ENTROPY-CONSTRAINED DESIGN OF QUADTREE VIDEO CODING SCHEMES

T. Wiegand, M. Flierl, and B. Girod

University of Erlangen-Nuremberg, Germany

ABSTRACT

The variable length code design of a complete quadtree-based video codec is addressed, in which we jointly optimize the entropy-coding for the parameters of motion-compensated prediction together with the residual coding. The quadtree coding scheme selected for this optimization allows easy access to the rate-distortion costs, thus making it possible to perform rate-distortion optimized bit allocation without exhaustive computation. Throughout the paper, we view the quadtree coder as a special case of tree-structured entropy-constrained vector quantization and derive a design algorithm which iteratively descents to a (locally) optimal quadtree video codec. Experimental results evaluate the performance of the proposed design algorithm.

1 INTRODUCTION

In past years, numerous standards such as H.261, MPEG-1, and MPEG-2 have been introduced to address the compression of video data. The most recent standard H.263 [1] is targeting low bit-rate applications. With regards to efficient realization, a key problem for H.263 as well as the other standards is the operational control of the encoder. To improve overall encoder performance, rate-distortion theory has been successfully applied to optimize the operational control of the encoder, e.g. see [2, 3]. All these approaches suffer from the fact that the associated complexity required for the computation of the rate-distortion costs in the optimization process is far too demanding for real-time implementations.

Recently, in the H.263+ and MPEG-4 standardization process, a video coding scheme has been introduced, which permits a comparably easy access to all rate-distortion costs in the encoding process [4]. The video coder presented is based on quadtree-structured variable block size motion-compensated prediction. In contrast to the existing standards the encoding in the quadtree codec [4] is performed within one step wherein the motion compensation and an intensity update between the image segments to be encoded is performed jointly, distinguishing this scheme from quadtree coders known from literature [5, 6].

The variable length code design for the video standards is performed by measuring statistics for parts of the video algorithm without referring to the entire scheme or the bit allocation. This may lead to degraded rate-distortion performance. In order to design the quadtree codec, we utilize an iterative algorithm which takes into account the complete codec and the bit allocation.

2 THE QUADTREE CODEC

The quadtree structure is a means for indicating the use of various quantizers that have been adaptively chosen by the video encoder for the corresponding image segments. However, the tree structure we associate to the coder presented in [4] is different to the tree structures described in [5, 6] or [7]. Therefore, we will first describe the video codec and later relate the tree structure to it. A block diagram showing the structure of the quadtree codec is depicted in Fig. 1.

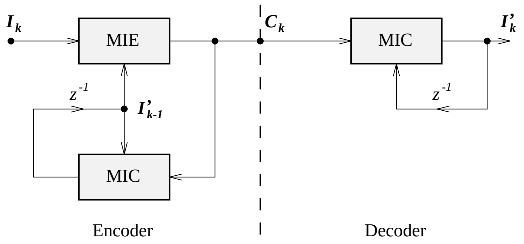

Figure 1: The parameters, i.e., the displacement vectors and intensity updates, are estimated using the motion and intensity estimation (MIE) at the encoder's side. These parameters together with the quadtree split information are transmitted to the decoder. The encoded image is reconstructed using the motion and intensity compensation (MIC).

Assuming a translational motion model, motion estimation for an image segment is often regarded as finding a segment in the previously decoded frame using some distortion measure, such as the sum of the squared differences of the two sets of samples. In contrast, the motion estimation in the scheme presented in [4] is performed by first computing the quantized difference between the mean intensity of the segment to be encoded and the mean intensity of the segment indicated by the displacement vector. With that, distortion for the motion estimation is computed as the sum of the squared distances of the two sets of samples while subtracting their quantized mean difference. In the following, we will call the quantized difference of the means the *intensity update*.

At this point, the encoding is done for the quadtree coder. The displacement vectors and the intensity updates related to image segments are the only informations that are transmitted to the decoder. With that, we have comparably easy access to the distortion generated by the quadtree encoding. Hence, it is easy to assess that the computation of the overall distortion for video coders such as H.263 which employ motion-compensated DCT is far more demanding.

The segmentation of an image is conducted as follows. For instance, a QCIF-frame is divided into 99 macroblocks each comprising image blocks of size 16 × 16 samples for the luminance component (Y) and 8 × 8 samples for the color components (Cb and Cr).

Each macroblock may be assigned a displacement vector, which is applied to the Y, Cb, and Cr components simultaneously. Whether or not the displacement vector is assigned the macroblock or a segment of it is specified using *shape* codes. The shape code indicates exactly which of the four quadrants will receive further information. This way the macroblock may be split into zero to four quadrants and the remaining part, which, if it exists, must be assigned a displacement vector. The splitting is conducted recursively to each quadrant until the smallest block size that is 2 × 2 samples in the QCIF case is reached. The set of admissible shapes for a macroblock or quadrants except quadrants of smallest block size is depicted in Fig. 2. Quadrants of smallest block size must only use shape 0.

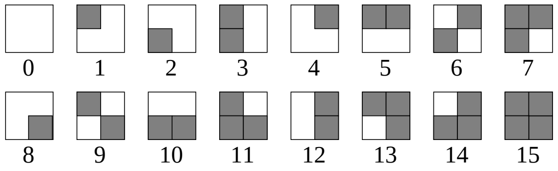

Figure 2: Set of admissible shapes 0 to 15. The shaded areas are quadrants of the block which receive further information.

Segments of smaller quadrants than the macroblock, for which the shape code indicates no split, are assigned a displacement vector and additionally a intensity update. Apart from the macroblock displacement vector, each component Y, Cb, and Cr is associated different split information, displacement vectors and intensity updates. For a detailed description of the quadtree codec see [4].

3 BIT ALLOCATION

Before explaining how we relate tree structures to the macroblock structure let us first introduce some necessary notation. Inspired by Chou et al. [7] we adopt the following definitions. A *tree* $\mathcal{T}$ is defined as a *finite set of nodes* $\mathcal{T} = \{t_0, t_1, t_2, \cdots\}$ with a root node $t_0 = \text{root}(\mathcal{T})$. The tree refers to the nodes *and* their structured relation. The subset of *terminating nodes* of a tree $\mathcal{T}$ is denoted by $\tilde{\mathcal{T}} \subseteq \mathcal{T}$. Terminating nodes are nodes that are no root node of trees except singletons, i.e., trees of height 0. The subset $\dot{\mathcal{T}} \subseteq \mathcal{T}$ with $\tilde{\mathcal{T}} \cup \dot{\mathcal{T}} = \mathcal{T}$ is called the *interior of the tree* $\mathcal{T}$. The sets $\dot{\mathcal{T}}$ and $\tilde{\mathcal{T}}$ are disjunct, i.e., $\tilde{\mathcal{T}} \cap \dot{\mathcal{T}} = \emptyset$.

A subtree of a tree $\mathcal{T}$ is a tree rooted at some node $t \in \mathcal{T}$. We distinguish two special classes of subtrees. If the terminating nodes of a subtree rooted at node t are a subset of the terminating nodes of $\mathcal{T}$ it is called a *branch* of $\mathcal{T}$ and denoted as $\mathcal{T}_t$ with $\tilde{\mathcal{T}}_t \subseteq \tilde{\mathcal{T}}$. Note that if $t = t_0$ then $\mathcal{T}_t = \mathcal{T}_{t_0} = \mathcal{T}$. On the other hand, if the root node t of the subtree coincides with the root node of a tree $\mathcal{T}$ it is called a *pruned subtree* of $\mathcal{T}$ and denoted as $\mathcal{T}^p$. Accordingly, a pruned subtree of a branch $\mathcal{T}_t$ is denoted by $\mathcal{T}_t^p$. A pruned subtree $\mathcal{T}^p$ is obtained by the pruning operation $\mathcal{T}^p \preceq \mathcal{T}$ and pruning is performed by recursively removing terminating nodes.

The mapping of the trees on the blocks in our video frame is as follows. The full tree $\mathcal{T} = \mathcal{T}_{t_0}$ is uniquely associated with a macroblock denoted as $\mathcal{I}_{i_0}$. A set $\mathcal{I}_i$ refers to samples in the image and their structured relation, i.e., their position in the image. An example illustrating the mapping of the tree structure on the macroblock structure is shown in Fig. 3.

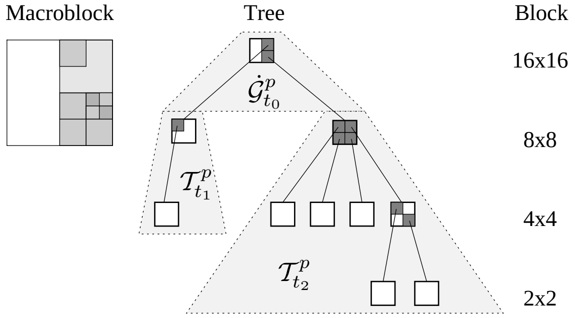

Figure 3: An example macroblock partition with its associated pruned subtree that may result by our encoding algorithm. In the macroblock, darker areas indicate smaller blocks while in the tree shaded areas are quadrants which receive further information given by subtrees.

Whether the macroblock's quadrants receive further information is specified by the shape codes. We relate the shape code associated with block $\mathcal{I}_i$ to a subtree of height 1 (a root plus its children) rooted at t and write it as $\mathcal{G}_t$. The various shapes 0 to 15 correspond to all pruned versions $\mathcal{G}_t^p \preceq \mathcal{G}_t$. Hence, the pruned subtree $\mathcal{G}_t^p = \text{root}(\mathcal{G}_t)$ is associated with shape 0, whereas the full tree $\mathcal{G}_t$ relates to shape 15. By separating $\mathcal{G}_t$ into the interior of the tree $\dot{\mathcal{G}}_t$ and the set of terminating nodes $\tilde{\mathcal{G}}_t$, a branch $\mathcal{T}_t$ can be decomposed as follows

$$\mathcal{T}_t = \dot{\mathcal{G}}_t \bigcup_{s \in \tilde{\mathcal{G}}_t} \mathcal{T}_s \qquad (1)$$

This separation corresponds to separating the block $\mathcal{I}_i$ into the segment $\dot{\mathcal{I}}_i$ (white shape area in Fig. 2) and the set of quadrants (shaded shape areas in Fig. 2) and is illustrated on the example of a pruned subtree in Fig. 3.

The optimum pruned subtree of a branch $\mathcal{T}_t$ including the case $t = t_0$ is the one that minimizes a tree functional $J(\mathcal{T}_t^p)$ over the set of all subtrees that can be pruned off $\mathcal{T}_t$

$$J(\hat{\mathcal{T}}_t^p) \;=\; \min_{\mathcal{T}_t^p : \mathcal{T}_t^p \preceq \mathcal{T}_t} J(\mathcal{T}_t^p). \qquad (2)$$

With (1) we can recast (2) as

$$\begin{aligned} \min_{\mathcal{T}_t^p} J(\mathcal{T}_t^p) &= \min_{\mathcal{G}_t^p, \mathcal{T}_s^p} J(\dot{\mathcal{G}}_t^p \bigcup_{s \in \tilde{\mathcal{G}}_t^p} \mathcal{T}_s^p) \\ &= \min_{\mathcal{G}_t^p} \left[J(\dot{\mathcal{G}}_t^p) + \sum_{s \in \tilde{\mathcal{G}}_t^p} \min_{\mathcal{T}_s^p} J(\mathcal{T}_s^p) \right] (3) \end{aligned}$$

Realizing that for the smallest quadrants we only permit shape 0, the related admissible set of the pruned subtrees $\mathcal{G}_t^p$ reduces to the case $\mathcal{G}_t^p = \text{root}(\mathcal{G}_t)$ and therefore, the second term in (3) vanishes for these. Hence, the optimization can be conducted recursively using (3).

In the video coder, the function $J(\dot{\mathcal{G}}_t^p)$ in (3) is in fact a Lagrangian cost functional. To each interior of the tree $\dot{\mathcal{G}}_t^p$ we associate parameters, i.e., the displacement vectors and intensity updates writing them $\boldsymbol{p}_t = (d_x, d_y, q)$. Thus, the tree functional reads

$$J(\dot{\mathcal{G}}_t^p, \boldsymbol{p}_t) = D(\dot{\mathcal{G}}_t^p, \boldsymbol{p}_t) + \lambda R(\dot{\mathcal{G}}_t^p, \boldsymbol{p}_t), \qquad (4)$$

where a distortion term $D(\cdot)$ is weighted versus a rate term $R(\cdot)$ by a Lagrange multiplier λ. Remembering that we relate the image segment $\dot{\mathcal{I}}_i$ to $\dot{\mathcal{G}}_t^p$ the distortion $D(\cdot)$ is given by

$$D(\dot{\mathcal{G}}_t^p, \boldsymbol{p}_t) = \sum_{x,y \in \dot{\mathcal{I}}_i} ||I_k(x,y) - I'_{k-1}(x + d_x, y + d_y) - q||^2 \qquad (5)$$

where $I_k(\cdot)$ is the intensity value of a pixel in the current original frame and $I'_{k-1}(\cdot)$ is the intensity of a decoded pixel in the preceding frame. The rate term $R(\cdot)$ in Eq. (4) is computed by simple VLC table look-up and given by the sum of the rates for the shape, the displacement vector, and the intensity update.

4 CODEBOOK DESIGN

We view the MIC as entropy-constrained mean-shape vector quantization (ECMSVQ), wherein the image blocks to be encoded are quantized using shape codebooks that consist of image blocks of the same size in the previously decoded frame. A particular codebook entry is addressed by the displacement vector. Hence, each image block to be encoded uses its own shape codebook, which is dependent on the position of the block in space and time. Consequently, the number of displacement vectors that is determined by the accuracy of the MIC and the motion search range, relates to the shape codebook size in VQ.

The quadtree-structured video coder can be identified as a tree-structured ECMSVQ with three types of variable length code (VLC) tables at each block size for: 1) the tree structures, 2) the displacement vectors, and 3) the intensity updates. There is no intensity update for the largest size blocks for Y, Cb, and Cr, no motion vector for the largest size blocks for Cb and Cr, and no tree structure information for all smallest size blocks. Hence, the training is performed on 10 VLC tables for Y and 6 tables for Cb as well as Cr in the QCIF case. In order to design the quadtree codec, we have utilized an iterative algorithm that is similar to the well known algorithm for the design of unstructured ECVQs [8].

Our design algorithm can be summarized as follows:

Quadtree codec design algorithm

Step 1: Given a set of training sequences, and the quadtree codec as described above. Initialize the VLC tables of the quadtree codec with fixed length codes (assuming uniform distributions).

Step 2: Encode all training sequences using the quadtree codec and the rate-constrained bit allocation described in section 3.

Step 3: Update VLC tables for the entire coder, i.e., the codes for shapes, displacement vectors, and intensity updates separately for each block size, using the probabilities produced in **Step 2**.

Step 4: Compute Lagrangian cost function for all training sequences. Convergence achieved? Yes: stop, No: go to **Step 2**.

Let us attempt to discuss particular parameter settings in our design algorithm in the next section within the context of the simulations we have done.

5 SIMULATION RESULTS

For our simulation results, we are using 18 training sequences each covering 10 seconds of video. The encoding frame rate is 7.5 *Hz.* The sequences are: *Akiyo, Bream, Coastguard, Children, Container, Foreman, Fun Fair, Hall Monitor, Mobile & Calendar, Mother & Daughter, News, Sean, Silent Voice, Stefan, Table Tennis, Total Destruction, Tunnel,* and *Weather.* For test purpose we selected sequences which are *not* inside the training set. The results presented in this paper are obtained on the sequence *Car Phone* at frame rate 7.5 *Hz.* We obtained consistent results for other test sequences as well.

In Fig. 4, the convergence of the design algorithm when training on the set of QCIF sequences is shown using the test sequence. The training was done for only one value of the Lagrange parameter $\lambda = 150$, that was chosen so as to obtain very low bit-rates. The plots illustrate the results produced when running the quadtree coder for the test sequence in QCIF resolution using various values of the Lagrange parameter $100 \leq \lambda \leq 500$. Training is stopped after the seventh iteration resulting in our optimal quadtree video coding scheme.

In Fig. 6, the performance of the optimal quadtree codec is compared to the TMN 1.6 codec, the test model for H.263 standard. The TMN 1.6 codec is available as public domain software by anonymous ftp to `bonde.nta.no`. The TMN 1.6 coder is run with constant quantizer and all options turned on except the PB-frames and Syntax-Based Arithmetic coding modes to obtain one point in the rate-distortion plot. In order to generate the curve we are varying the quantization parameter for the TMN 1.6 codec. At low bit-rates, improvements are obtained with the quadtree codec against TMN 1.6 of about 0.5 dB PSNR.

In Fig 7, we show the sensitivity of the design algorithm against variations of the Lagrange parameter in the design process in order to demonstrate the impact of the rate-constrained to the VLC training imposed by the Lagrange parameter. For these simulation results we have trained the quadtree codec using $\lambda = 0, 150, 400$. The results are generated again by varying the Lagrange parameter for the test sequence.

When coding CIF sequences, the number of pixels relating to each macroblock is scaled by 4 to 32×32 pixels such that each frame is again separated into 99 macroblocks as in the QCIF case. Hence, the resulting smallest block size is 4×4 samples. In Fig. 8, results are shown, when coding the test sequence in CIF resolution using the VLCs obtained by training with QCIF as well as CIF sequences. For the training with CIF sequences, we are using the CIF versions of the 18 training sequences that we used for the QCIF results. The settings for the training with the CIF sequences are similar to the QCIF case. The curves in Fig. 8 suggest that it does not matter at which resolution the training is performed. The comparison with the TMN 1.6 codec in case of CIF resolution is presented in Fig. 9 demonstrating improvements up to 1.8 dB PSNR at the low bit-rate end. All results presented are generated from decoded bit streams. Several decoded sequences will be shown in a video presentation for subjective evaluation.

6 CONCLUSIONS

The design of a quadtree video codec is presented. The impact of the rate-distortion optimized bit allocation on the variable length code design is demonstrated. At very low bit-rates the quadtree codec shows marginal improvements for QCIF but significant gains for sequences in CIF resolution when comparing to H.263 TMN 1.6 codec.

References

[1] ITU-T Recommendation H.263, "Video Coding for Low Bitrate Communication", Draft, Dec. 1995.

[2] T. Wiegand, M. Lightstone, D. Mukherjee, T. G. Campbell, and S. K. Mitra, "Rate-Distortion Optimized Mode Selection for Very Low Bit Rate Video Coding and the Emerging H.263 Standard", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 2, pp. 182–190, Apr. 1996.

[3] G. M. Schuster and A. K. Katsaggelos, "Fast and Efficient Mode and Quantizer Selection in the Rate Distortion Sense for H.263", in *Proceedings of the SPIE Conference on Visual Communications and Image Processing*, Orlando, USA, Mar. 1996, pp. 784–795.

[4] ISO/IEC JTC1/SC29/WG11 MPEG96/M0553, "Iterated Systems MPEG-4 Video Submission Technical Description", Mar. 1996.

[5] G. J. Sullivan and R. L. Baker, "Efficient Quadtree Coding of Images and Video", *IEEE Transactions on Image Processing*, vol. 3, no. 3, pp. 327–331, May 1994.

[6] P. Strobach, "Tree-Structured Scene Adaptive Coder", *IEEE Transactions on Communications*, vol. 38, no. 4, pp. 477–486, Apr. 1990.

[7] P. A. Chou, T. Lookabaugh, and R. M. Gray, "Optimal Pruning with Applications to Tree-Structured Source Coding and Modeling", *IEEE Transactions on Information Theory*, vol. 35, no. 2, pp. 299–315, Mar. 1989.

[8] P. A. Chou, T. Lookabaugh, and R. M. Gray, "Entropy-Constrained Vector Quantization", *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 37, no. 1, pp. 31–42, Jan. 1989.

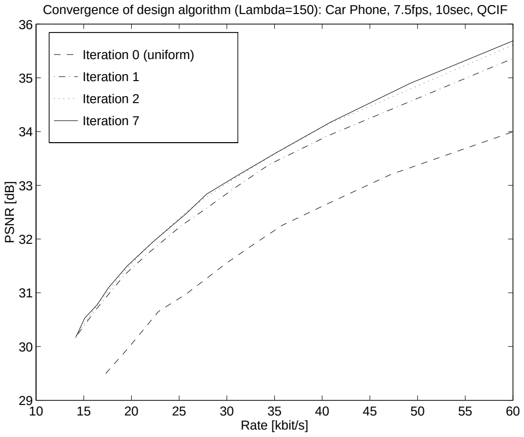

Figure 4: Convergence of design algorithm measured in terms of PSNR vs. rate for *Car Phone* coded in QCIF resolution.

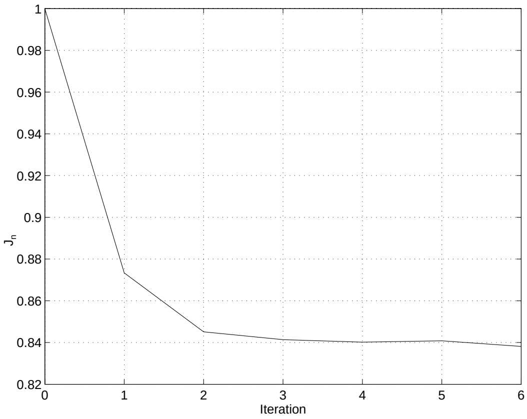

Figure 5: Convergence of design algorithm measured in the decrease of overall costs for the complete training set.

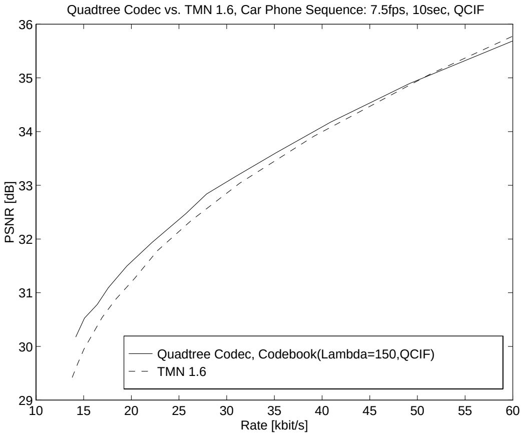

Figure 6: Comparison of optimal quadtree codec to H.263 TMN 1.6 codec when coding *Car Phone* in QCIF resolution.

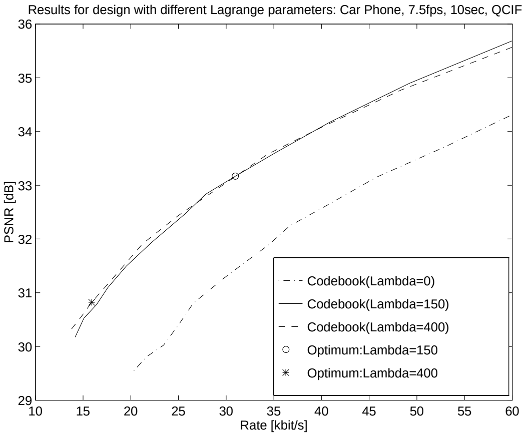

Figure 7: Sensitivity of resulting rate-distortion performance when training with different Lagrange parameters.

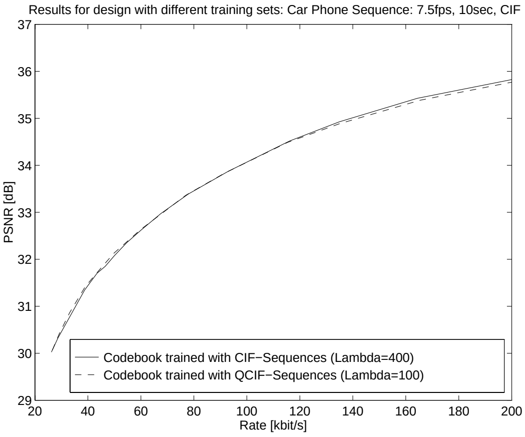

Figure 8: Sensitivity of the codebook against variation of the resolution from QCIF to CIF for the quadtree codec.

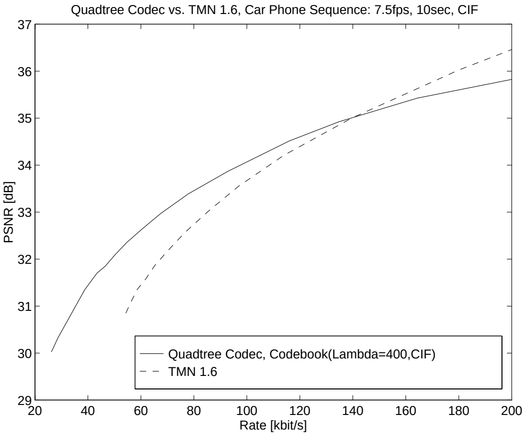

Figure 9: Comparison of optimal quadtree codec to H.263 TMN 1.6 codec when coding *Car Phone* in CIF resolution.