VIDEO CODING WITH QUAD-TREES AND ADAPTIVE VECTOR QUANTIZATION

Marcel Wagner[*] *and Dietmar Saupe*[†]
[*]Institute of Computer Science, Freiburg University, Am Flughafen 17, 79110 Freiburg, Germany
e-mail: wagner@informatik.uni-freiburg.de
[†]Institute of Computer Science, Leipzig University, Augustusplatz 10/11, 04109 Leipzig, Germany
e-mail: saupe@informatik.uni-leipzig.de

ABSTRACT

This paper presents recent progress of our ongoing research evaluating the prospects of adaptive vector quantization (AVQ) for very-low-bitrate video coding. In contrast to conventional state-of-the-art video coding based on entropy coding of motion compensated residual frames in the frequency domain, adaptive vector quantization offers the potential to adapt its codebooks to the changing statistics of image sequences. The basic building blocks of our current AVQ video codec are (1) block-based coding in the wavelet domain where wavelet coefficients correspond to (overlapping) spatial regions, (2) hierarchical organization of the wavelet coefficients using quad-tree structures, (3) three way coding mode decision for each block (block replenishment, product code vector quantization, new VQ block with codebook update), and (4) rigorous rate/distortion optimization for all coding choices (image partition and block coding mode). This video codec does not apply motion compensation, however. A comparison with standard transform coding (H.263) shows that inspite of the improvements of our coder over previously published AVQ video coders it still shows a performance gap of about 1 dB for some test sequences. We conclude that motion compensation is essential also for codecs based on AVQ. First preliminary tests show that AVQ coders that incorporate motion compensation can become competitive with standard transform coding.

1 INTRODUCTION

Vector quantization (VQ) has been proven a powerful compression scheme for coding of images and image sequences [1, 2]. However, in most schemes a static codebook is used. The vector quantizer applies only one fixed codebook neglecting that image sequences typically are not stationary and that different sequences have different statistical behavior. Thus, adaptive vector quantization (AVQ) for image sequences was proposed [3, 4]. In order to improve the AVQ-codec, a rate-distortion (RD) optimization was investigated in [5, 6, 7]. It was shown in [8] that vector quantization in the wavelet domain leads to additional coding gains.

In this paper, we present a new video coding approach for very low bitrates. It is based on hierarchical AVQ in the wavelet domain and does not use motion compensation. We show that the AVQ-scheme presented in [8] can be significantly improved by the use of hierarchical quad-tree (QT) coding. Coding results and comparisons with standard transform coding (H.263) are provided. In addition we describe first experiments combining our AVQ scheme with motion compensation.

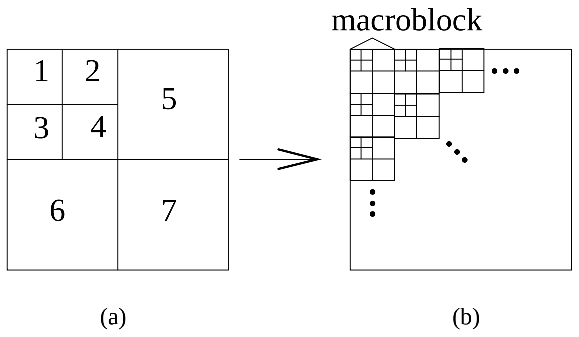


Figure 1: Regrouping of macro blocks in the wavelet domain. (a) Subband grouping. Coefficients are grouped corresponding to their frequency. (b) Macro block grouping. Coefficients are grouped corresponding to their spatial position.

2 OUTLINE OF THE CODEC

In a preprocessing step a two times octave-band wavelet transform (9/7 biorthogonal filter [9]) is applied to each frame. The coefficients of the transformed frames are regrouped into macro blocks each corresponding to a 16×16-pixel area in the spatial domain. The organization of the wavelet coefficients of a macro block is shown in Fig. 1. Each macro block consists of 16 coefficients from subbands 1,2,3,4, respectively, and 64 coefficients from subbands 5,6,7, respectively, a total of 256 coefficients.

For every macro block there are three possible decomposition levels from coarse to fine. Level-0 describes the whole macro block with all 256 wavelet coefficients. Alternatively, the level-0 block can be decomposed into four level-1 blocks each one corresponding to a spatial 8×8-pixel area, containing four coefficients from subbands 1,2,3,4, respectively, and 16 coefficients from subbands 5,6,7, respectively. Thus a level-1 block consists of 64 coefficients. Moreover, one

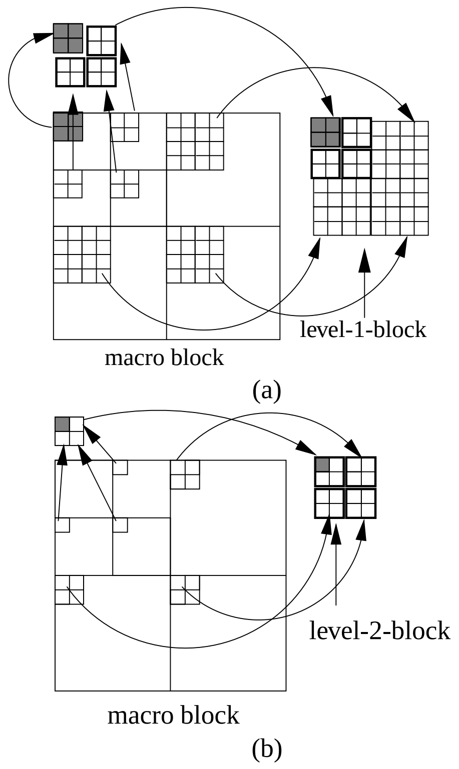


Figure 2: Grouping of wavelet coefficients in (a) level-1-blocks and (b) level-2-blocks

can decompose each level-1 block into four level-2 blocks each one corresponding to 4×4-pixel blocks in the spatial domain containing one coefficient from subbands 1,2,3,4, respectively, and four coefficients from subbands 5,6,7, respectively. The grouping of the wavelet coefficients for level-1 and level-2 blocks within a macro block is depicted in Fig. 2. Obviously, the decomposition of a macro block can be described by a quad-tree.

To create one dimensional vectors, the blocks are scanned line by line, excluding coefficients from several subbands, depending on the block level. There are three kinds of vectors level-0, level-1 and level-2 vectors. For level-0 vectors we take all coefficients of the level-0 blocks, i.e. it is 256 dimensional. For level-1 vectors, only the coefficients from subbands 2,3 and 4 of level-1 blocks are taken. The coefficients of higher subbands are set to zero, whereas the four coefficients of subband 1 are handled separately. Thus the dimension of level-1 vectors is 12. For level-2 vectors all coefficients of level-2 blocks are taken except the coefficient of subband 1. This is, analogous to level-1 vectors, treated separately. Therefore the dimension of level-2 vectors is 15.

For level-0 vectors there is only one encoding mode. Replenishment is applied, i.e. the content of the same position in the previously decoded transformed frame is restored. For level-1 and level-2 vectors the AVQ-approach is applied. There are three encoding modes:

Mode 0. Replenishment mode. This works like the replenishment mode for level-0 blocks, i.e. the content of the same position in the previously decoded frame is restored.

Mode 1. VQ-mode. Low-pass coefficients are scalar quantized and encoded separately. The vector containing the remaining coefficients is quantized and encoded by the VQ.

Mode 2. Update mode. Low-pass coefficients are scalar quantized and encoded. The vector containing the remaining coefficients is scalar quantized and encoded.

For level-1 vectors and level-2 vectors different vector quantizers are used since level-1 vectors are 12-dimensional and level-2 vectors are 15-dimensional. In order to maintain an efficient variable length encoding, the codebooks $\mathcal{C}_0 = \{x_1^0, \ldots, x_{n_0}^0\}$ and $\mathcal{C}_1 = \{x_1^1, \ldots, x_{n_1}^1\}$ of the two vector quantizers are organized as follows. A frequency count is assigned to every vector. At the beginning of the encoding of a frame, the vectors are sorted with respect to the frequency count. The most frequent vector can be found at the beginning of the codebook, the least frequent vector at the end. The vector at the beginning of the codebook can be encoded with the shortest variable length code (VLC), the vector at the end with the longest. This frequency count is incremented by one every time a vector is used by mode-1-encoding. When the encoding of a frame is finished, all vectors are sorted again. After that, the vectors encoded in mode 2 have to be inserted in the codebook by a heuristic rule [6]. The new vectors are assigned a frequency count of $f(\lfloor \frac{n}{2} \rfloor) + 1$ with codebook size n and $f(i)$ describing the frequency count of the ith vector in the codebook; i.e. the frequency count of the vector in the middle of the codebook is taken and incremented by one. After all mode-2-vectors have been inserted, the vectors with the least frequency count are removed in order to maintain a constant codebook size.

For variable length coding, an adaptive arithmetic coder is used. The decision how to decompose the quad-trees and how to choose coding modes is done with an algorithm based on the generalized BFOS algorithm [10]. Thus, the modes and the quad-tree decompositions are selected in a *rate-distortion optimal* fashion. For the encoding of the chrominance components, about $\frac{1}{10}$th of the given bitrate is used.

3 RESULTS OF THE QUAD-TREE CODEC

In this section, we provide some coding results of our new codec and comparisons with the tmn codec [11]. The experiments are made with QCIF frames and a codebook size of $n_0 = 64$ and $n_1 = 512$. The bin size of the scalar quantizer for the low-pass coefficients and for the mode-2-encoding is 16. The codec uses the frame rate of 8.3 frames/s, i.e every third frame is used. Generally, the peak signal to noise ratios (PSNR) are computed using only the luminance component.

Figure 3 and 4 show the average PSNR of the AVQ-codec with and without quad-trees for bitrates between 8000 to 28800 bits/s. The first 20 PSNR values are skipped since the initialization phase should not affect the average PSNR

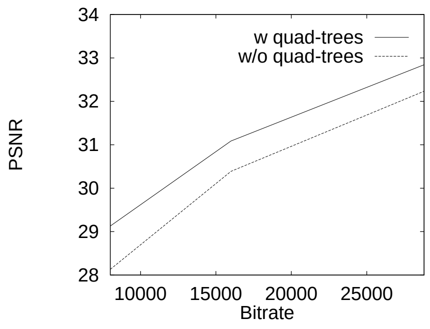


Figure 3: Average PSNR for *mthr & dotr*.

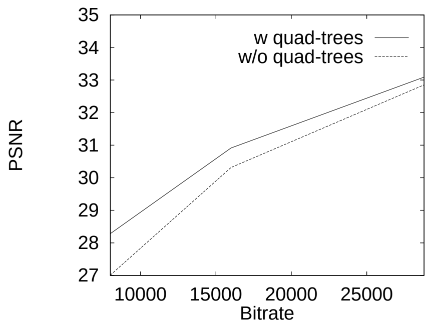


Figure 4: Average PSNR for *salesman*.

performance. For bitrates around 28800 bit/s, the quad-tree coder is about 0.5 dB PSNR better. For lower bitrates around 8000 bit/s, the application of quad-trees improves the encoding by more than 1 dB. Both figures show that the use of quad-trees leads to a large coding gain.

Unfortunately only a small number of comparable results have been reported for VQ coding of video sequences without motion compensation. Thus, we cannot provide fair comparisons with other VQ-approaches. To give the reader something at hand to asses the performance of our codec, we provide a comparison with the tmn codec. The tmn coder (version 3.0) is used with syntax based arithmetic coding at a bitrate of 8000 bit/s and a frame rate of 8.3 frames/s. The target bitrate is achieved by the off-line rate control method of the tmn codec, i.e. the codec tries to adjust an average bitrate of $\frac{8000}{8.3}$ bits/frame over the whole sequence. The result is depicted in Figure 5. One can see that at the beginning the performance gap between these two encoding schemes is very large. The gap is decreasing during the encoding of the first 100 frames. After that an average difference of about 1.2 dB PSNR between the tmn and the quad-tree codec is maintained over the rest of the sequence. After subsequent 100 frames the average performance gap is about 0.9 db PSNR. This shows the adaptability of our approach that need more than 100 frames to evolve. Note that we are not using motion compensation.

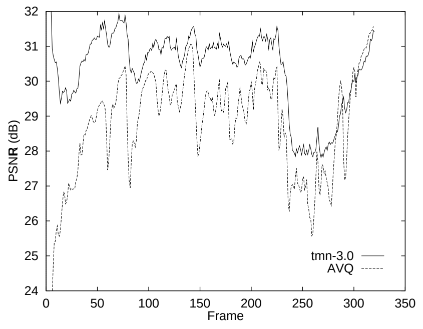


Figure 5: PSNR course for *mthr & dotr* at 8000 bit/s for the tmn and the AVQ-codec.

4 EXPERIMENTS WITH MOTION COMPENSATION

In this section we describe an experiment combining the AVQ concept with motion compensation. For this, we substitute the inter frame discrete cosine transform (DCT) coding of the tmn-codec [11] by AVQ coding. Unlike Sect. 2 we now have to organize the inter-frame macro blocks of the tmn-codec in the *spatial domain*. However, this still can be made by quad-tree structures used in Sect. 2.

Again there are three block levels: Level-0 blocks describing the whole macro block, level-1 blocks describing 8×8-pixel blocks and level-2 blocks describing 4×4-pixel blocks. Level-0, level-1 resp. level-2 vectors can be created by scanning level-0, level-1 and level-2 blocks line by line.

Level-0 vectors can only be encoded with replenishment. Level-1 vectors have the additional option to be represented by a mean. For level-2 vectors the full AVQ scheme is applied:

Mode 0. Replenishment mode.

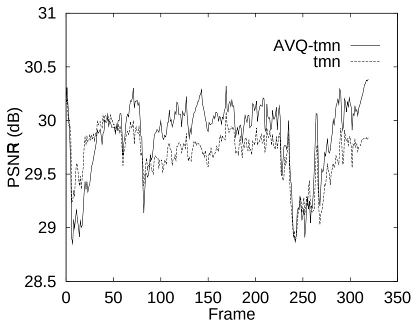
Mode 1. VQ-mode. The full vectors is vector quantized.

Mode 2. Update mode. The vector is scalar quantized and transmitted to the decoder.

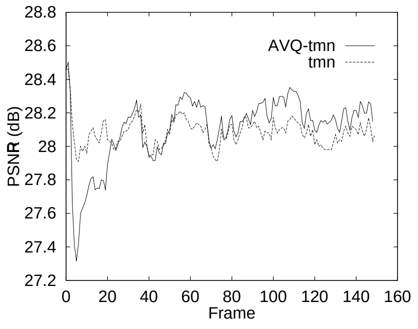
The codebook organization is made as described in Sect. 2. Apparently this time there is only one codebook $\mathcal{C}$. The codebook $\mathcal{C}$ contains only level-2 vectors with a codebook size of 512.

Within this framework we make the following experiment. First we run the pure tmn-codec with a fixed quantization parameter q and the *syntax based arithmetic coder* option. The bits used for every frame are stored. After that we apply the new AVQ-tmn codec on the same frames, using for every frame the same number of bits that the pure tmn-codec consumed. The frame format is QCIF and every third frame is taken.

Figure 6a shows the result for the *Mother & Daughter* sequence and the quantization parameter $q = 20$ resulting in an average of 800 bits per frame. At the beginning, the pure tmn codec outperforms the AVQ-tmn codec by about 0.4 dB PSNR. This is reversed after 50 frames. Then the AVQ-tmn



(a)



(b)

Figure 6: Comparison of the pure tmn codec and the AVQ-tmn codec for the (a) *Mother & Daughter* and (b) *Salesman* sequence. The tmn codec uses the quantization parameter $q = 20$. The resulting average number of bits per frame is around 800.

codec is about 0.4 dB PSNR better than the pure tmn codec. A similar observation can be made for the salesman sequence in Fig. 6b. The coding gain is, however, smaller than 0.2 dB PSNR.

The results of our first experiments show that, after an initialization phase, the AVQ performs better or at least as well as the discrete cosine transform. Note that the AVQ-tmn uses RD-optimization for the mode-decision. In order to provide a fair comparison the pure tmn codec should also apply RD-optimization for mode decision. However, applying RD-optimization only on mode decision, without optimizing motion compensation, leads to an improvement less than 0.2 dB PSNR [12]. But optimizing motion compensation would also improve the results of our AVQ-tmn codec.

5 CONCLUSION

In this paper we have presented a video encoding scheme without motion compensation. This approach is based on AVQ and quad-tree coding. The codec adapts its codebook on a frame by frame basis. It was shown that the new codec improves our previously published AVQ-codec. A comparison with standard transform coding (H.263) shows that inspite of the improvements of our coder over previously published AVQ video coders it still shows a performance gap of about 1 dB for some test sequences. We conclude that motion compensation is essential also for codecs based on AVQ. In fact, first results presented in this paper investigating the combination of our AVQ-approach with motion compensation indicate that the performance may improve to the level of standard techniques based on DCT and beyond. In future work we will consider this issue in more detail.

References

[1] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, 1992.

[2] M. Goldberg, P.R. Boucher, and S. Shlien. Image compression using adaptive vector quantization. *IEEE Trans. Commun.*, COMM-34(2):180–187, Feb. 1986.

[3] X. Wang, S. Shende, and K. Sayood. Online compression of video sequences using adaptive VQ codebooks. In *Proc. DCC'94 Data Compression Conference*, Snowbird, Utah, March 1994.

[4] D. Saupe and B. Butz. Real-time very low bit rate video coding with adaptive mean-removed vector quantization. In *Proc. of the ICIP'97*, Santa Barbara, 1997.

[5] J. E. Fowler and S. C. Ahalt. Adaptive vector quantization using generalized threshold replenishment. In *Proceedings of the 1997 IEEE Data Compression Conference*, 1997.

[6] R. Hamzaoui, D. Saupe, and M. Wagner. Rate-distortion based video coding with adaptive mean-removed vector quantization. In *Proc. of the ICIP'98*, Chicago, USA, October 1998.

[7] M. Lightstone and S. K. Mitra. Image-adaptive vector quantization in an entropy-contrained framework. *IEEE Transaction on Image Processing*, 6(3), March 1997.

[8] M. Wagner, R. Herz, H. Hartenstein, R. Hamzaoui, and D. Saupe. A video codec based on R/D-optimized adaptive vector quantization. In *Proc. DCC*, page 556, 1999. Full paper available as Technical Report 119, Institut für Informatik, Universität Freiburg.

[9] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies. Image coding using wavelet transform. *IEEE Transactions on Image Processing*, 1(2):205–220, April 1992.

[10] P.A. Chou, T. Lookabaugh, and R.M. Gray. Optimal pruning with applications to tree-structured source coding and modeling. *IEEE Trans. on Inform. Theory*, IT-35:299–315, March 1989.

[11] Tmn 3.0 (h.263) codec. Released by the Signal Processing and Multimedia Group, University of British Columbia, `http://spmg.ece.ubc.ca`.

[12] G.J. Sullivan and T. Wiegand. Rate-distortion optimization for video compression. *IEEE Signal Processing Magazine*, pages 74–90, 1998.