesizer approaches.

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 texture-synthesis 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] 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.