RD-Optimization of Hierarchical Structured Adaptive Vector Quantization for Video Coding[1]

Technical Report 139

Marcel Wagner[1] and Dietmar Saupe[2]

[1]Universität Freiburg, Am Flughafen 17, 79110 Freiburg, Germany
wagner@informatik.uni-freiburg.de
[2]Universität Leipzig, Augustusplatz 10/11, 04109 Leipzig, Germany
saupe@informatik.uni-leipzig.de

Abstract

This paper contains two contributions to very-low-bitrate video coding. First, we show that in contrast to common practice incremental techniques for rate-distortion optimization such as the generalized BFOS algorithm may clearly outperform the standard technique based on Lagrangian multipliers. This is relevant in cases where the computation of RD-points has a low complexity. Second, we report on 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

[1]Presented at IEEE Data Compression Conference 2000, Snowbird, Utah.

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. A non-standard RD optimization scheme based on the generalized BFOS [9, 1] is applied and compared with Lagrangian multiplier based standard RD techniques. In addition complexity considerations are presented. Coding results and comparisons with standard transform coding (H.263) are provided.

2 Outline of the codec

In a preprocessing step a two times octave-band wavelet transform (9/7 biorthogonal filter [10]) is applied to each frame. Transformed frames are decomposed into macroblocks each corresponding to a 16×16-pixel area in the spatial domain. The organization of the wavelet coefficients of a macroblock is shown in Figure 1.

For every macroblock there are three possible encoding levels from coarse to fine. Level 0 coding describes the whole macroblock. Alternatively, the level-0 block can be decomposed into four level-1 blocks corresponding to a spatial 8×8-pixel area. Moreover, one can decompose each level-1 block into four level-2 blocks each one corresponding to 4×4-blocks in the spatial domain. The grouping of the wavelet coefficients for level-1 and level-2 blocks within a macroblock is

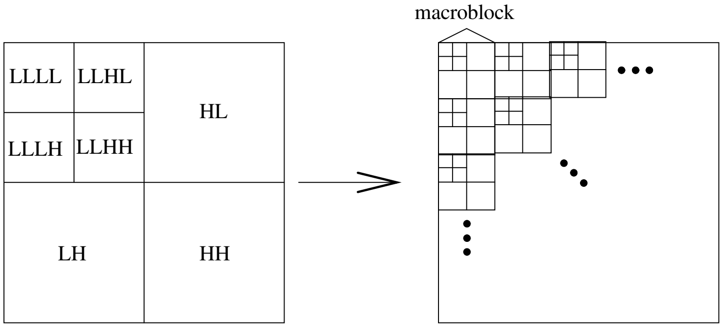


Figure 1: Grouping of macroblocks in the wavelet domain.

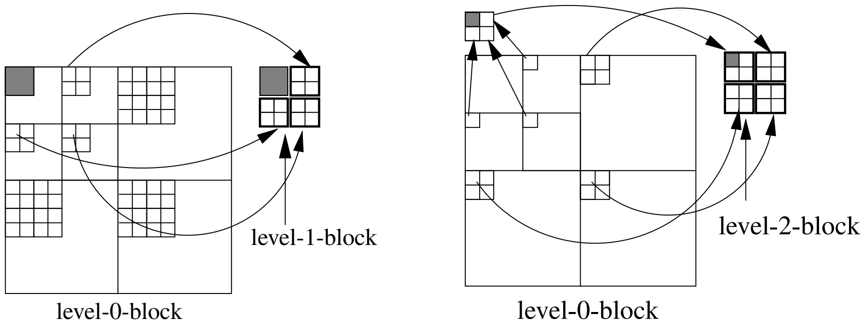


Figure 2: Grouping of the wavelet-transform coefficients.

depicted in Figure 2. Note that for the blocks corresponding to 8×8-pixel areas we have indicated only the 4×4- low frequency coefficients. The other high frequency coefficients are set to zero. Obviously, the decomposition of a macroblock can be described by a quad-tree.

For level-0 blocks 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 blocks the AVQ-approach is used. 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 transformed frame is restored.

Mode 1. VQ-mode. Low-pass coefficients (4 for level-1 and 1 for level-2 blocks) are quantized and encoded separately. The vector containing the remaining coefficients is quantized and encoded by the VQ.

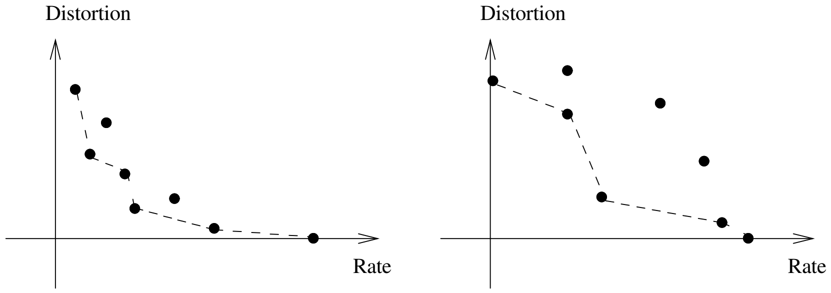
Mode 2. Update mode. Low-pass coefficients are scalar quantized and encoded. The vector containing the remaining coefficients is scalar quantized and encoded.

For level-1 blocks and level-2 blocks different vector quantizers are used since the vectors are 12-dimensional for level-1 blocks and 15-dimensional for level-2 blocks and since blocks from different levels show differing distributions. In order to maintain an efficient variable length encoding, the codebooks $\mathcal{C}_1 = \{x_1^1, \ldots, x_{n_1}^1\}$ and $\mathcal{C}_2 = \{x_1^2, \ldots, x_{n_2}^2\}$ 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 [9]. Thus, the modes and the quad-tree decompositions are selected in a *rate-distortion optimal* fashion. Details and justification for this non-Lagrangian optimization approach will be found in the next section. For the encoding of the chrominance components, about $\frac{1}{10}$th of the given bitrate is used.

3 Rate-distortion optimization

In this section, the two main techniques for RD-optimization are described and compared: Lagrangian optimization and incremental computation of the rate-distortion curve. Usually, computation of RD-points that are needed for the optimization is much more complex than the optimization itself. In this case, the



X_0 X_1

Figure 3: Operational rate-distortion curve.

complexity of the RD-optimization technique can be neglected. However, this assumption is not true for "fine-grained" RD-optimization as in the context of our video codec. A small number of operations is needed to compute the RD-points as we will see in section 3.3. Hence a complexity consideration of RD-optimization algorithms is necessary. We will show that for RD-points created by our AVQ-codec the predominating Lagrangian multiplier approach [11] is outperformed by an algorithm based on the BFOS-approach [1, 9].

In general, encoding of a frame introduces a bit allocation problem. Given M macroblocks $X_0, \ldots, X_{M-1}$ that have to be encoded independently and N encoding options $I = \{i_0, \ldots, i_{N-1}\}$ for each macroblock, every encoding option i of a macroblock X_m describes an RD-point (R_m^i, D_m^i) with encoding rate R_m^i and the corresponding distortion D_m^i. Thus, in order to minimize the overall distortion D subject to a given bitrate constraint R, one has to find $D = \min \sum_{m=0}^{M-1} D_m^{i_m}$, subject to $\sum_{m=0}^{M-1} R_m^{i_m} \leq R$, where the options $(i_0, \ldots, i_{M-1}), i_m \in I$ describe a possible encoding of the frame. In principle, this problem can be solved by dynamic programming [12], however, with great computational expense. Figure 3 shows a set of RD-points for two different macroblocks. The dashed line describes the points that could contribute to a solution of the bit allocation problem, the so called *operational RD-curve*.

In order to reduce the computational complexity one can consider only solutions on the lower part of the convex hull (LCH) of the RD-points. There are two main approaches for such RD-optimization techniques. The first is based on a Lagrangian multiplier (LM) approach [13, 11]. The second is based on an incremental computation of the convex hull [14, 9].

3.1 The Lagrangian multiplier

In classical LM approaches continuous differentiable functions are assumed for minimization, but it was shown in [13] that this technique also works in the discrete case. Thus, the constrained problem

$$\text{minimize} \sum_{m=0}^{M-1} D_m^{i_m}, \text{ subject to} \sum_{m=0}^{M-1} R_m^{i_m} \leq R \tag{1}$$

can be transformed into an unconstrained problem

$$J(\lambda) = \sum_{m=0}^{M-1} J_m(\lambda) = \sum_{m=0}^{M-1} \min_{i_m \in I} \lambda \cdot R_m^{i_m} + D_m^{i_m}. \tag{2}$$

Moreover, it can be shown that if $R^* = \sum_{m=0}^{M-1} R_m^{i_m}$ is the solution of the unconstrained problem (2) then it is also a solution of the constrained problem (1) with $R = R^*$. A geometrical interpretation of the solution is depicted in Figure 4. A line with slope $-\lambda$ is drawn through the origin. Then it is shifted towards the LCH. The first point that is met is the point minimizing $J_m(\lambda)$.

A problem of the LM approach is to determine the value of λ that yields a solution for the given bitrate R. Usually a bisection search is applied [15, 11].

The complexity of this approach can be estimated by the complexity of the computing of the $D_m^i \cdot \lambda + R_m^i$ term in every macroblock X_m and for every encoding option i. If the algorithm uses l iterations to find an optimal λ, a total number of $l \cdot |I| \cdot M \cdot 2$ operations (the cost for the minimization is neglected) are needed [3].

For our AVQ-codec a special LM approach for quad-trees [16] is necessary.

3.2 Incremental computation of the rate-distortion curve

Encoding a frame amounts to choosing an option $i_m \in I$ for every block X_m. Every combination of options $(i_0, \dots, i_{M-1}), i_m \in I$ can be assigned to a global RD-point described by the total rate $R = \sum_{m=0}^{M-1} R_m^{i_m}$ and the total distortion $D = \sum_{m=0}^{M-1} D_m^{i_m}$. Starting from the minimal (maximal) possible total rate, the algorithm by Westerink *et al.* [14] computes the global LCH for increasing (decreasing) total rate until the target rate is reached. This algorithm works as follows: First, the smallest achievable bitrate is searched, i.e. for every macroblock

[3] We assume equal complexity of the addition, multiplication, and division operation. This is, however, platform dependent.

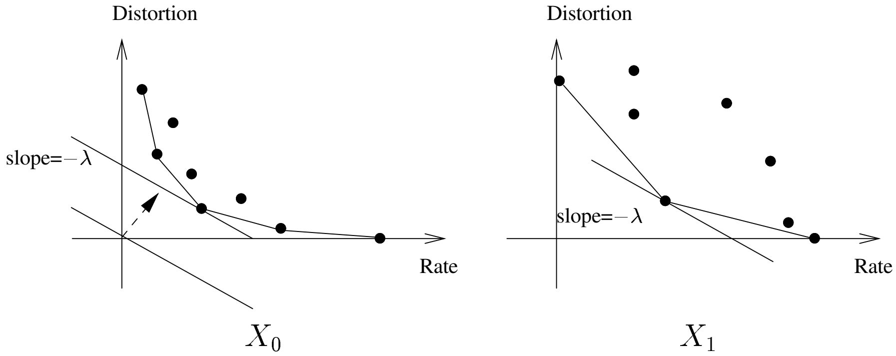


Figure 4: Geometrical interpretation of the Lagrangian multiplier.

X_m the encoding option i_m with the smallest rate is computed. For every macroblock X_m the coding option j_m describing the next point on the local LCH is determined. Let λ_m be the slope between the RD-points specified by i_m and j_m. The macroblock X_n with the "steepest", i.e. minimal, λ_n is selected. The coding option of X_n is changed $i_n := j_n$. Then the next point on the local LCH of X_n is searched and the values of j_n and i_n are updated accordingly. After that, again the "steepest" λ_m is searched and so forth. This algorithm is detailed in Figure 5.

The BFOS algorithm [1, 9] generalizes this approach in order to make it applicable for tree structured coding. However, for this approach a monotonicity condition is assumed, i.e. it must be assured that a larger tree leads to a larger encoding cost and to a smaller distortion. In addition it is assumed that there is only one encoding option for a node. Both assumptions are not met in our hierarchically organized AVQ-codec. Thus, a more general approach (comparable to [17], pp. 33–38) without these restrictions is needed for our codec structure.

Since the core of this new algorithm can still be described by the algorithm in Figure 5, we use this algorithm for the complexity estimation. Therefore, we consider two phases, the *initialization* and the *optimization* phase. The initialization phase is dominated by $\frac{D^i_m - D^{i_m}_m}{R^i_m - R^{i_m}_m}$ calculations. Thus, $|I| \cdot M \cdot 3$ operations are needed. Let k be the number of steps on the LCH that are performed during the optimization phase. If we neglect the search for the macroblock n with the smallest lambda (this search is only logarithmic in M), a step in the optimization phase is dominated by the search for the next lambda. Thus, the number of operations is $|I| \cdot 3$ per step. This yields a total number of $k \cdot |I| \cdot 3$ operations. Note that this computation can be made more efficient since the fractional term must only be evaluated if $R^i_m > R^{i_m}_m$ *and* $D^i_m < D^{i_m}_m$. In summary, the total complexity of the optimization is $3 \cdot |I| \cdot (k + M)$.

3.3 Complexity

Since both algorithms compute the same solution of the RD-problem, an interesting point of the comparisons of these algorithms is the complexity.

For simplicity, we considered in the discussion of the two optimization techniques above only macroblocks. However, as described in section 2 a macroblock may be split into several blocks of several levels. Only for the four level-1 and 16 level-2 blocks vector quantization is applied. The four level-1 blocks are vector quantized using a codebook size $n_1 = 64$ and a vector dimension of 12. The 16 level-2 blocks are vector quantized using $n_2 = 512$ and a vector dimension of 15. Thus, vector quantization for level-2 blocks needs about 32 times more operations than for level-1 blocks. Hence it is sufficient for the estimation of the complexity to take only the level-2 blocks into account. We are using the QCIF format, in which a frame consists of 99 macroblocks each containing 16 level-2 blocks, i.e. $M = 99 \cdot 16 = 1584$. Thus, from here on M is the total number of all level-2 subblocks. The three encoding modes lead to $|I| = 514$ encoding options (one for mode 0, 512 for mode 1, and one for mode 2).

First, we consider the complexity of computing the RD-points. Obviously, the complexity is dominated by the computation of the distortions for each encoding option. In the context of our video codec, the encoding options are the different vectors that could represent a certain part of the macroblock. For level-2 blocks the vectors are 15-dimensional. In order to compute the sum of squared errors, 15 multiplications and 14 additions are necessary. Thus, 29 operations are needed to compute one RD-point for one subblock. The total number of operations to compute the RD-points for level-2 blocks is $29 \cdot |I| \cdot M$.

Now we compare the complexity of the algorithms described in section 3.1 and 3.2. We applied our AVQ-codec to the standard test sequences *Miss America*, *Salesman* and *Mother and Daughter*. In the first run we did the RD-optimization with the LM approach. In order to find the initial interval for the bisection search, we used the lambda computed in the last frame and doubled resp. halved it until a rate-interval was found that includes the target rate. Then the bisection search was applied terminating when the rate interval remained fixed in one iteration. The average number l of iterations per frame can be seen in Table 1a. This experiment was repeated with the BFOS based optimization technique from section 3.2. The average number of k steps of the optimization phase of a frame is depicted in 1b. Note that one step of the BFOS approach has the complexity of $\frac{3}{2M}$th of one LM

bitrate (bit/s)	8000	16000	28800
Miss America	9.1	9.8	10.3
Salesman	8.8	9.8	9.8
Mthr. & Dotr.	9.1	9.8	10.3

(a)

bitrate (bit/s)	8000	16000	28800
Miss America	200	351	572
Salesman	232	344	457
Mthr. & Dotr.	247	386	554

(b)

Table 1: Average number of (a) iterations for the LM approach, (b) steps of the BFOS optimization phase.

	w/o opt.	w. opt.
computing RD-points	3.63s	1.02
LM approach	2.27s	1.73s
BFOS initialization	0.28s	0.1s
BFOS optimization	0.06s	0.04s

Table 2: Runtime per frame of the AVQ-codec.

iteration.

In order to find the RD-optimal solution, around $l = 9$ iterations are needed for the Lagrangian multiplier approach. For the BFOS approach, at most $k = 572$ steps are needed. Including the initialization phase, the complexity to find the optimal solution is in the worst case around 2 iterations of the LM approach, which makes it at least 4.5 times as fast as the LM method.

These observations are endorsed by runtime experiments made on a silicon graphics O2 with an R10000 processor. Table 2 describes the average per frame runtime for computing the RD-points, the LM iterations, and the *initialization* and *optimization* phase of the BFOS algorithm. Furthermore, first the codec was compiled without any optimization by the compiler and then with compiler optimizations. Without optimization, the BFOS approach including *initialization* and *optimization* is around 7 times faster than the LM algorithm. The time needed for the LM-approach is about the same order of magnitude as the computation of the RD-points. Actually the LM technique performs as expected since $\frac{3.63}{2.27} \approx \frac{29 \cdot |I| \cdot M}{9 \cdot 2 \cdot |I| \cdot M}$. In Table 2 one can see that our AVQ-codec based on the BFOS technique currently needs less than 1.2 seconds per frame for VQ and RD-optimization. Together with the wavelet transform (not optimized) one frame is encoded in less than 1.5 seconds. We conclude that the Lagrangian multiplier method as a choice for the RD-optimization can be significantly inferior to incremental methods like the generalized BFOS algorithm.

INITIALIZATION
for all macroblocks $X_m, 0 \leq m < M$ **do**
$i_m := \arg\min\{R_m^i | i \in I\}$
$j_m := \arg\min_{i \in I}\{\frac{D_m^i - D_m^{i_m}}{R_m^i - R_m^{i_m}} | R_m^i > R_m^{i_m} \wedge D_m^i < D_m^{i_m}\}$
$\lambda_m := \frac{D_m^{j_m} - D_m^{i_m}}{R_m^{j_m} - R_m^{i_m}}$
end for
$rate := \sum_{m=0}^{M-1} R_m^{i_m}$
OPTIMIZATION
while $rate < target\_rate$ **do**
$n := \arg\min\{\lambda_m | 0 \leq m < M\}$
$rate+ = R_n^{j_n} - R_n^{i_n}$
$i_n := j_n$
$j_n := \arg\min_{i \in I}\{\frac{D_n^i - D_n^{i_n}}{R_n^i - R_n^{i_n}} | R_n^i > R_n^{i_n} \wedge D_n^i < D_n^{i_n}\}$
$\lambda_n := \frac{D_n^{j_n} - D_n^{i_n}}{R_n^{j_n} - R_n^{i_n}}$
end while

Figure 5: Algorithm for direct computation of the convex hull.

4 Results

In this section, we provide some coding results of our new codec and comparisons with the tmn codec [18]. The experiments are made with QCIF frames and a codebook size of $n_1 = 64$ and $n_2 = 512$. 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. As usual, the peak signal to noise ratios (PSNR) are computed using only the luminance component.

Figure 6a and 6b 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 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. 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 7. The left figure shows the *Mother and Daughter* sequence. 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. A similar observation can be made for the *salesman* sequence in Figure 7b. After 100 frames the average PSNR value gap of the remaining frames is around 1.1 db. This shows the adaptability of our approach that needs more than 100 frames to evolve. Note that we are not using motion compensation.

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. RD-optimization techniques and performance considerations were presented. These considerations lead to the result that for our encoding context the BFOS based optimization technique performs better than the Lagrangian multiplier approach. Future work will reconsider this issue in a more general setting using a model for the distribution of RD-points. It was shown that the new codec improves our previously published AVQ-codec. However, a comparison with standard transform coding (H.263) shows that inspite of the improvements 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 and our future work will investigate the combination of our AVQ-approach with motion compensation. In fact, in current tests we are using AVQ to encode motion prediction residual frames and first results indicate that the performance may improve to the level of standard techniques based on the discrete cosine transform (DCT) and beyond.

References

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

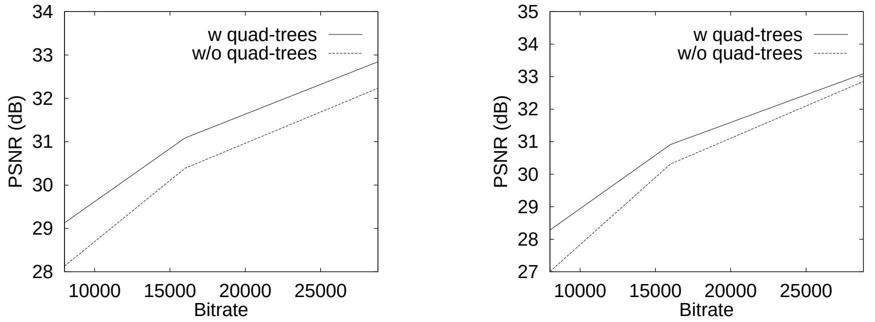


Figure 6: Average PSNR for (left) *Mother and Daughter* and (right) *salesman*.

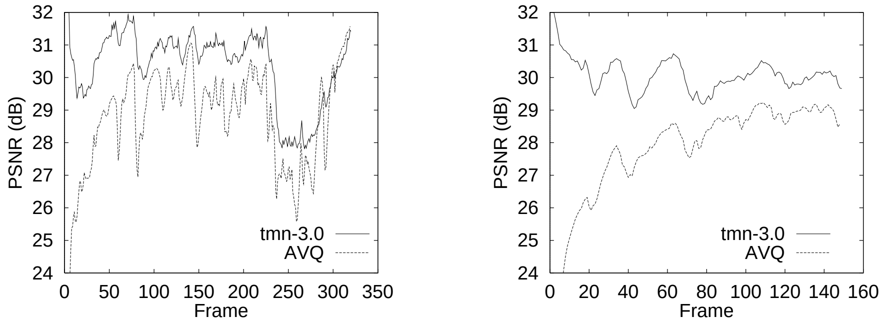


Figure 7: PSNR course for (left) *Mother and Daughter* and (right) *salesman* for the tmn and the AVQ-codec at 8000 bit/s.

Publishers, 1992.

[2] M. Goldberg and H. Sun. Image sequence coding using vector quantization. *IEEE Trans. Commun.*, COMM-34(7):703–710, July 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. Poster abstract in proc. of DCC'99. Full paper available as Technical Report 119, `ftp://ftp.informatik.uni-freiburg.de/documents/reports/report119/ report00119.ps.gz`.

[9] 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.

[10] 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.

[11] Y. Shoham and A. Gersho. Efficient bit allocation for an arbitrary set of quantizers. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 36(9):1445–1453, September 1988.

[12] A. Ortega and K. Ramchandran. Rate-distortion methods for image and video compression. *Signal Processing Magazine*, 15(6), November 1998.

[13] H. Everett III. Generalized lagrange multiplier method for solving problems of optimum allocation of resources. *Operations Research*, 11:399–417, 1963.

[14] P.H. Westerink, J. Biemond, and D.E. Boekee. An optimal bit allocation algorithm for sub-band coding. In *Proc. ICASSP'88*, pages 757–760, 1988.

[15] K. Ramchandran and M. Vetterli. Best wavelet packet bases in a rate-distorion sense. *IEEE Trans. Image Processing*, 2(2):160–175, April 1993.

[16] G. J. Sullivan and R. L. Baker. Efficient quadtree coding of images and video. *Trans. image proc.*, 3(3):327–331, May 1994.

[17] G. J. Sullivan. *Low-Rate Coding of Moving Images Using Motion Compensation, Vector Quantizatin and Quadtree Decomposition*. PhD thesis, University of California, Los Angeles, 1991.

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