A Video Codec Based on R/D-Optimized Adaptive Vector Quantization[1]

Technical Report 119

Marcel Wagner*, Ralf Herz*, Hannes Hartenstein*,
Raouf Hamzaoui†, and Dietmar Saupe†

*Universität Freiburg, Am Flughafen 17, 79110 Freiburg, Germany
{wagner,herz,hartenst}@informatik.uni-freiburg.de
†Universität Leipzig, Augustusplatz 10/11, 04109 Leipzig, Germany
{hamzaoui,saupe}@informatik.uni-leipzig.de

Abstract

We present a video encoding scheme based on mean-removed vector quantization where the shape codebook is adapted to the source on a frame-by-frame basis. In contrast to previously published adaptive vector quantization (AVQ) approaches the decision how blocks are encoded and how the codebook is updated is done in a rate-distortion optimized fashion. Furthermore, we show that the translation of our rate-distortion optimized AVQ into the wavelet domain leads to an improved coding performance. We present coding results as well as an empirical analysis of the short- and long-term behavior of the adaptive codebook.

1 Introduction

Vector quantization (VQ) is known to be an optimal compression method for a stationary and ergodic source, provided that arbitrarily large vector dimension and codebook size are employed [1]. In practical VQ encoding schemes of images and video sequences, hardly one of these assumptions is fulfilled; the dimension and codebook are limited in size and, perhaps the worst, video sources are not stationary. Therefore, adaptive VQ (AVQ) schemes have been developed that adapt the vector quantizer to the source

[1]Presented at IEEE Data Compression Conference 1999, Snowbird, Utah.

during the encoding [1]–[6]. In this paper we present a new AVQ-based video coder for very low bitrates. The coder is rate-distortion optimal in the setting of frame replenishment with mean-removed VQ where the codebook of shape blocks is updated once per frame. First results for an implementation of such a scheme have been reported in our previous paper [5]. Here we extend the method to incorporate a wavelet image transform before coding in order to enhance the compression performance. We offer an empirical analysis of the coder showing that with the long-term memory inherent in AVQ one can achieve the same encoding quality as with comparable standard transform coding ($H.263$).

2 Outline of the codec

In this section we give a brief outline of the AVQ-codec as proposed in [5]. We begin with reviewing the basic AVQ terminology.

A vector quantizer is based on a codebook $\mathcal{C} = \{c_1, ..., c_s\}$ with s vectors $c_i \in \mathbb{R}^N$, the index set $\mathcal{I} = \{1, ..., s\}$ and a variable-length-code (VLC) given by a set $\mathcal{V}$ of s codewords; the specification of the vector quantizer is completed by giving the functions α, β, γ where $\alpha : \mathbb{R}^N \to \mathcal{I}$ maps a vector to an index, $\beta : \mathcal{I} \to \mathcal{C}$ maps an index to a vector of $\mathcal{C}$, and $\gamma : \mathcal{I} \to \mathcal{V}$ maps an index to a VLC-word. The VQ-encoding $\mathcal{E} : \mathbb{R}^N \to \mathcal{V}$ is given by $\gamma \circ \alpha$, and the decoding $\mathcal{D} : \mathcal{V} \to \mathcal{C}$ by $\beta \circ \gamma^{-1}$.

In an *adaptive* VQ-scheme the aforementioned parameters depend on time $t \in \mathbb{N}$, i.e., at time t vector quantizer $(\mathcal{C}_t, \mathcal{I}_t, \mathcal{V}_t, s_t, N_t, \alpha_t, \beta_t, \gamma_t)$ is used. The codebooks $\mathcal{C}_t$ are required to be subsets of a *universal codebook* $C^\star$, i.e., $\forall t \in \mathbb{N} : \mathcal{C}_t \subseteq \mathcal{C}^\star$. Finally one has to specify *update rules* which govern the changes of the AVQ from time t to $t+1$. For ease of notation we drop the index t where it is not essential.

In the following we describe the core of our AVQ-based video coder. Let the video source be described by a sequence of N-dimensional vectors that are obtained by splitting the frames into $n \times n$ pixel blocks, i.e., $N \equiv n^2$, and scanning blockwise row by row. In our experiments we have set $n = 4$. We denote the current frame by $X = (x_0, ..., x_{l-1})$ with $x_i \in \mathbb{R}^N$; l denotes the number of blocks in a frame. The previously decoded frame is denoted by $X^r = (x_0^r, ..., x_{l-1}^r)$. It is called the the *reference frame* and known to the encoder as well as the decoder

Now, we encode X by creating a frame $\hat{X} = (\hat{x}_0, ..., \hat{x}_{l-1})$ with vectors of the reference frame X^r (mode 0), the current codebook $\mathcal{C}$ (mode 1) and the universal codebook $\mathcal{C}^\star$ (mode 2). Each vector x_k in the current frame X can thus be encoded in one of the following ways:

Mode 0 Vector x_k is encoded by x_k^r, i.e., $\hat{x}_k := x_k^r$.

Mode 1 Vector x_k is encoded with mean removed vector quantization [1]. The mean m_k of x_k is computed and quantized to $\hat{m}_k = Q_m(m_k)$, where Q_m is a uniform scalar quantizer for the mean values; then x_k is encoded by $\hat{x}_k := \beta \circ \alpha(x_k - \hat{m}_k \mathbf{1}_N) + \hat{m}_k \mathbf{1}_N$.

Mode 2 Vector x_k is encoded with a proper vector $c_k^\star \in \mathcal{C}^\star$ and the quantized mean $\hat{m}_k = Q_m(m_k)$ of x_k, i.e. $\hat{x}_k := c_k^\star + \hat{m}_k \mathbf{1}_N$.

Here, $\mathbf{1}_N$ denotes the vector $(1, ..., 1)$ of dimension N. The decision which mode to choose for each block should be made in a r/d-optimal fashion; see Section 3. The resulting modes are collected in a mode map $M = (M_0, ..., M_{l-1}), M_k \in \{0, 1, 2\}$ that is sent to the decoder using run length encoding. For each vector x_k in mode 1 or 2 additional information has to be sent. In both cases we send an index for the quantized mean $Q_m(m_k)$. If x_k is encoded in mode 1 we also send an index of $\mathcal{C}$, otherwise a vector in the universal codebook. In fact, instead of sending an index of $\mathcal{C}^\star$, we transmit x_k with differential pulse code modulation (DPCM) and use a uniform scalar quantizer Q_d to quantize the residuals. The universal codebook, thus, consists of all vectors $x \in \mathbb{R}^N$ after DPCM transmission and quantization, i.e., $\mathcal{C}^\star \subsetneqq \mathbb{R}^N$.

When the encoding of the frame is finished, we have to update the AVQ. The update rules we use are heuristically determined. The intention is to drop the vectors in $\mathcal{C}_t$ that are seldom used and replace them by new vectors that were transmitted in mode 2. In addition, when we assign the variable length code (VLC) $\mathcal{V}_t$ to the vectors c_i in $\mathcal{C}_t$, we would like to assign the shortest codes in $\mathcal{V}_t$ to the most frequent vectors in $\mathcal{C}_t$. Thus, we introduce a frequency count $f_t : \mathcal{C}_t \to \mathbb{N}$ that is responsible for the order of the codebook. The codebook is adapted to satisfy $i < j \Rightarrow f_t(\beta_t(i)) \geq f_t(\beta_t(j))$. This means, the most frequent vector in $\mathcal{C}_t$ has the lowest index and the least frequent vector has the highest index. Altogether, the codebook update operates in four steps: 1. The frequency counts are updated accounting for the use of codebook vectors in the current frame. 2. The least frequently used vectors are identified. 3. These vectors are replaced by the vectors from the universal codebook that were transmitted for the current frame. 4. The new codebook is sorted with respect to the frequency counts, (see [5] for more details). Note that neither the dimension, nor the codebooksize changes. So we have $\mathcal{I}_t = \mathcal{I}_{t+1}$, $s_{t+1} = s_t$ and $N_{t+1} = N_t$.

The pair $(\mathcal{V}, \gamma)$ as well as the VLC-encoding of the quantization indices from Q_m and Q_d can be realized by Huffman coding. However, it is convenient to replace Huffman coding by adaptive arithmetic coding. The distribution of the codebook indices and the indices of the scalar quantizers is estimated with the corresponding frequencies from the previously encoded frames.

So far the given encoding process considers only the Y-component of YUV-frames. To also encode the UV-components, the codec sends with every vector x_k encoded in mode 1 or 2 an average U- and V-value until a certain bit budget is used up. Our encoder spends $\frac{1}{10}$th of the given bit budget on the UV-components.

3 Rate-distortion optimization

3.1 Basic rate-distortion AVQ

In classical AVQ schemes the quantization functions α_t select for a given vector $x \in \mathbb{R}^N$ the nearest neighbor in the codebook C_t, i.e., $\alpha_t(x) = \arg\min_{i \in \mathcal{I}} d(x, \beta_t(i))$ where $d(x, y)$ denotes the squared L_2-distance $\|x - y\|_2^2$. Thus, the decision for a codebook vector solely depends on the resulting distortion. In addition, the decision whether a codebook update is necessary is based only on the distortion $d(x, c^*)$ with $c^\star \in \mathcal{C}_t$ the nearest neighbor of x. If $d(x, c^\star)$ is larger than a certain threshold, vector x is inserted in the codebook [2].

In rate-distortion based AVQ the selection is based on a combination of the resulting rate and distortion, i.e., it is based on the Lagrangian cost function

$$J^i_{\lambda,t}(x) = d(x, \beta_t(i)) + \lambda \cdot |\gamma_t(i)| \tag{1}$$

Here, $\lambda \in \mathbb{R}^+$ denotes the Lagrangian multiplier, and $|\gamma_t(i)|$ is the rate (codeword length) of codeword $\gamma_t(i)$. In the following we simply write J^i_λ when we assume a fixed x and t. In order to find the 'best' codebook entry for a fixed value of λ, the quantization function α_t of an RD-based AVQ selects for $x \in \mathbb{R}^N$ codebook entry $\alpha_t(x) = \arg\min_{i \in \mathcal{I}}[J^i_\lambda(x)]$. To decide, whether the codebook has to be updated, one also needs a function $J_\lambda^\Delta(x)$ that indicates whether an update pays off in terms of rate versus distortion performance. Such an updating criterion can consider the impact of an codebook update on the coding of future vectors, as it has been proposed in [4], or it can be online as in the generalized threshold replenishment (GTR) method of Fowler [3].

In the GTR-scheme one determines $\min[\{J_\lambda^i\}_{i=1}^s \cup \{J_\lambda^\Delta\}]$ for a fixed (estimated) value of λ. This can be interpreted geometrically as follows [7]: Every pair in $\mathcal{P} = \{(d(x, \beta_t(i)), |\gamma_t(i)|)\}_{i=1}^s \cup \{(d(x, x^\Delta), r^\Delta)\}$, where x^Δ is a proper vector of $\mathcal{C}^\star$ and r^Δ the rate to transmit its index, is a point in the rate-distortion plane (see Figure 1). To find the point according to $\min[\{J_\lambda^i\}_{i=1}^s \cup \{J_\lambda^\Delta\}]$, one draws at first a line with slope $-\lambda$ through the origin of the plane. Then, one shifts this line towards the convex hull of $\mathcal{P}$ without changing the slope. The first point met is the desired point (see Figure 1). Of course, in the case when multiple points are met, a tie breaking rule is needed. The multiplier λ can be interpreted as the negative "slope" of the (operational convex) rate-distortion curve.

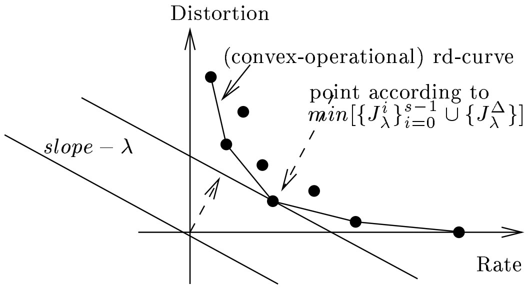

Figure 1: Rate-Distortion Plane.

The problem of these "lambda-based" rate-distortion optimizations is to determine the lambda that corresponds to a given bitrate. Since the rate-distortion curve is usually not known in advance, the lambda is often somehow estimated or iteratively optimized. Both approaches have disadvantages: The estimation approach can lead to a poor performance if the lambda is not a good estimation for the negative "slope" of the rate-distortion curve. The iterative optimization approach is very time consuming because one has to repeat an already complex rate-distortion optimization for several lambdas in order to find the best one. In our approach, we directly use the convex hull of the rate-distortion points for the rd-optimization. We determine the optimal lambda with a procedure that is less complex than an iterative optimization.

3.2 R/D-optimized AVQ with bitrate control

The Lagrangian cost functions for the different modes are given by

Mode 0 $J^0_\lambda(x_k) = d(x_k, x^r_k)$

Mode 1 $J^i_\lambda(x_k) = d(x_k, \beta_t(i) + \hat{m}_k\mathbf{1}_N) + \lambda\left(|\gamma_t(i)| + |\hat{m}_k|\right)$, $i = 1, \ldots, s$

Mode 2 $J^{s+1}_\lambda(x_k) = J^\Delta_\lambda = d(x_k, x^\Delta_k + \hat{m}_k\mathbf{1}_N) + \lambda(r^\Delta + |\hat{m}_k|)$

Here $|\hat{m}_k|$ denotes the rate needed to encode mean $\hat{m}_k$. Now, if the lambda is given, we can decide in which mode M_k a vector x_k has to be encoded by computing $\min_{i=0}^{s+1} J^i_\lambda(x_k)$ for every x_k independently. In fact, this method would be very similar to the online algorithm by Fowler [3] with an additional inter-frame mode (mode 0). But, in contrast to Fowler's GTR scheme, we compute the optimal lambda for a given bitrate with the bit allocation algorithm by Westerink *et al.* [8]. With this approach the codebook is only changed after a whole frame is encoded. Thus, we apply the update-rules of our AVQ only when the encoding of a frame has been finished. The algorithm works as follows: Every mode gives us one or more points on the rate-distortion-plane:

Mode 0: $(r^0_k := 0, d^0_k := d(x_k, x^r_k))$

Mode 1: $\{(r^j_k := |\gamma_t(j)| + |\hat{m}_k|, d^j_k := d(x_k, \hat{m}_k\mathbf{1}_N + \beta_t(j))\}_{j=1}^{s}$

Mode 2: The rate r^Δ_k is needed to transmit $x_k - \hat{m}_k$ with DPCM. That sums up to the rate $r^{s+1}_k = |\hat{m}_k| + r^\Delta_k$. The received vector x^Δ_k introduces the distortion $d^{s+1}_k = d(x_k, \hat{m}_k\mathbf{1}_N + x^\Delta)$.

For every block x_k from the current frame we get an independent rate-distortion plane with $s+2$ points and, thus, an (operational convex) rate-distortion curve. We start the rate-distortion optimization by assuming that all vectors x_k are encoded in mode 0 and with rate 0. These points lie always on the lower bound of the convex hull, because all the other points have a larger rate. The aim of the following algorithm is to analyze the (operational convex) rate-distortion curves and to choose those points that give the highest gain in rate-distortion sense. When the target bitrate is achieved, the algorithms stops.

INITIALIZATION: for all vectors x_k, $0 \leq k < l$, of a frame: set mode $M_k := 0$, rate $r_k := 0$, distortion $d_k := d^0_k$ and VQ-index $i_k := 0$. Set $\lambda := \infty$.
PREPROCESSING:
for all $k = 0, \ldots, l-1$ **do**

for all j such that $d_k^j < d_k$ and $1 \leq j \leq s+1$ **do**
compute $\lambda_k^j := \frac{d_k^j - d_k}{r_k^j - r_k}$
end for
find $j_k := \arg\min_{j=1}^{s+1} \lambda_k^j$.
end for
OPTIMIZATION:
while $rate < target\_rate$ **do**
find $u := \arg\min_{k=0}^{l-1} \lambda_k^{j_k}$
if $j_u < s$ **then** $M_u := 1$ **else** $M_u := 2$
$\lambda := -\lambda_u^{j_u}$, $i_u := j_u$, $r_u := r_u^{j_u}$, $d_u := d_u^{j_u}$, $rate := \sum_{k=0}^{l-1} r_k$
for all j such that $d_u^j < d_u$ and $1 \leq j \leq s+1$ **do**
compute $\lambda_u^j := \frac{d_u^j - d_u}{r_u^j - r_u}$
end for
find $j_u := \arg\min_{j=1}^{s+1} \lambda_u^j$.
end while

The algorithm terminates because *rate* increases every time it is recomputed. When the algorithm stops, we know not only the optimal lambda λ, but also the mode map M and, if necessary, the index i_k of $\mathcal{C}$ for mode-1 encoding of x_k. Thus, the encoding procedure is completely specified by the above algorithm.

4 AVQ in the wavelet domain

Many of the image and most of the video compression algorithms make use of transform coding. Normally, a decorrelating transform is applied that concentrates the signal energy in a few components. This property is then exploited in subsequent quantization and VLC-encoding. Theoretically, a vector quantizer is able to encode decorrelated vectors more efficiently than a scalar quantizer because of the so-called sphere-packing property; moreover a vector quantizer is able to exploit higher order statistical dependencies [1]. Thus transform coding with VQ theoretically outperforms scalar quantization schemes. But we expect much more than only exploiting higher order statistics from combining these schemes. Since we deal with an adaptive vector quantizer, we have to transmit the codebook updates. In Section 3, we did this by DPCM coding in the spatial domain. If we use transform coding, transmission of vectors can be more efficient.

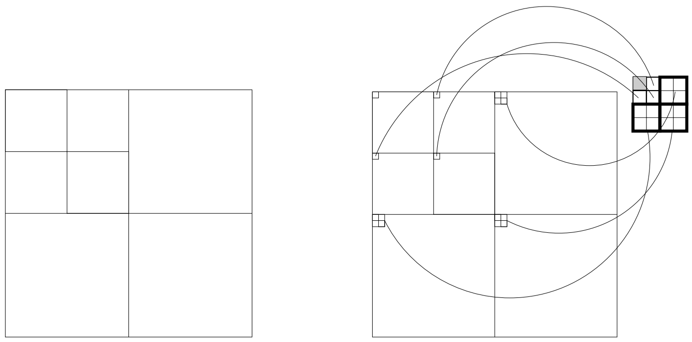

Figure 2: Left: Subband decomposition. Right: Grouping of wavelet coefficients

In connection with adaptive VQ and frame replenishment the wavelet image transform (see, e.g., [9]) offers advantages because of its good localization in frequency *and* space. As in Section 2, we would like to have a sequence of vectors (of wavelet coefficients) that belong to the spatial 4×4-blocks, scanned from the upper-left to the bottom-right of the frame. The frame is decomposed with a 2-times octave-band decomposition (see Figure 2) using the 9/7-filters of [9]. The coefficients of the subbands that belong to a certain location in the original frame, are grouped into 15-dimensional vectors (Fig. 2). The lowpass coefficient is treated like the mean in Section 3. After this, the AVQ-video proceeds as specified in Section 3. Of course, the dimension N_t is now 15. Since the components of the vectors of wavelet coefficients are less correlated, we transmit the components of the vector no longer with DPCM. Instead, we quantize every component independently with the scalar quantizer Q_d.

5 Results

In our implementation the encoder processes every 3rd frame of some test sequences which are assumed to have an original frame rate of 25 frames per second (fps). Thus, we have a frame rate of about 8.3 fps. The coder assigns a constant number of $\frac{bitrate}{8.3}$ bits to each frame. The picture format is QCIF, thus, there are 1584 blocks of size $N = 4 \times 4$ per frame. The codebooksize s is 512. The encoding starts with an empty codebook, i.e., all vectors in the codebook are zero vectors. For the AVQ-codec without wavelet-transform we use the scalar quantizer for the means, Q_m, with bin

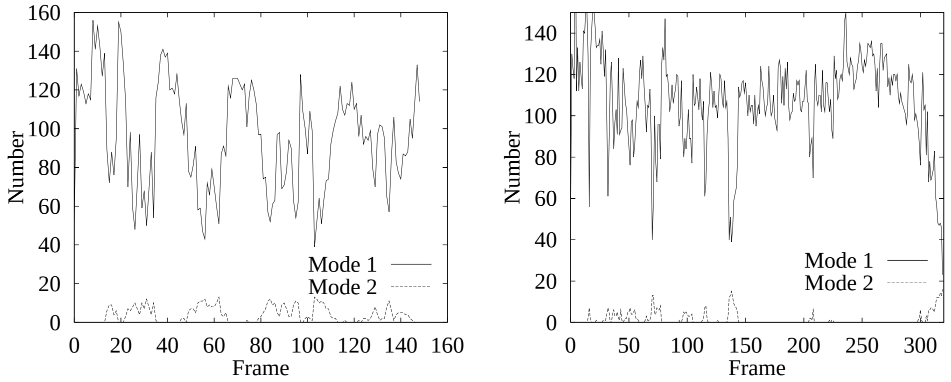

Figure 3: Mode statistics at 16 kb/s. Left: *Salesman* Right: *Mother and Daughter*

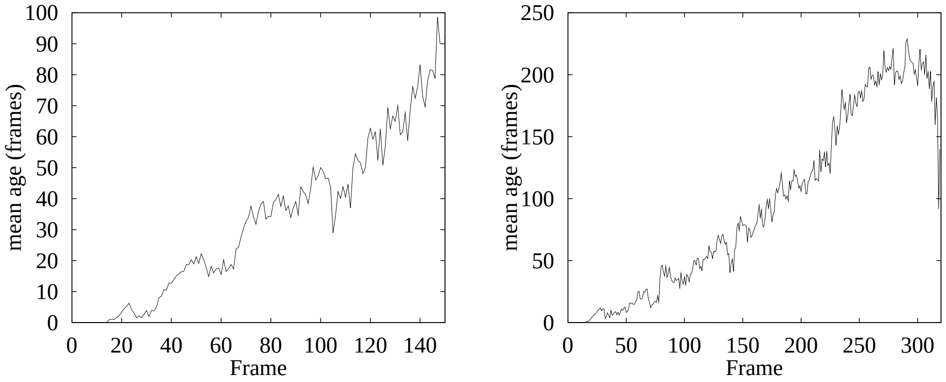

Figure 4: Mean age of mode 1 vectors. Left: *Salesman*. Right: *Mother and Daughter*

size 4 and the quantizer for the DPCM encoding, Q_d, with bin size 8. For the codec with wavelet-transform, the bin sizes are $Q_m = 8$ and $Q_d = 16$.

First, we analyze the adaptivity of the rate-distortion optimized AVQ. For this, we investigate the change of the codebook in time. The statistics of the number of encoded vectors in mode 1 and mode 2 is shown in Figure 3 for the *Mother and Daughter* and the *Salesman* sequences at a bitrate 16000 bits/s.

The curve describing the number of vectors encoded in mode 2 shows that the AVQ does not change its codebook continuously; there are phases in which the codebook does not change. The existence of such phases indicates that the vectors inserted in previous frames are often useful for the encoding

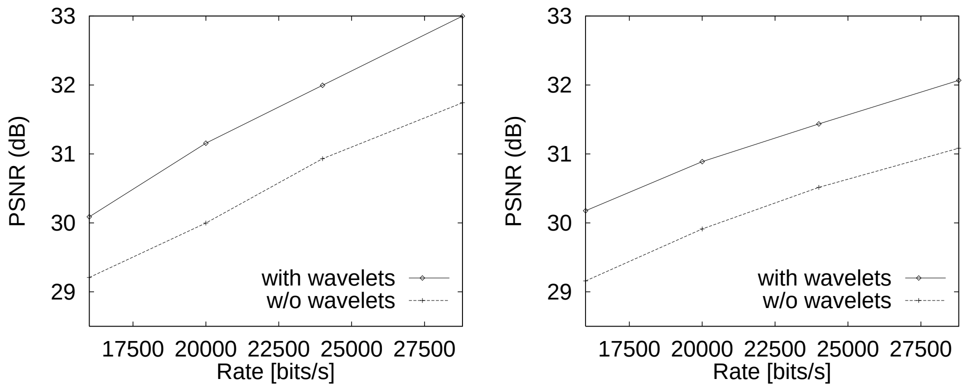

Figure 5: Average PSNR. Left *Salesman*, right *Mother and Daughter*

of subsequent frames. This causes the question whether the AVQ creates codebooks that are only suitable for the current and some next frames or whether the AVQ-coder rather creates a global codebook that covers the general characteristics of the sequence. To answer this question we take a deeper look at the vectors that are used for mode 1 encoding. We give all vectors in the codebook a time-stamp (frame number at insert time) and compute for each frame the average age of its mode-1 encoded blocks, see Figure 4. For both sequences tested the average codebook vector age is rising. Even vectors that were inserted more than 100 frames in the past, are used for mode 1 encoding. This suggests, that the AVQ-codec builds a codebook that covers the characteristics of the sequence rather than creating codebooks that are useful only for the following frames. Thus, the frames are encoded with a *long-term codebook*.

Now the coding gain of the wavelet-transform is considered. In Figure 5 the average PSNR-values for the test-sequences *Mother and Daughter* and *Salesman* are shown for several bitrates between 16000 bits/s and 28800 bits/s. The first 20 frames are not taken into account for the PSNR calculation, in order to give the codec time to adapt to the test-sequences. As expected, the AVQ-encoding performs better with the wavelet-transform than without it. Where does this gain come from? For an explanation, the average costs of the vectors in mode 1 (m1) and 2 (m2) are considered. The results are shown in Table 1 for the *Mother and Daughter* and *Salesman* sequence at the bitrates 28800 bits/s and 16000 bits/s. The computation of the average discards the first 20 frames, in order to bypass the initialization phase. With respect to our expectations outlined in Section 4 it is not surprising

Bitrate	w/o wavelets				with wavelets			
	Salesman		Mthr. & Dotr.		Salesman		Mthr. & Dotr.	
	m1	m2	m1	m2	m1	m2	m1	m2
16000	5.30	35.93	3.81	16.61	4.45	19.42	2.56	9.47
28800	6.60	47.49	5.60	30.30	6.25	27.06	4.21	19.38

Table 1: Average costs for different modes.

that for mode 2 encodings less bits are used in the wavelet-transform codec than in the spatial domain codec. But, moreover, we can observe that we are also able to encode in mode 1 with less costs. This shows that the wavelet domain vectors have a suitable statistical behavior for vector quantization.

Unfortunately only a small number of results have been reported for AVQ coding of video sequences so far. Thus, we cannot provide fair comparisons with other AVQ approaches. To give the reader something at hand to assess the performance of our AVQ codec, we compare it with a slightly modified version of UBC's tmn-3.0 $H.263$ encoder. Since, we are not using motion compensation yet, but we are using a weak form of inter frame coding by mode 0 encoding, we permit the tmn encoder only doing the same. Therefore the tmn motion estimation is employed with search range zero. Further, the tmn-encoder uses its offline rate-control mode, in order to improve its rate-distortion performance. Figure 6 compares these coders. For *Salesman*, the tmn encoder has the most time the better performance, but it is striking that the AVQ-codec is gaining on it; at the end, our AVQ codec is within a small fraction of a decibel PSNR. For the *Mother and Daughter* sequence the AVQ-codec provides the better performance after some adaption time (cf. Figure 6). Note that the AVQ-codec achieves its best performance for *Mother and Daughter* after 150 frames, more frames, than the *Salesman* sequence has at all. Presumably, the *Salesman* sequence is too short for the AVQ-codec to develop its long-term capability. The results show that our AVQ-codec is able to meet the results obtained with other transform-based codecs.

6 Conclusion

In this paper we have presented a video codec based on rate-distortion optimized adaptive vector quantization. This codec adapts its codebook on a frame by frame basis and creates a codebook that gathers the most char-

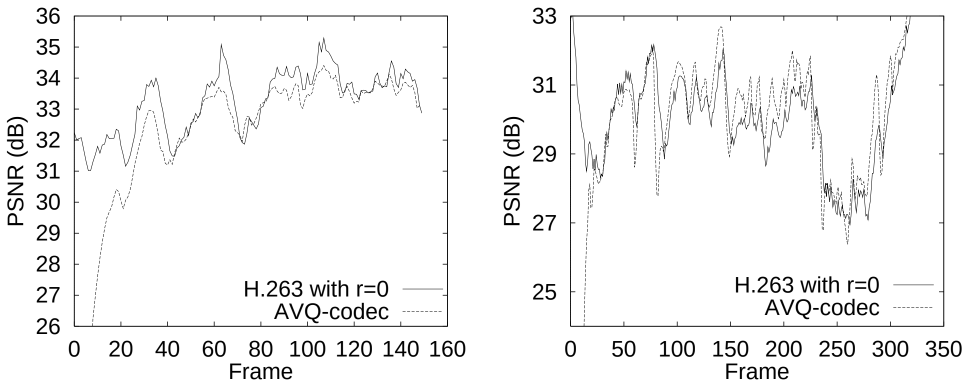

Figure 6: PSNR values for the wavelet-AVQ-codec and the tmn encoder with search range 0. Left *Salesman* with 28800 bits/s, right *Mother. and Daughter* 16000 bits/s.

acteristic vectors of the whole tested sequences. To improve the encoding performance, we translated the AVQ in the wavelet domain; experimental results show that our vector representation of the transform coefficients works well with vector quantization.

Future work will focus on a detailed analysis of the short- and long-term behavior of the codebook. Another interesting investigation will be to combine our approach with motion compensation and adaptive subdivision techniques.

References

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

[2] 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.

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

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

[5] 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.

[6] 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.

[7] G. M. Schuster and A. K. Katsaggelos. *Rate-Distortion Based Video Compression.* Kluwer Academic Publishers, 1997.

[8] 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.

[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.