OPTIMAL PER-PIXEL ESTIMATION FOR SCALABLE VIDEO CODING

Athanasios Leontaris and Pamela C. Cosman

Department of Electrical and Computer Engineering
University of California, San Diego, La Jolla, CA 92093-0407
{aleontar,pcosman}@code.ucsd.edu

ABSTRACT

We address the problem of real-time mode decisions for enhancement layer coding in the context of fine granularity scalable coding. While traditional mode selection strategies take into account the potential future drift stemming from coding the current frame with a particular enhancement layer coding mode, they usually fail to acknowledge the effect of past drift from previous partially reconstructed enhancement references. We introduce an optimal per-pixel drift estimation algorithm that calculates the effect of bandwidth variations and outages on the enhancement reference. Experimental results show the algorithm's advantages.

1. INTRODUCTION

Fine Granularity Scalable (FGS) video coding has emerged as an important research topic in recent years. Instead of compressing for a given target rate, it is desirable to compress for a range of bit rates at which the sequence can be potentially decoded. This is critical for internet video media streaming, because the Quality of Service policy of the internet service provider will not usually guarantee a constant bandwidth. The sole standardized effort on FGS video coding has been the MPEG-4 FGS Signal-to-Noise Ratio scalability extension [1]. The base layer consists of a standard single-layer MPEG-4 bitstream while the enhancement layer is coded with the bitplane technique and references only the base layer reconstruction of the image. Bitplane coding provides a completely embedded stream that can be arbitrarily truncated to fit the available bandwidth.

In [2], Wu et al. introduced progressive fine granularity scalability (PFGS), which uses an additional enhancement layer reference to improve motion prediction. Thus, assuming availability of the base layer and enhancement layer references, one frame is encoded with the former as a reference and the next one with the latter as a reference, alternating between those two layers. In [3], performance was improved by doing reference layer selection on a macroblock basis, yielding thus MB-PFGS. Most recently, He et al. [4] combined H.264/AVC with MB-PFGS to produce a scalable coder that outperformed MPEG-4 FGS, using an improved motion estimation scheme that employs information both from the base and the enhancement layer.

Rate-distortion optimization for scalable video coding was recently treated in [5]. A drawback of rate-distortion optimization is the added computational overhead of calculating the distortion and rate usage for every possible mode. Joint rate-distortion optimization of the base and enhancement layer yields a sizeable gain, but the drawbacks are even higher complexity and a base layer that is highly unoptimal if decoded on its own. In addition, distortion is usually obtained through models, limiting the accuracy of those methods. Motivated by this, and taking advantage of the findings in [4], we set out to devise a low complexity decision mechanism that does not use rate-distortion criteria, but relies instead on accurate drift estimation.

The paper is organized as follows. Section 2 gives an overview of the enhancement layer coding modes, and describes our algorithm for optimal per-pixel estimation. In Section 3 we discuss the algorithm implementation and in Section 4 experimental results and a brief discussion are presented. The paper is concluded in Section 5.

2. OPTIMAL PER-PIXEL ESTIMATION OF DRIFT

Base layer macroblocks (MBs) are encoded with one of the many possible modes defined in the standard. For the enhancement layer, every MB can be encoded with three possible coding modes (Fig. 1). Top dark gray squares denote base layers, bottom light gray squares denote enhancement references, and white squares with dashed lines denote partially decoded (top) or higher (bottom) enhancement layers. Base layer MBs are always reconstructed exclusively from previous base layers. Black arrows denote prediction, while white arrows denote reconstruction.

The first coding mode is *LPLR*, where an enhancement MB is predicted and reconstructed from the previous base layer. Using this mode, no prediction/reconstruction mismatch is possible and it also stops drift from previous frames. The coding efficiency is degraded due to the low quality motion compensation and reference.

The two other coding modes involve prediction from the enhancement layer reference. In the *HPHR* mode, the enhancement MB is both predicted and reconstructed from the enhancement layer reference. This mode provides the highest compression performance, if the previous enhancement reference was received in its entirety. If not, then we have drift. To counter this, the *HPLR* mode is used, where prediction still takes place from the enhancement reference, but reconstruction now uses the previous base layer. The quality is lower than HPHR, but drift is effectively contained.

The function $g(.)$ corresponds to motion compensation, hence $\beta = g(\alpha)$, where α is the previous frame, and β is the motion compensated prediction of the current frame. Let f_k denote the probability that the received enhancement layer portion has been truncated at rate R_k (i.e., available bandwidth at a particular moment is R_k), for $k = 0$ to $N-1$, where $R_i < R_k$ for $i < k$,

We would like to thank Dr. Feng Wu for making available the source code of the H.26L-PFGS codec. This work was supported in part by the National Science Foundation, the Office of Naval Research, and the CoRe program of the State of California.

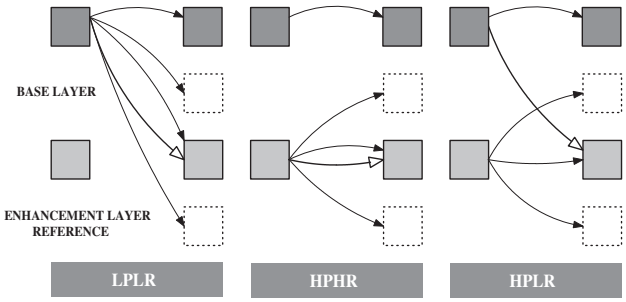


Fig. 1. Enhancement Layer Coding Modes.

and N is the number of discrete operational rates. Let R_{er} denote the enhancement reference rate. Even if rate R is available to the decoder, such that $R > R_{er}$, the enhancement reference will still be decoded at rate R_{er}. The frame decoded at rate R will be used only for display purposes by the decoder. It is left out of the decoding loop. Disregarding the effects of the loop filter and quarter-pel accurate motion compensation used in baseline H.264, we observe that, at the decoder, a reconstructed enhancement reference frame $\tilde{p}^i_{er}$ at frame number i can be written as:

$$\tilde{p}^i_{er} = g(p^{i-1}_b) + \tilde{r}^i \tag{1}$$

for LPLR and HPLR modes. Subscripts b and er refer to base layer and enhancement reference, respectively. Term p^{i-1}_b is a deterministic value known by both encoder and decoder, since the base layer is always assumed to be received in full. Term $\tilde{r}^i$, the reconstructed residue from the received segment of the enhancement layer, can vary according to channel conditions and thus has to be modeled, by the encoder, as a random variable. This residue differs for LPLR and HPLR because of the separate references, though the equations are unaffected. For HPHR we obtain:

$$\tilde{p}^i_{er} = g(\tilde{p}^{i-1}_{er}) + \tilde{r}^i \tag{2}$$

The previous enhancement reference frame $\tilde{p}^{i-1}_{er}$ has to be considered random by the encoder, since the encoder is not sure if the received portion of the enhancement layer was enough to reconstruct the enhancement reference frame in full. We use the expected value of the previous enhancement layer reference to write:

$$E\{\tilde{p}^i_{er}\} = g(E\{\tilde{p}^{i-1}_{er}\}) + E\{\tilde{r}^i\} \tag{3}$$

We use j to denote that value among the possible truncation rates where $R_{j-1} \leq R_{er} \leq R_j$, and the encoder can calculate:

$$E\{\tilde{p}^{i-1}_{er}\} = \sum_{k=0}^{j-1} f_k p^{i-1}_{er}(k) + p^{i-1}_{ER} \sum_{k=j}^{N-1} f_k \tag{4}$$

where p^{i-1}_{ER} denotes the enhancement reference frame reconstructed fully, and $p^{i-1}_{er}(k)$ denotes the enhancement frame reconstructed at rate R_k. For HPLR and LPLR modes, we have:

$$p^i_{er}(k) = g(p^{i-1}_b) + r^i(k) \tag{5}$$

and for HPHR mode we have:

$$p^i_{er}(k) = g(E\{\tilde{p}^{i-1}_{er}\}) + r^i(k) \tag{6}$$

For $k \geq j$ we set $p^{i-1}_{er}(k) = p^{i-1}_{ER}$ in computing the expected value in Equation (4), since the truncated rate is enough to recover the enhancement reference in full. In a similar manner to Equation (4), the encoder also calculates:

$$E\{\tilde{r}^i\} = \sum_{k=0}^{j-1} f_k r^i(k) + r^i_{ER} \sum_{k=j}^{N-1} f_k \tag{7}$$

where $r^i(k)$ denotes the enhancement residue truncated at rate R_k, and r^i_{ER} the enhancement residue required to reconstruct the enhancement reference in full. Thus, $E\{\tilde{p}^i_{er}\}$ can now be optimally calculated.

The recursive property of our algorithm is apparent, as $p^i_{er}(k)$ requires previously estimated values for its calculation. Per-pixel recursive estimation was previously shown to be effective in packet loss scenarios [6]. Hence we can now summarize our estimates for HPLR and LPLR as:

$$E\{\tilde{p}^i_{er}\} = g(p^{i-1}_b) + E\{\tilde{r}^i\} \tag{8}$$

and for HPHR as:

$$E\{\tilde{p}^i_{er}\} = g(E\{\tilde{p}^{i-1}_{er}\}) + E\{\tilde{r}^i\} \tag{9}$$

These equations are used at the encoder to calculate the estimates of drift optimally. This algorithm is called DEPP (Drift Estimate Per-Pixel).

3. ALGORITHM IMPLEMENTATION

Mode selection for the enhancement layer is accomplished by using the methodology in [3]. Instead of employing the enhancement reference to produce the predictions, we used our recursive per-pixel estimates. In [3], HPLR/HPHR mode selection for the enhancement layer is accomplished by choosing HPHR over HPLR when the following inequality is satisfied:

$$\|h - p_e\| \times k < \|p_b - p_e\| \tag{10}$$

where k is a constant, and h denotes the block in the original current frame, and choosing LPLR over either HPLR or HPHR if the first term of the following expression is smaller than the latter:

$$\min\left(\|p_b - \hat{p}_b\|, \|p_b - \hat{p}_e\|\right) \tag{11}$$

The DCT residues encoded in the enhancement layer are $p_b - \hat{p}_b$ and $p_b - \hat{p}_e$, respectively, for the LPLR mode and for either HPHR or HPLR. LPLR decoded reference segments will not propagate drift, because of the base layer p_b, since it is always received in full. Hat denotes reconstructed values.

In both of these expressions from [3], we replace p_e and $\hat{p}_e$, respectively, with the estimated predictions $g(p^{i-1}_b)$ or $g(E\{\tilde{p}^{i-1}_{er}\})$, depending on the mode. Now we only need calculate the term $E\{\tilde{p}^{i-1}_{er}\}$. In our implementation we set $f_k = 1$ and $N = 1$ for a given truncation rate $R_k < R_{er}$.

From Eq. (8) and (9) the recursive equations we employed are thus for LPLR and HPLR:

$$E\{\tilde{p}^i_{er}\} = g(p^{i-1}_b) + r^i(k) \tag{12}$$

and for HPHR:

$$E\{\tilde{p}^i_{er}\} = g(E\{\tilde{p}^{i-1}_{er}\}) + r^i(k) \tag{13}$$

where $r^i(k)$ corresponds to the aforementioned R_k.

We recursively estimate the enhancement references with Eq. (12) and (13). However, during mode selection, we only make use of the estimated predictions $g(p_b^{i-1})$ and $g(E\{\tilde{p}_{er}^{i-1}\})$ and we do not add the partial residue. Only after the enhancement layer bitstream has been fully produced, we update the estimates using Equations (7), (12) and (13), in contrast with ROPE [6] that uses the current estimates for mode selection. We instead employ the predictions from the previous estimated reference. This is done since the calculation of the current estimates requires the truncation of the enhancement layer under construction, and every enhancement mode decision we make changes the way the final layer will look. More complex implementations of our approach would be possible if the probabilities f_k were exactly known, and if we employed approximations of the truncated residuals for the drift calculation.

4. EXPERIMENTAL RESULTS AND DISCUSSION

We employed the H.26L-PFGS video codec, comprised of an H.264 TML9 base layer codec and an enhancement layer codec with MPEG-4 FGS syntax. The base layer bit rate ranges from 5.8 to 17.8 kbps depending on the particular sequence, as the choice of different motion vectors can lead to different bit rate requirements. The quantization parameter was set to $QP = 27$. We measured the performance of the scalable codec by truncating the enhancement bit rate of each frame in 250 byte / 2000 bit intervals. Because the frame rate is 10fps, this leads to intervals of 20kbps. The bit rate axis in Fig. 3 corresponds to the total transmission bit rate, comprised by the base layer that can vary, and the additional enhancement bit rate that comes in chunks of 20kbps. The term regular codec refers to the one in [4].

Integer motion vectors are employed, and the loop filter is disregarded. Regarding efficient techniques for adapting per-pixel estimates to fractional pel motion vectors, see [7]. We set $R_k = 0.5 \times R_{er}$ for $R_{er} = 90$kbps and $R_k = 0.65 \times R_{er}$, when $R_{er} = 70$kbps. This means that, regardless of how many 20kbps chunks of enhancement layer bits actually are received, the encoder runs its recursions by always assuming that network conditions force the enhancement layer to be truncated at some 50% or 65% of the rate needed for full reconstruction of the enhancement reference. The encoder thus *assumes* that there is drift on every enhancement reference, whether or not there actually is. Algorithm performance would improve if the encoder could accurately know the bandwidth available and the likely truncation rates.

From the experimental results in Fig. 3, we see that for one sequence there is a tiny performance loss for high to very high bit rates (where the PSNR is, in any case, over 35dB, and so the small loss is perceptually not significant), but for all sequences tested there is a substantial gain of 1dB at low to medium rates (rates where a 1dB gain is more perceptually important). For most sequences we see gains across all bit rates. If our approach is used to decide solely between HPHR and HPLR, it performs less well compared to using it for selection among LPLR, HPHR, and HPLR.

In Fig. 2 we provide results for variable bandwidth scenarios. Fig. 2(a) corresponds to the bandwidth variation pattern (enhancement truncation length in $10 \times$ kilobits) per 10 frames $BW_1 = [2, 4, 6, 8, 10, 12, 14, 16, 2, 4]$ and Fig. 2(b) corresponds to $BW_2 = [16, 14, 12, 10, 8, 6, 4, 2, 16, 14]$. Our proposed scheme outperforms the regular one [4] especially for abrupt transitions in the medium range of bit rates. Performance gains of more than 2dB for several frames are registered, while the average gain in PSNR is 0.8-0.9dB.

The memory complexity of this algorithm is modest, requiring the storage of an additional frame-sized matrix in floating-point format for each frame. In addition, single byte pel values are stored for entire decoded frame residuals truncated at the specified intermediate rates and then used for the updating step. In our implementation just one intermediate decoded residual was buffered. However, when more accurate channel bandwidth distribution information is available, $N > 1$ will have to be buffered.

Computational complexity is low and consists of a single recursive updating step. After the enhancement layer bitstream has been fully produced, the final updating and storage of the enhancement reference estimates, to be used for the coding of the next frame, takes place. This operation comprises a handful of add and memory access operations and a memory copy, and is computationally insignificant compared to rate-distortion base-layer mode selection or motion estimation.

One additional decoding step is used in our implementation to decode the truncated bitstream at an intermediate position and produce the partial residues. The complexity consists of applying inverse DCT and inverse quantization and then storing the pixel values. The calculated enhancement estimates are simply plugged into the MB-PFGS framework requiring no additional modification or complexity (except perhaps for the use of floating-point arithmetic in calculating the enhancement mode selection inequalities).

5. CONCLUSION

In conclusion, our low complexity drift estimation approach yielded substantial performance gains of about 1dB for most sequences across most truncation rates. This was true even though the encoder persisted with a simplistic assumption about the truncation rates, an assumption that did not hold true in the actual simulations, for which the enhancement reference truncation rates varied substantially. The reason is that even for $N = 1$ and hence a crude channel description, the recursion property of this algorithm imbues the codec with memory.

This drift estimation algorithm can be extended in various ways. First, it can be adapted to fractional-pel motion vectors, such as half-pel ones. Second, since the $\|.\|$ metric that corresponds to Mean Squared Error involves squares of estimates (random variables), the second moment $E\{(\tilde{p}_{er}^i)^2\}$ is required. Third, we can incorporate our drift estimator into a rate-distortion mode selection scheme that jointly optimizes over base and enhancement layer coding modes.

6. REFERENCES

[1] W. Li, "Overview of fine granularity scalability in MPEG-4 video standard," *IEEE Trans. on CSVT*, vol. 11, no. 3, pp. 301–317, Mar. 2001.

[2] F. Wu, S. Li, and Y.-Q. Zhang, "A framework for efficient progressive fine granularity scalable video coding," *IEEE Trans. on CSVT*, vol. 11, no. 3, pp. 332–344, Mar. 2001.

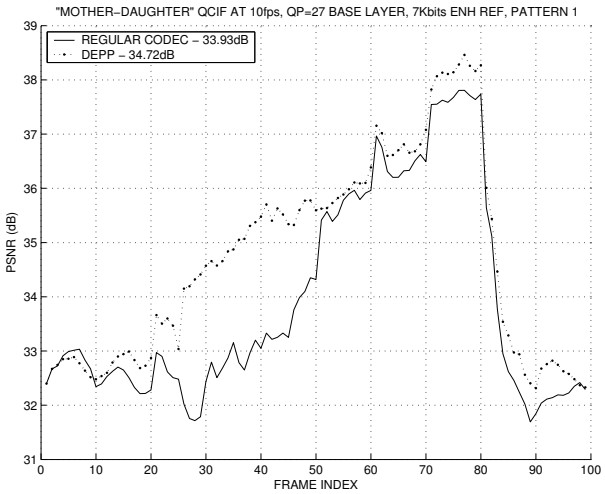
[3] X. Sun, F. Wu, S. Li, W. Gao, and Y.-Q. Zhang, "Macroblock-based progressive fine granularity scalable video coding," in *Proc. IEEE Int. Conf. on Mul. and Ex.*, 2001, pp. 461–464.

[4] Y. He, F. Wu, S. Li, Y. Zhong, and S. Yang, "H.26L-based fine granularity scalable video coding," in *Proc. IEEE ISCAS*, 2002, vol. IV, pp. 548–551.

[5] Z. Yang, F. Wu, and S. Li, "Rate distortion optimized mode decision in the scalable video coding," in *Proc. IEEE ICIP*, 2003, vol. 3, pp. 781–784.

[6] R. Zhang, S. L. Regunathan, and K. Rose, "Video coding with optimal inter/intra-mode switching for packet loss resilience," *IEEE Jnl. on Selected Areas in Comm.*, vol. 18, no. 6, pp. 966–976, June 2000.

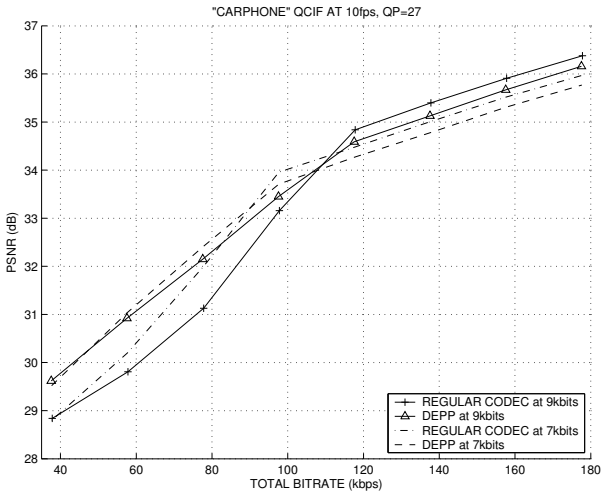
[7] A. Leontaris and P. C. Cosman, "Video compression with intra/inter mode switching and a dual frame buffer," in *Proc. IEEE DCC*, Mar. 2003, pp. 63–72.



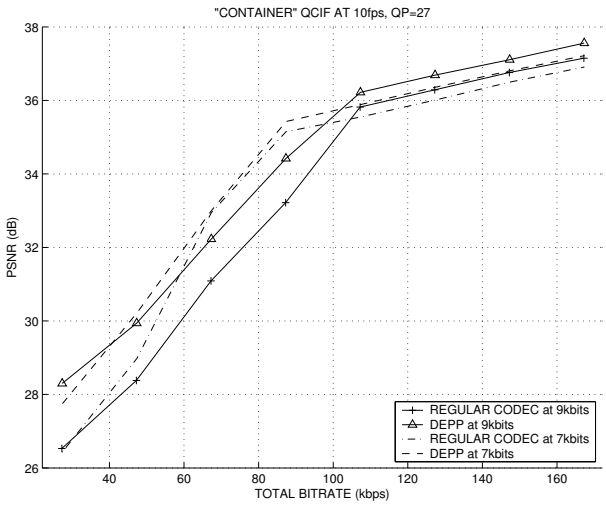
(a)

(b)

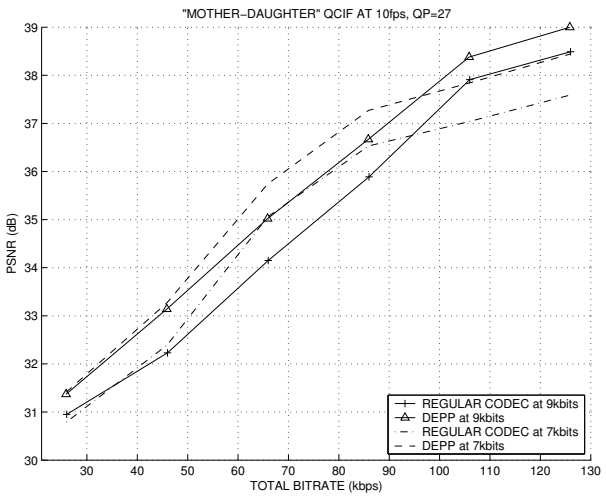
Fig. 2. PSNR vs. frame index for bandwidth variation pattern: (a) 1. (b) 2.



(a)



(b)



(c)

Fig. 3. PSNR vs. bit rate (a) Carphone. (b) Container. (c) Mother-Daughter.