OPTIMIZATION OF TRANSFORM COEFFICIENT SELECTION AND MOTION VECTOR ESTIMATION CONSIDERING INTER-PICTURE DEPENDENCIES IN HYBRID VIDEO CODING

Brad Schumitsch[*]

Stanford University, CA, USA
Electrical Engineering Department
schumitsch@stanford.edu

Heiko Schwarz, and Thomas Wiegand

Fraunhofer Institute for Telecommunications
Heinrich-Hertz-Institut, Berlin, Germany
{hschwarz,wiegand}@hhi.de

ABSTRACT

Considering inter-picture dependencies when selecting transform coefficient levels in hybrid video coding can be done via formulating the decoding process as a linear signal model and solving a quadratic program. The basic method assumes motion estimation and quantization parameters as being given and then selects the transform coefficient levels. However, when motion vectors are determined in advance, motion estimation must be conducted using uncoded reference pictures which is known to deliver inferior results compared to motion estimation on decoded reference pictures. In this work, we expand the basic method to incorporate the case where the motion estimation is considering decoded reference pictures. We propose an approach that iterates between transform coefficient selection and motion estimation. We find that a simple two-pass iteration works reasonably well. Our simulation results using an H.264/AVC-conforming encoder show coding gains up to 1 dB in comparison to the quantization method specified in the test model of H.264/AVC.

1. INTRODUCTION

Hybrid codecs, such as H.264/AVC [1, 2] are the most successful class of video compression designs. These codecs produce a motion-compensated prediction as well as a prediction residual. A hybrid video encoder makes many decisions in its attempt to achieve the best possible tradeoff between bit-rate and distortion given constraints on delay and complexity. Because of the use of motion-compensated prediction, many inter-picture dependencies exist among these decisions that are typically relevant for many pictures of a coded video sequence.

There has been a large amount of work on optimization problems in hybrid video coding by other researchers in the past. One particular focus has been on Lagrangian optimization methods [3, 4, 5, 6] which we also utilize in our work. Work on considering the dependencies between the various encoding decisions has focused on modelling these dependencies by trellises which allows the use of dynamic programming methods. Bit-allocation to DCT coefficients was proposed by Ortega and Ramchandran for MPEG-2 Video [7], and a version that handles the more complex structure of the entropy coding of H.263 has been developed in [8]. The selection of other coding parameters such as motion vectors and macroblock modes has been optimized in [9, 10, 11, 12, 13].

In [14], we have presented a method that optimizes the selection of transform coefficient levels considering their impact across multiple pictures given the motion vector field and macroblock mode selection. For that, the bit allocation problem to transform coefficients in hybrid video coding was formulated as a quadratic program allowing the consideration of the impact of the selection of a particular transform coefficient when being referenced in motion-compensating other samples. Since the motion vectors are an input to the optimization problem, this formulation requires constant motion vectors. In this work, we have extended the method in [14] towards motion estimation on decoded reference pictures. We show that the motion vectors need not be completely determined for the entire se-

[*]Funded by a Fulbright Fellowship.

quence to utilize the optimization problem approach. Applying an iterative method allows us to use the linear signal model to select transform coefficients while using motion vectors based on the reconstructed sample values.

This paper is organized as follows. In the next section, we define our problem. For completeness, in section 3 we include how to optimize transform coefficient levels taking into account inter-picture dependencies by using a linear signal model and reformulating the problem as a quadratic program. Section 4 shows by repeatedly applying the method for selecting transform coefficient levels, better motion estimation is possible than if it is only run once each picture. Section 5 gives results.

2. PROBLEM STATEMENT AND BACKGROUND

Consider the encoding process of a hybrid video encoder such as H.264/AVC [1, 2]. At this stage, let's assume that the motion vectors and quantization parameters are already determined for the entire sequence, although this assumption will be relaxed in section 4. The first picture is coded as an intra picture and all other parts of the remaining pictures are coded using motion-compensated prediction. The task that remains for the encoder is to determine the transform coefficient levels that represent the residual signal in order to optimize some cost function of image fidelity and bit-rate. A common technique is to use a Lagrangian formulation and to minimize a linear combination of distortion and bit-rate, $D + \lambda R$ [4, 5]. The most common distortion measure is the mean squared error, which we also use. The bit-rate is typically a rather complicated function $R(\mathbf{c})$ of the quantized transform coefficient levels $\mathbf{c}$.

Let's assume the encoding of a video sequence with K pictures of width W and height H samples with a dynamic range of $\boldsymbol{\mathcal{A}} = \{0..255\}$ and let $N = K \times W \times H$ be the number of samples. The vector $\boldsymbol{v} \in \boldsymbol{\mathcal{A}}^N$ represents the original N sample values with v_i being the i'th sample value. Let $\boldsymbol{s}(\boldsymbol{c}) \in \boldsymbol{\mathcal{A}}^N$ represent the reconstructed sample values after decoding with s_i being the i'th decoded sample value corresponding to v_i. Hence the problem of selecting transform coefficient levels $\boldsymbol{c}$ can be written as:

$$\text{minimize}\{(\boldsymbol{v} - \boldsymbol{s}(\boldsymbol{c}))^T(\boldsymbol{v} - \boldsymbol{s}(\boldsymbol{c})) + \lambda R(\boldsymbol{s}, \boldsymbol{c})\} \quad (1)$$

Let $\boldsymbol{s_B}$ be the decoded samples for the current block $\boldsymbol{B}$ and $\boldsymbol{c_B}$ be the transform coefficient levels for the block $\boldsymbol{B}$. Some previous publications on transform coefficient optimization [7, 8] only considered the choice of $\boldsymbol{c_B}$ to have an impact on $\boldsymbol{s_B}$ with regards to distortion or bit-rate. The impact on other blocks was ignored.

Moreover, many encoding algorithms (cp. [15]) determine the transform coefficient levels c_i of a block ignoring the dependency of R on $\boldsymbol{c_B}$ by simple quantization of the associated transform coefficient levels t_i according to

$$c_i = \text{sgn}(t_i) * \lfloor (|t_i| + f * q)/q \rfloor \quad (2)$$

with q being the quantization step size and f being the dead-zone control parameter. But this way of obtaining the levels is optimal only with respect to mean square error distortion measured between the original and reconstructed samples for the current block. The impact of the introduced quantization error on samples referring to this block by motion compensation is not considered.

3. TRANSFORM COEFFICIENT LEVEL SELECTION

In this section, we briefly explain a method of selecting transform coefficient levels considering inter-picture dependencies that are introduced by motion compensation in hybrid video coding. As mentioned above, in the next section, we will show how this method can be extended towards motion estimation on decoded reference pictures.

This description is done in 3 parts. First, we express the relationship between the transform coefficient levels and the decoded samples via a linear signal model for the decoder. Then, we set up the optimization problem in (1) as a quadratic program. Third, we explain a heuristic that obtains integer-valued transform coefficient levels.

3.1. Linear Signal Model for Hybrid Video Decoding

We assume that a decoded sample s_i can be represented as a linear combination of previously decoded samples, the corresponding residual sample, and a static predictor. Hence, a linear model for $\boldsymbol{s}$ as a signal equation can be written as follows

$$\boldsymbol{s} = \hat{\boldsymbol{s}} + \boldsymbol{u} + \boldsymbol{p} = \boldsymbol{M}\boldsymbol{s} + \boldsymbol{T}\boldsymbol{c} + \boldsymbol{p} \tag{3}$$

The $N \times 1$ vector $\boldsymbol{s}$ is a column vector containing all decoded samples of the pictures that will be jointly optimized. The $N \times N$ matrix $\boldsymbol{M}$ expresses the motion compensation, i.e., mapping the decoded sample s_j onto the decoded sample s_i. Note that for the linear signal model $\boldsymbol{M}$ is a constant. The rows of the $N \times N$ matrix $\boldsymbol{T}$ provide inverse scaling and transform of the transform coefficient levels $\boldsymbol{c}$ in order to obtain the decoded residual signal $\boldsymbol{u}$. The column vector $\boldsymbol{p}$ is a static predictor which represents motion-compensated prediction samples referencing decoded samples that are not part of the vector $\boldsymbol{s}$.The construction of $\boldsymbol{M}, \boldsymbol{T}, \boldsymbol{c}$, and $\boldsymbol{p}$ is described in detail below.

The matrix $\boldsymbol{M}$ is constructed using the motion vectors and reference picture indices that are assumed to be fixed while optimizing transform coefficient levels. The values in the row $\boldsymbol{m}_i$ of $\boldsymbol{M}$ express how each decoded sample in $\boldsymbol{s}$ contributes to the motion-compensated prediction sample for s_i. For example, let's assume the prediction sample s_i is motion-compensated with integer-sample accuracy referencing the sample s_z. Then $m_{iz} = 1$ and $m_{ij} = 0 \; \forall \; j \neq z$. In a more complicated example, assume that the prediction sample for s_i is the result of 1/2 pixel motion estimation, where the H.264/AVC 6-tap filter must be applied. In this case, there are 6 non-zero entries in $\boldsymbol{m}_i$, [1 -5 20 20 -5 1]/32. The indices of these non-zero entries depend on the motion vector, the reference picture index for motion compensation, and the position of the current sample. Note that $\boldsymbol{m}_i$ could have 36 non-zero entries if the 1/2 pixel filter must be applied twice to construct the prediction sample. If B-pictures are used, $\boldsymbol{m}_i$ could have up to 72 non-zero entries. Note also that we are ignoring any rounding in the description of fractional-sample interpolation.

The matrix $\boldsymbol{T}$ is constructed using the 4x4 inverse transform and the inverse scaling equations of H.264/AVC. Let $\boldsymbol{s_B}$ be a 4x4 block of decoded samples. Ignoring rounding, the residual samples for $\boldsymbol{u_B}$ that are used to obtain $\boldsymbol{s_B}$ are given by a linear combination of 16 transform coefficient levels in $\boldsymbol{c_B}$. The weights in this linear combination are determined by the inverse transform used, the position of the residual sample within $\boldsymbol{B}$, the position of the transform coefficient level within $\boldsymbol{B}$, and quantization parameter for $\boldsymbol{B}$. Note that the non-zero entries of $\boldsymbol{T}$ are located according to the ordering of the c_i relative to the positions in the pictures.

The vector $\boldsymbol{p}$ contains the motion-compensated prediction signal for samples whose prediction depends on samples outside of the K pictures currently being optimized. For example, the intra picture is currently not optimized by our algorithm and its samples are therefore outside the vector $\boldsymbol{s}$. The contribution of these intra samples to the values of all samples in $\boldsymbol{s}$ is expressed after motion-compensating them towards each $\boldsymbol{s}$.

3.2. Quadratic Program Formulation

Given the signal model for hybrid video decoding in (3), the minimization problem in (1) for transform coefficient level selection can be written as

$$\begin{aligned} &\text{minimize} && \{(\boldsymbol{v}-\boldsymbol{s})^T(\boldsymbol{v}-\boldsymbol{s}) + \lambda R(\boldsymbol{v}, \boldsymbol{c})\} && (4) \\ &\text{subject to} && \boldsymbol{s} = \boldsymbol{M}\boldsymbol{s} + \boldsymbol{T}\boldsymbol{c} + \boldsymbol{p} && (5) \end{aligned}$$

Except for the functional relationship between R and $\boldsymbol{c}$, this is very close to a quadratic program. A quadratic program is a problem of the form

$$\begin{aligned} &\text{minimize} && \{\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{f}^T\boldsymbol{x}\} \\ &\text{subject to} && \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & && \boldsymbol{E}\boldsymbol{x} \leq \boldsymbol{g} \end{aligned} \tag{6}$$

where $\boldsymbol{x}$ is a column vector of real variables and $\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x}$ is a convex function in $\boldsymbol{x}$. The advantage of having our problem in the form of a quadratic program is that efficient algorithms exist to find the optimal $\boldsymbol{x}$.

Note that the actual bit-rate is a very complex function of $\boldsymbol{c}$. However, transform coefficient levels with a smaller absolute value almost always result in a smaller rate. Therefore, in order to obtain a piece-wise linear approximation of $R(\boldsymbol{c})$, the following rate model is used

$$R(\mathbf{c}) \approx \sum_i \max(0, |c_i| - \hat{w}) \tag{7}$$

The reason for the introduction of the integer scaler $\hat{w} \in \boldsymbol{\mathcal{A}}$ will be explained in the next section and $\hat{w}$ can be assumed for now to be equal to 0. We make our problem a quadratic program by allowing $\boldsymbol{s}$, $\boldsymbol{c} \in \mathbb{R}^n$ and introducing another variable $\boldsymbol{r} \in \mathbb{R}^n$ such that

$r_i > |c_i| \;\forall\; i$. Our problem is now:

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{s}^T\boldsymbol{s} - 2\boldsymbol{s}^T\boldsymbol{v} + \lambda\mathbf{1}^T\boldsymbol{r} \\ \text{subject to} \quad & \boldsymbol{s} = \boldsymbol{M}\boldsymbol{s} + \boldsymbol{T}\boldsymbol{c} + \boldsymbol{p} \\ & \boldsymbol{r} \geq \boldsymbol{c} - \hat{w}\mathbf{1} \\ & \boldsymbol{r} \geq -\boldsymbol{c} - \hat{w}\mathbf{1} \\ & \boldsymbol{r} \geq \mathbf{0} \end{aligned} \tag{8}$$

where $\mathbf{1}$ is a vector with every entry equal to one, $\mathbf{0}$, is a vector with every entry equal to zero, and our variables are $\boldsymbol{s}$, $\boldsymbol{c}$, and $\boldsymbol{r}$. It can be shown that the above formulation can be mapped into the quadratic program in (6). A mapping is given in the appendix.

3.3. Determination of Integer-Valued Transform Coefficient Levels

A quadratic program solver solver, such as MOSEK[16], can solve (8) and return the optimal real-valued values for the unknown variables. However, the transform coefficient levels must be integer valued. The simplest heuristic would be to round each non-integer c_i to the nearest integer. Although this yields a reasonable result for $\boldsymbol{c}$, the reconstructed samples based on integer valued $\boldsymbol{c}$ are different than those from the real valued $\boldsymbol{c}$. Thus the resulting PSNR is very likely to be lower than the above algorithm calculated it would be. This is somewhat ameliorated by rounding only a subset of $\boldsymbol{c}$, adding the effect of the determined elements of $\boldsymbol{c}$ into the static predictor $\boldsymbol{p}$, removing the unneeded columns of $\boldsymbol{T}$, and then re-solving (8) with the now smaller problem. This gives the quadratic program solver a chance to take the rounding into account for the remaining unknowns of $\boldsymbol{c}$, which is especially helpful for the unknowns of $\boldsymbol{c}$ in the same 4x4 blocks as the determined and rounded elements of $\boldsymbol{c}$. We repeat this process until all of $\boldsymbol{c}$ has been rounded.

The iterative algorithm is given as follows. Note that $\boldsymbol{t_i}$ is the ith column of $\boldsymbol{T}$.

```
0 Initialize $w = \delta$
1 Set $\hat{w} = \lfloor w \rfloor$
2 Solve (8) obtaining non-integer valued elements
  in $\boldsymbol{c}$
3 For all $c \leq w$, $\hat{c_i} = \lfloor c_i + 0.5 \rfloor$, remove the row
  $\boldsymbol{t}_i$ from $\boldsymbol{T}$, and update $\boldsymbol{p} = \boldsymbol{p} + \hat{c}_i\boldsymbol{t}_i$
4 For all $c > w$, assign them to $\boldsymbol{c}$
5 Set $w = w + \Delta$
6 Set $\lambda = \lambda_i$ in optimization (8)
7 If $\boldsymbol{c}$ is not empty, go to step 1, otherwise stop
```

When the algorithm is finished $\boldsymbol{c}$ is empty, and the solution to the problem in (8) is in $\hat{\boldsymbol{c}}$ with elements $\hat{c}_i$

Conceptually, we are solving for the transform coefficient levels whose value are equal to $\hat{w} = \lfloor w \rfloor$ each time. Our rate function acts as a penalty function on $\sum |c_i|$. The $\hat{w}$ in the rate function adds a "free" zone for c_is less than $\hat{w}$. This type of penalty function in a quadratic program tends to have solutions with a relatively large number of c_is at $\hat{w}$ as desired [17].

By changing λ with each value of $\hat{w}$ we are able to effectively represent rate models other than the linear one shown in (7). For example, by decreasing λ in a logarithmic manner, we can better approximate a logarithmic rate function. Empirically, we have found that letting $\lambda_0 = 4\lambda_1$ and $\lambda_1 = \lambda_2 = \lambda_3$ works reasonably well. We have chosen these values in our experiments in section 5.

The choice of δ and Δ is a trade-off between computation time and coding efficiency. Empirically, we have found that the values of $\delta = 0.5$ and $\Delta = 1$ offer a reasonable trade-off. We have used these values in our experiments in section 5.

4. MOTION VECTOR UPDATE

4.1. Problem

The above approach enables us to select transform coefficient levels given the quantization parameters and motion vector field. However, it is known that selecting motion vectors based on decoded reference pictures gives superior performance compared to selecting motion vectors on uncoded reference pictures. Thus we would like to be able to change motion vectors as we encode a picture. A challenge is that the motion vectors must be held fixed for the *future* pictures while the transform coefficient levels are selected for the current picture using the above algorithm. Thus, we have a chicken-and-egg problem. We would like to use motion vectors for future pictures that are derived from the decoded picture for the current picture, but this decoded picture depends on transform coefficients that we would like to calculate using inter-picture optimization which needs motion vectors for future pictures.

4.2. Solution

Our proposed solution to the chicken-and-egg problem is to use an iterative approach. The basic idea is to use a guess for the transform coefficient levels and use this guess to compute motion vectors for future pictures. We then refine our choice of transform coefficients for the current picture using inter-picture optimization. Next, we recalculate the motion vectors based on the new transform coefficients.

In our following description we let K be the number of pictures we wish to jointly optimize over, and k be a parameter that can be passed to the algorithm of section 3. As we will see k is not always equal to K.

We notice that for the first picture of the group of K pictures the motion-compensated prediction samples can be computed exactly, since all the previous pictures are already encoded and decoded reference pictures are available. The transform coefficient levels are then selected for this picture without looking at future pictures. That is, the algorithm in section 3 is run with $k = 1$. These levels are used to calculate the motion vectors for the 2nd picture. Then the transform coefficients are jointly selected for both of these pictures, again by running the algorithm in section 3 with $k = 2$. The motion vectors for picture 2 are recalculated based on the new transform coefficients of picture 1 and the coefficients from the 2nd picture are then used to select the motion vectors for the 3rd picture. At which point, we can call the transform selection algorithm with $k = 3$. This process is repeated until $k = K$ pictures.

Then the transform coefficient levels and motion vectors for the first picture are saved, while all other transform coefficient levels and motion vectors are discarded. We then, in a "sliding window" fashion, optimize the next group of K pictures (pictures 2 though $K+1$ in the old ordering) using the technique described in the previous paragraph.

To summarize, we propose using the following steps to encode a video sequence:

0 Encode the intra-picture use a conventional technique. Please note that our method does not optimize the Intra picture.

1 Consider the first group of K inter-pictures.

2 Set $k = 1$

3 Compute the motion vectors for the 1st picture in the group of K pictures

4 Run the inter-picture transform coefficient levels optimization with parameter k.

5 Recompute the motion vectors for the first $k + 1$ pictures in the group.

6 Set $k = k + 1$

7 If $k \leq K$, go to step (4)

8 If $k > K$, save transform coefficient levels and motion vectors for first picture. If more pictures exist, go to step (2) for next group of K pictures (picture 2 though $K + 1$ in the old ordering).

5. RESULTS

We have conducted experiments to verify our algorithm using a video codec that conforms to H.264/AVC [1]. As mentioned earlier, we fix the quantization parameter. Motion estimation is performed as described in section 4 and is conducted using the Lagrangian approach as described in [18]. The Lagrange parameter is also chosen according to [18].

The first picture is coded as an intra picture and all remaining pictures are coded as inter pictures either using P or B slices. The deblocking filter is used, and inter-picture prediction utilizes 5 reference pictures. Note that we disallow the use of intra macroblock modes within inter pictures.

Results reported in Fig. 1 and Fig. 2 are obtained from the QCIF sequences Flowergarden and Tempete. For all sequences, 50 pictures are encoded at 30 Hz.

Fig. 1 shows the results when coding the first picture as an intra picture and all other pictures with P slices. We consider 1, 2 and 3 pictures (K) jointly. The case $K = 1$ was obtained using the test model for H.264/AVC. When moving from considering $K = 1$ to $K = 2$ pictures jointly, a PSNR gain 0.4 dB can be measured at low-bit rates and 0.9 dB at high-bit rates for Flowergarden (top) and 0.4 dB for Tempete (bottom). Moving from $K = 2$ to $K = 3$ provides another 0.4 dB at low-bit rates and another 0.1 dB at high rates giving a total gain of 0.8 dB and 1.0 dB respectively for Flowergarden. For Tempete this provides another 0.3 dB, giving a total gain of 0.7 dB.

Fig. 2 shows the results when coding the first picture as an intra picture, every second pictures with P slices and the immediate pictures with B slices. For Flowergarden (top), when moving from considering $K = 1$ to $K = 3$ pictures jointly, a PSNR gain of

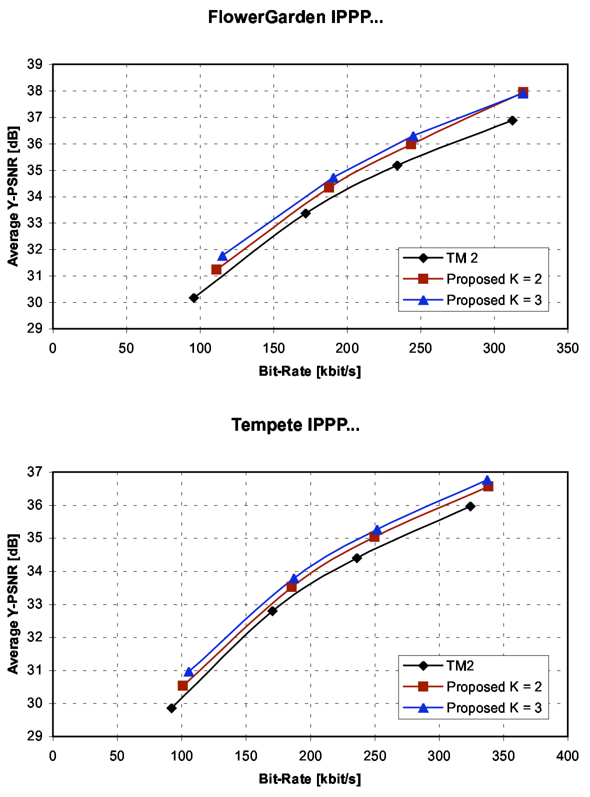

Fig. 1. PSNR vs. bit-rate for the sequences Flowergarden (top) and Tempete (bottom) when using IPPP... coding with five reference pictures.

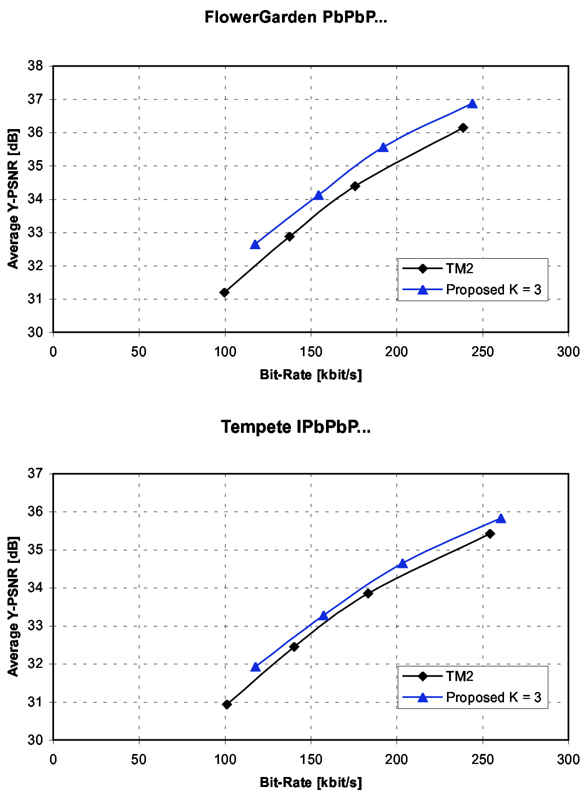

Fig. 2. PSNR vs. bit-rate for the sequences Flowergarden (top) and Tempete (bottom) when using IBPBPBP... coding with five reference pictures.

about 0.7 dB is observed. For Tempete (bottom), a gain of 0.3 dB is seen.

6. CONCLUSIONS AND FUTURE WORK

We have expanded a strategy of selecting transform coefficient levels considering the inter-picture dependencies produced by motion compensation in hybrid video coding to work with motion vectors on reconstructed pictures. The algorithm consists of iterating between estimating transform coefficients and motion vectors. The simulation results using the video coding standard H.264/AVC show coding gains of up to 1.0 dB in comparison to the quantization strategy specified in the test model of H.264/AVC.

Opportunities exist to further expand this method. One direction is to include the motion vectors variations within one pass of the optimization. Another is to include quantization parameter changes.

Although the current work does not include intra-pictures and intra-blocks, there may be a fairly straightforward extension to these cases.

After a group of K pictures are considered and encoded using the ideas in the paper, instead of retaining the transform coefficient levels and motion vectors for the entire 1st picture of this group, one could only retain the transform coefficient levels from samples that did do not depend on other samples in the group of K pictures for their motion compensated prediction. However, this would increase encoding time.

Reducing coding time is also future work. The current algorithm is not ideal for real-time encoding or low-delay encoding. Future work will investigate both ways to reduce encoding time while maintaining the majority of the gains, as well as the tradeoffs in general between the computational complexity and coding efficiency gains for various features of the algorithm.

7. REFERENCES

[1] ITU-T Recommendation H.264 & ISO/IEC 14496-10 AVC. Advanced Video Coding for Generic Audiovisual Services. 2003.

[2] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the H.264/AVC Video Coding Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):688–703, July 2003.

[3] H. Everett III. Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources. *Operations Research*, 11:399–417, 1963.

[4] Y. Shoham and A. Gersho. Efficient Bit Allocation for an Arbitrary Set of Quantizers. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 36:1445–1453, September 1988.

[5] P. A. Chou, T. Lookabaugh, and R. M. Gray. Entropy-Constrained Vector Quantization. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(1):31–42, January 1989.

[6] G. J. Sullivan and R. L. Baker. Rate-Distortion Optimized Motion Compensation for Video Compression Using Fixed or Variable Size Blocks. In *Proc. GLOBECOM'91*, pages 85–90, Phoenix, AZ, USA, December 1991.

[7] K. Ramchandran, A. Ortega, and M. Vetterli. Bit Allocation for Dependent Quantization with Applications to Multiresolution and MPEG Video Coders. *IEEE Transactions on Image Processing*, 3(5):533–545, September 1994.

[8] J. Wen, M. Luttrell, and J. Villasenor. Trellis-Based R-D Optimal Quantization in H.263+. *IEEE Transactions on Circuits and Systems for Video Technology*, 1998. Submitted for publication.

[9] A. Ortega, K. Ramchandran, and M. Vetterli. Optimal Trellis-Based Buffered Compression and Fast Approximations. *IEEE Transactions on Image Processing*, 3(1):26–40, January 1994.

[10] 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*, 6(2):182–190, April 1996.

[11] J. Lee and B. W. Dickinson. Joint Optimization of Frame Type Selection and Bit Allocation for MPEG Video Coders. In *Proceedings of the IEEE International Conference on Image Processing*, volume 2, pages 962–966, Austin, TX, USA, November 1994.

[12] M. C. Chen and A. N. Willson. Rate-Distortion Optimal Motion Estimation Algorithm for Video Coding. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 4, pages 2096–2099, Atlanta, GA, USA, May 1996.

[13] G. M. Schuster and A. K. Katsaggelos. A Video Compression Scheme with Optimal Bit Allocation Among Segmentation, Motion, and Residual Error. *IEEE Transactions on Image Processing*, 6(11):1487–1502, November 1997.

[14] Heiko Schwarz Brad Schumitsch and Thomas Wiegand. Inter-Frame Optimization of Transform Coefficient Selection in Hyprid Video Coding. In *Proceedings of the Picture Coding Symposium*, San Francisco, CA, December 2004.

[15] H. S. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky. Low-Complexity Transform and Quantization in H.264/AVC. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):598–603, July 2003.

[16] MOSEK ApS. *The MOSEK optimization tools version 3.1 (Revision 28) User's manual and reference*. MOSEK ApS, Copenhagen, Denmark, 2002.

[17] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, United Kingdom, 2004.

[18] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. J. Sullivan. Rate-Constrained Coder Control and Comparison of Video Coding Standards. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):688–703, July 2003.

Appendix: Quadratic Program Mapping

For completeness, we include how the the optimzation problem in (8) can be mapped into the standard form of a quadratic program (6). This mapping is done by the following equations.

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{s} \\ \boldsymbol{c} \\ \boldsymbol{r} \end{bmatrix} \tag{9}$$

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{10}$$

$$\boldsymbol{f} = \begin{bmatrix} -2\boldsymbol{v} \\ \mathbf{0}_N \\ \lambda \mathbf{1}_N \end{bmatrix} \tag{11}$$

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{I} - \boldsymbol{M} & -\boldsymbol{T} & \mathbf{0} \end{bmatrix} \tag{12}$$

$$\boldsymbol{b} = \boldsymbol{p} \tag{13}$$

$$\boldsymbol{g} = \begin{bmatrix} -\hat{w}\mathbf{1}_N \\ -\hat{w}\mathbf{1}_N \\ \mathbf{0}_N \end{bmatrix} \tag{14}$$

$$\boldsymbol{E} = \begin{bmatrix} \mathbf{0} & \boldsymbol{I} & -\boldsymbol{I} \\ \mathbf{0} & -\boldsymbol{I} & -\boldsymbol{I} \\ \mathbf{0} & \mathbf{0} & -\boldsymbol{I} \end{bmatrix} \tag{15}$$

where $\mathbf{0}$ is the $N \times N$ all zero matrix, $\boldsymbol{I}$ is the $N \times N$ identity matrix, $\mathbf{0}_N$ is a $N \times 1$ column vector of all zeros, and $\mathbf{1}_N$ is a $N \times 1$ column vector of all ones.