First publ. in: Mathematics, key technology for the future : joint projects between universities and industry / Willi Jäger ... (eds.). - Berlin [u.a.] : Springer, 2003. - S. 520-530. - ISBN 3-540-44220-0

Video Coding with Adaptive Vector Quantization and Rate Distortion Optimization

M. Wagner[1] and D. Saupe[2]

[1] Institut für Informatik, Universität Freiburg, Am Flughafen 17, 79110 Freiburg
[2] Institut für Informatik, Universität Leipzig, Augustusplatz 10/11, 04109 Leipzig

Abstract. The goal of this project is the evaluation and development of a new adaptive image sequence coding approach. The codec is based on adaptive vector quantization and has been combined with several video coding techniques like wavelet transform, quad-trees, and rate distortion optimization. In addition we provide a comparison with a state of the art video codec (H.263) and describe new experiments with motion compensation. This project was supported by Micronas Intermetall GmbH.

1 Introduction

As a part of the project "Adaptive Verfahren zur Bild- und Bildsequenzkodierung", we evaluated the prospects of a new adaptive image sequence coding scheme for very low bitrates. During the project, we developed and successively improved this new coding approach. The results have been published in [1–4].

The codec is based on hierarchical adaptive vector quantization (AVQ) in the wavelet domain and, unlike standard video coding, does not apply motion compensation. The hierarchical organization of the wavelet coefficients is made by a special tree data structure called *quad-trees*[5]. To achieve optimized results, we adopt techniques from *rate distortion theory*, the mathematical foundation of lossy data compression[6–8].

Vector quantization (VQ) has been proven a powerful compression scheme for coding of images and image sequences [9,10]. 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 [11,12]. In order to improve the AVQ-codec, a rate-distortion (RD) optimization was investigated in [4,13,14]. It was shown in [1–3] that vector quantization in the wavelet domain and hierarchical organization of the wavelet coefficients leads to additional coding gains.

2 Outline of the Codec

In a preprocessing step a two times octave-band wavelet transform (9/7 biorthogonal filter [15]) 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.

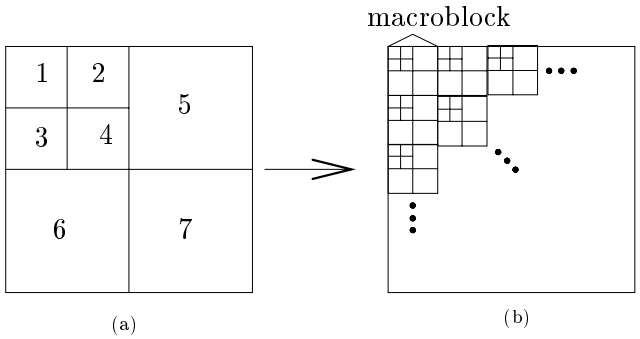

Fig. 1. Regrouping of macro blocks in the wavelet domain. (a) Subband grouping. Coefficients are grouped corresponding to their frequency. (b) Macro block grouping. Coefficients are grouped corresponding to their spatial position.

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 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.

In order to create vectors from blocks, 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

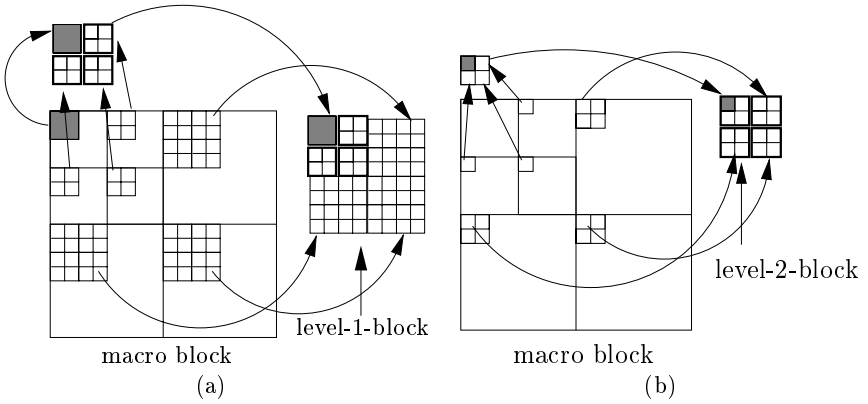

Fig. 2. Grouping of wavelet coefficients in (a) level-1-blocks and (b) level-2-blocks

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 $\mathcal{C}_1 = \{x_0^1, \ldots, x_{n_1-1}^1\}$ and $\mathcal{C}_2 = \{x_0^2, \ldots, x_{n_2-1}^2\}$ 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 [4]. 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 [8]. 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 Rate-Distortion Optimization

In the last section we discussed the encoding options that could be used in order to encode a frame. We didn't consider so far the issue of how to select these options and which quad-tree decomposition should be used. Apparently this is a constrained optimization problem. Given a frame $\mathbf{X}$ consisting of M macro blocks $(X_0, \ldots, X_{M-1})$ and a previously decoded frame (reference frame) $\mathbf{X^r} = (X_0^r, \ldots, X_{M-1}^r)$. Then the objective is to construct a representation frame $\hat{\mathbf{X}} = (\hat{X}_0, \ldots, \hat{X}_{M-1})$ by means of encoding options described in Sect. 2 in order to achieve the smallest possible overall distortion

$$D = d(\mathbf{X}, \hat{\mathbf{X}}) = \sum_{m=0}^{M-1} d(X_m, \hat{X}_m)$$

$$\text{subject to} \quad R = r(\hat{\mathbf{X}}) = \sum_{m=0}^{M-1} r(\hat{X}_m) \leq R_t, \quad (1)$$

with $d(x, y) = \| x - y \|_2^2$, $r(\hat{X}_m)$ describing the bits needed to encode macro block $\hat{X}_m$, and the target rate R_t.

Having the set of all possible encoding options $\mathcal{I}$ for a macro block, including all encoding modes for all quad-tree decompositions, we can assign to each $i \in \mathcal{I}$ and each macro block X_m an encoding rate and the corresponding distortion (R_m^i, D_m^i). Thus we can define for each macro block a set of rate distortion points $\mathcal{P}_m = \{(R_m^i, D_m^i) : i \in \mathcal{I}\}$.

The constrained minimization (1) can be transformed into an unconstrained minimization with the well known Lagrangian multiplier approach:

$$\text{minimize} \sum_{m=0}^{M-1} D_m^{i_m} + \lambda \cdot \sum_{m=0}^{M-1} R_m^{i_m}, \quad i_m \in \mathcal{I} \quad (2)$$

The problem of (2) is that we have not only to determine the optimal encoding options but also a value of λ that yield a solution meeting the rate constraint. Thus we have to iterate the minimization several times to find an optimal λ. With a fixed lambda, the minimization can be performed for each macro block independently. The solution index of (2) for macro block X_m is denoted by i_m^*.

In order to compute the solution more efficiently we make use of the geometric interpretation of the Lagrangian multiplier approach. For this we consider for each macro block X_m the set of RD-points $\mathcal{P}_m$. The solution $(R_m^{i_m^*}, D_m^{i_m^*})$ of (2) for macro block X_m can be constructed by the following procedure. A line with slope λ is drawn through the origin. Afterwards the line is shifted towards the convex hull of $\mathcal{P}_m$. The first point met is the point $(R_m^{i_m^*}, D_m^{i_m^*})$. If we consider the lower convex hulls of the sets $\mathcal{P}_m$ as "rate distortion curves" then the solution of (2) can be interpreted as a "constant slope" condition (Figure 3). This suggests to compute the solution by direct exploration of the lower convex hulls of all sets $\mathcal{P}_m$. One simple representative of this kind of algorithms is the algorithm by Westerink et al. [16]. Starting from the minimal possible rate for each $\mathcal{P}_m$, the lower convex hulls are computed for increasing rates until the target rate is reached. This works as follows: First, the smallest achievable bitrate is searched, i.e. for every macro block X_m the encoding option $i_m \in \mathcal{I}$ with the smallest rate $R_m^{i_m}$ is computed. For every macro block X_m the coding option j_m describing the next point (with a higher rate) on the lower convex hull (LCH) is determined. Let λ_m be the slope between the RD-points specified by i_m and j_m. The macro block n with the "steepest", i.e. minimal, λ_n is selected. The coding option for X_n is changed $i_n \leftarrow j_n$. Then the next point on the LCH of $\mathcal{P}_m$ is searched and the value of j_n is updated accordingly. After that, again the "steepest" λ_m is searched and so forth.

The BFOS algorithm [8] 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, we developed a more general approach without these restrictions.

Such optimization techniques can be summarized as *incremental computation of the convex hull* [2].

4 Results

In this section, we provide some coding results of our new codec and compare it with older versions to show the successive improvement during the project. In addition we present comparisons to the state of the art video coding standard for very low bitrates (H.263). The experiments are made with QCIF

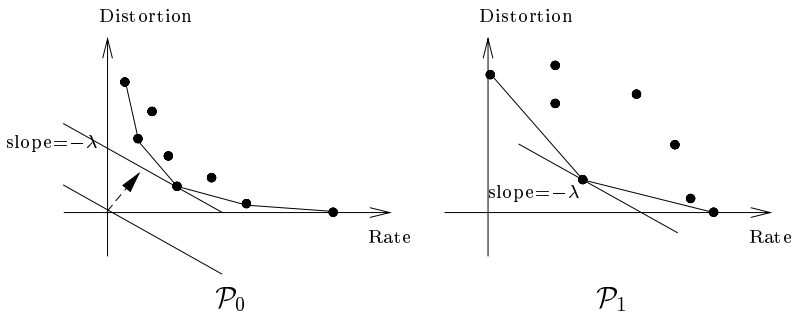

Fig. 3. Geometrical interpretation of the Lagrangian multiplier.

frames (176×144) and codebook sizes of $n_1 = 64$ and $n_2 = 512$. The codec uses a frame rate of 8.3 frames/s. As usual, the peak signal to noise ratios (PSNR) are computed using only the luminance component.

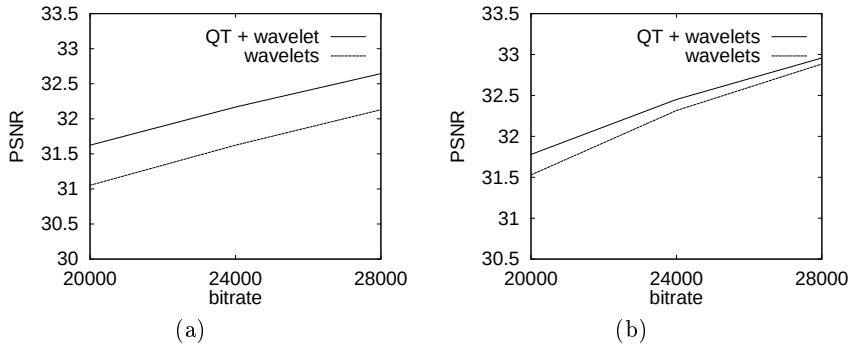

Fig. 4. Comparison of the average PSNR of the AVQ-codec with and without quad-trees for (a) *Mother & Daughter* and (b) *Salesman*.

Figure 4 shows the mean PSNR for several bitrates of our AVQ-codec and the previous version without quad-trees. It can be seen that the application of quad-trees leads to a coding gain of up to 0.5 db PSNR.

In order to show development of our codec during the project Fig. 5 is provided. This figure shows the average PSNR for several bitrates of preceding codecs. The codec with the least performance is purely based on AVQ in the spatial domain and is able to encode in real-time. The second best codec applies RD-optimization to the latter one. The best one additionally uses a wavelet transform and applies the AVQ scheme in the transform domain. Together with the improvement by using quad-trees the performance was improved up to 3.5 dB PSNR during the development.

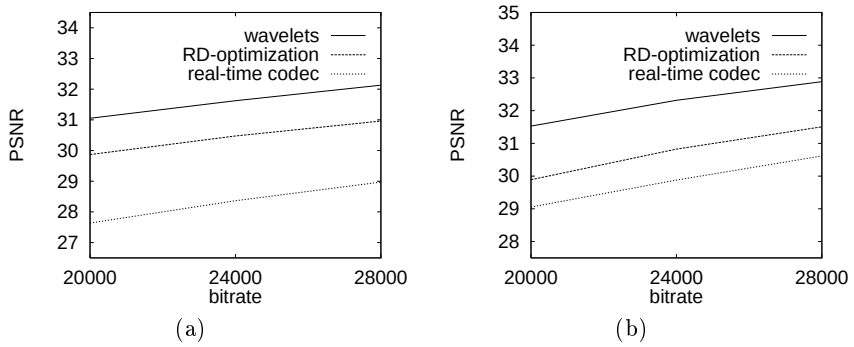

Fig. 5. Comparison of the average PSNR of the AVQ-codec for several development stages for (a) *Mother & Daughter* and (b) *Salesman*.

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 (H.263). The tmn codec (version 3.0) is used with syntax based arithmetic coding at a bitrate of 8000 bits/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 Fig. 6. Figure 6a shows the *Mother & 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 ab is about 0.9 db PSNR. A similar observation can be made for *Salesman* in Fig. 6b. This shows the adaptability of our approach that needs more than 100 frames to evolve. Note that we are not using motion compensation.

5 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 [17] 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

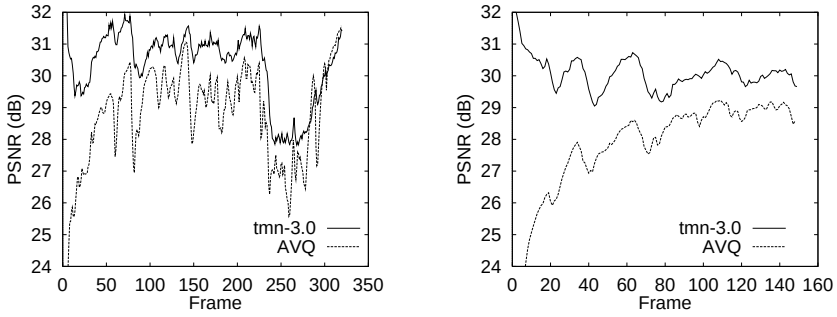

Fig. 6. PSNR course for (a) *Mother & Daughter* and (b) *Salesman* for the tmn and the AVQ-codec at 8000 bits/s

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 vector 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 7a 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 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. 7b. 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 [18]. But optimizing motion compensation would also improve the results of our AVQ-tmn codec.

6 Conclusion and Future Work

As a part of this project we have developed a new image sequence encoding scheme for very low bitrates. This scheme is based on AVQ and quad-tree coding in the wavelet domain. The codec uses an efficient RD-optimization mechanism.

The performance of the codec was successive improved during this project. However, the comparison with standard transform coding 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 experiments show promising results. Thus, for the rest of the project, we will investigate combinations between our AVQ-approach and motion compensation.

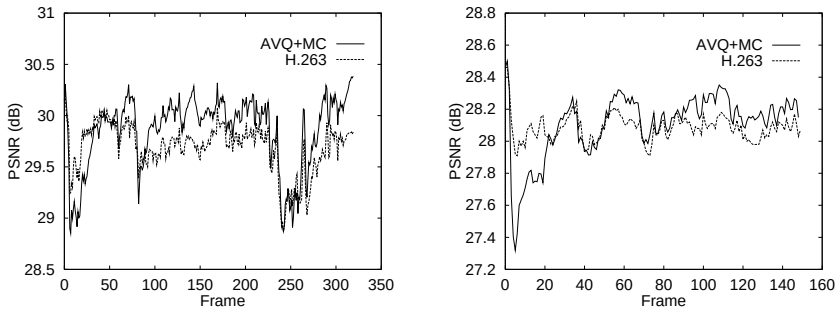

Fig. 7. Comparison of H.263 the AVQ motion compensated codec for the (a) *Mother & Daughter* and (b) *Salesman* sequence. The resulting average number of bits per frame is around 800.

References

1. M. Wagner and D. Saupe. Video coding with quad-trees and adaptive vector quantization. In *Proc. of EUSIPCO*, 2000.
2. M. Wagner and D. Saupe. Rd-optimization of hierarchical structured adaptive vector quantization for video coding. In *Proc. of DCC*, page 576, 2000. Full paper available as Technical Report 139, Institut für Informatik, Universität Freiburg.

3. 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.
4. 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.
5. H. Samet. The quadtree and related hierarchical data structures. *ACM Computing Surveys*, 16(2):188–216, June 1984.
6. A. Ortega and K. Ramchandran. Rate-distortion methods for image and video compression. *Signal Processing Magazine*, 15(6), November 1998.
7. T. Berger. *Rate Distortion Theory: A Mathematical Basis for Data Compression*. Prentice-Hall, Englewood Cliffs, NJ, 1971.
8. 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.
9. A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, 1992.
10. M. Goldberg and H. Sun. Image sequence coding using vector quantization. *IEEE Trans. Commun.*, COMM-34(7):703–710, July 1986.
11. 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.
12. 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.
13. J. E. Fowler and S. C. Ahalt. Adaptive vector quantization using generalized threshold replenishment. In *Proceedings of the 1997 IEEE Data Compression Conference*, 1997.
14. M. Lightstone and S. K. Mitra. Image-adaptive vector quantization in an entropy-contrained framework. *IEEE Transaction on Image Processing*, 6(3), March 1997.
15. 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.
16. P.H. Westerink, J. Biemond, and D.E. Boekee. An optimal bit allocation algorithm for subband coding. In *Proc. ICASSP'88*, pages 757–760, 1988.
17. Tmn 3.0 (h.263) codec. Released by the Signal Processing and Multimedia Group, University of British Columbia, `http://spmg.ece.ubc.ca`.
18. G.J. Sullivan and T. Wiegand. Rate-distortion optimization for video compression. *IEEE Signal Processing Magazine*, pages 74–90, 1998.