Global MINMAX Interframe Bit Allocation for Embedded Video Coding *

Michael F. Ringenburg, Richard E. Ladner,† Eve A. Riskin‡

Abstract

We present three new bit allocation techniques, the MultiStage algorithm, the Ratio algorithm, and the Pseudo I-frame method. The MultiStage and Ratio algorithms are global interframe bit allocation algorithms for embedded video coders, which attempt to minimize the maximum distortion of any frame. The Pseudo I-frame method breaks a video into segments where bit allocation algorithms should be run, and segments where constant rate bit allocation will suffice. We present experiments, using the Group Testing for Video coder [14], that compare our techniques to constant rate allocation and to a bit allocation algorithm due to Yang and Hemami [18].

1 Introduction

Many applications, such as DVD players and streaming Internet video, use variable bit rate compressed video streams which can be encoded off-line. Interframe bit allocation determines the number of bits to allocate to each frame given a bit budget B. Some algorithms attempt to minimize the average distortion of the video's frames (e.g., [2]); other algorithms attempt to minimize the maximum distortion of any frame (e.g., [18]); and still others work to satisfy buffer constraints (e.g., MPEG TM 5 [1]).

Embedded video coders allow for more flexible bit allocation because we can specify the exact number of bits to allocate to each frame, as opposed to adjusting quantizer step sizes as in coders based on quantization. Some objectives are simple to achieve under this framework. For instance, a common objective is to prevent buffer underflow at the receiver. We assume the receiver has a buffer which is filled at the constant bit rate C of the channel, and emptied at the variable bit rate of the video stream—i.e., decoding frame 1 removes v_1 bits from the buffer, decoding frame 2 removes v_2 bits from the buffer, etc. Underflow can be avoided by simply allocating C/h bits to each

*Supported in part by the National Science Foundation under grant number CCR-0104800, by the Army Research Office under grant number DAAD1919-02-1-0309 and by an Achievement Rewards for College Scientists (ARCS) Fellowship sponsored by the Washington Research Foundation (WRF).

†Address: Department of Computer Science and Engineering, Box 352350, University of Washington, Seattle, WA 98195–2350, Tel: +1-206-543-1695; Fax: +1-206-543-2969. E-mail: *miker,ladner@cs.washington.edu*.

‡Address: Department of Electrical Engineering Box 352500, University of Washington, Seattle, WA 98195–2500, Tel: +1-206-543-2150; Fax: +1-206-543-3842. E-mail: *riskin@ee.washington.edu*.

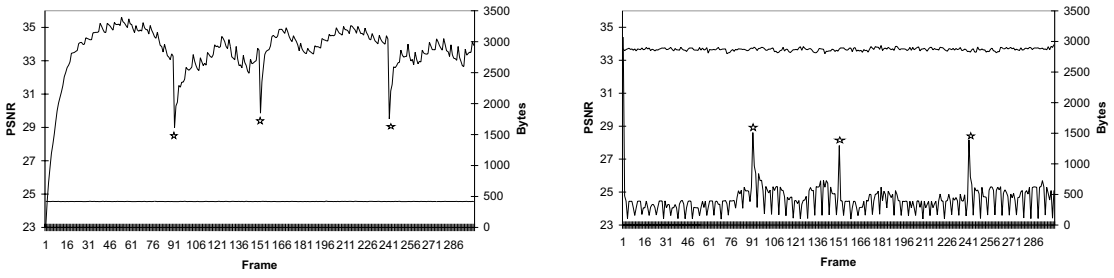

Figure 1: The bit rate and quality (PSNR) of each frame of the News video at 100 kbps and 30 fps after coding with GTV and applying (a) constant rate bit allocation and (b) our new MultiStage algorithm. The bottom line tracks the number of bytes allocated to each frame, and the top line tracks the PSNR of each frame. The stars indicate scene changes.

frame, where h is the frame rate. Unfortunately, a constant bit rate does not usually result in constant quality, as can be seen in Figure 1(a). Some frames require more bits to encode to the same quality because they are poorly predicted or because they have less regularity.

These frame-to-frame variations in quality are undesirable because they often appear as "flicker." Thus, in this paper we propose two global interframe bit allocation algorithms. The MultiStage algorithm achieves nearly constant video quality (see Figure 1b). The Ratio algorithm is faster than the Multistage algorithm, but still produces significantly less variation than constant rate coding. The switch from a constant bit rate to a variable bit rate may force a delay at the decoder to prevent buffer underflow. To combat this, we discuss a buffer control modification to these algorithms which allows the encoder to specify a maximum acceptable delay. We also propose a new approach called "Pseudo I-frames," to determine when bit allocation algorithms are necessary and when constant bit rate coding suffices. We evaluate the algorithms using the University of Washington's Group Testing for Video (GTV) coder [14].

In the rest of this section, we give an overview of standard video compression techniques, describe the GTV coder, and formalize the interframe bit allocation problem. In the next section, we discuss related work. In section 3, we describe the Multistage and Ratio algorithms, and the Pseudo I-frame method. In section 4, we compare the results of the algorithms. Finally, in section 5 we present some concluding remarks.

1.1 Video Compression and the GTV Coder

Most modern video coders use a three step process to encode each frame. The coder begins by running a motion compensation algorithm to generate a set of motion vectors which, when applied to previous and/or future frames, generates a predicted current frame. The difference between the predicted and actual current frames is the residual. In the second phase, the coder computes a transform, such as the Discrete Cosine Transform (DCT), on the residual frame to concentrate most of the energy in a few coefficients. In the last stage, the transformed residual is encoded using a lossy

compression scheme.

The Group Testing for Video (GTV) coder [14] is based on the Group Testing DCT Image coder [5]. The GTV coder includes only forward prediction—i.e., the motion compensation algorithm uses only the previous frame to predict the current frame. Residual frames are transformed using the 8×8 DCT, and the transformed residuals are encoded using the bit-plane coding and group testing process described by Hong, Ladner, and Riskin [5]. For our purposes, the most important feature of bit-plane coding is that it is embedded, which allows the bit allocation algorithms to specify an exact size for each frame.

1.2 The Bit Allocation Problem

In the bit allocation problem, we are given n frames $F_1, F_2, \ldots, F_n$ and a total bit budget B. We choose a number of bits v_i to allocate to each frame F_i such that $\sum_{i=1}^{n} v_i \leq B$. The quality of each frame F_i is a function $q_i(v_1, \ldots, v_n)$ of the number of bits allocated to F_i and to every other frame. If we limit ourselves to only forward prediction, then q_i is a function only of the allocations to the current and previous frames—thus we can write $q_i(v_1, \ldots, v_n) = q_i(v_1, \ldots, v_i)$.

We also consider the case where we have a constraint that we wish to avoid buffer underflow. Streaming video applications typically use a buffer on the receiving end to store incoming bits. The buffer is filled at the constant rate of the incoming channel, and is emptied at the variable rate of the video. Buffer underflow occurs if we attempt to remove more bits than the buffer contains. When this occurs, we must wait for more data before the next frame can be displayed. In the case we consider, we are given a buffer which is preloaded by introducing a pre-specified delay. The initial buffer occupancy O_0 will then be the delay D times the rate of the CBR channel C. In other words, $O_0 = DC$. After each frame F_i, we add C/h bits to the buffer, where h is the frame rate, and remove v_i bits. Thus, for $i > 0$, we have $O_i = O_{i-1} + (C/h) - v_i = DC + i(C/h) - \sum_{j=1}^{i} v_j$. Our constraint is that $O_i \geq 0$ for all $i \geq 0$. In other words, we need to ensure that when the receiver is ready to decode frame i, all of the bits that it needs are in the buffer.

Yang and Hemami [18] mention two commonly used objective functions for measuring the success of bit allocation. The first approach (taken by [2] and [7]) attempts to maximize the average quality or, equivalently, to minimize the average distortion of all the frames (the Mean Mean-Squared Error, or MMSE, approach). In [18], [7], [16] and [4], they instead maximize the minimum quality or, equivalently, minimize the maximum distortion of any frame (the MINMAX approach). This yields a more constant quality, which reduces "flicker." For this reason, we choose to use MINMAX in this work.

2 Related Work

The bit allocation problem has received a great deal of attention in the literature. Mohr [8] showed that the general bit-allocation problem is equivalent to the Multiple-

Choice Knapsack Problem, and is thus NP-hard. A number of authors studied bit allocation in the context of image compression [10, 15, 17, 3]. More recent work examined the bit allocation problem in the context of video compression. Ramchandran, Ortega, and Vetterli [9] were among the first to examine bit allocation in the context of dependent coding (such as motion-compensated video). They focused on quantizer-based coders such as MPEG. Sermadevi and Hemami [13] use piecewise linearity, convexity, and independent coding assumptions to derive a linear programming based MINMAX bit allocation algorithm. They then show that the derived algorithm can be successfully applied to MPEG-style video coders even though the linearity, convexity and independence assumptions do not generally hold. MINMAX style algorithms are also discussed by Hoang [4], Uz [16] and Lin and Ortega [7]. Lin and Ortega also discuss MMSE methods. Schuster *et al.* [12] compare MMSE and MINMAX bit allocation methods. Lee and Ortega [6] examine MINMAX bit allocation in the presence of buffer constraints.

Recent work also investigates bit allocation methods for embedded video coders. Cheng, Li, and Kuo [2] derive an MMSE bit allocation algorithm for a wavelet-based coder using the Lagrangian method and estimates of coding efficiency and frame dependency parameters for each frame. Yang and Hemami [18] were the first to propose a MINMAX algorithm for embedded video coders. Their algorithm consists of an initial estimation stage for the first two frames of every group of pictures (GOP), and an adaptive adjustment stage for the subsequent frames of the GOP. This algorithm is related to ours, in that it is also designed for embedded coders and targets the MINMAX metric. In addition, their initial estimation stage inspired the MultiStage algorithm's constant quality stage (described in section 3.1). In Section 4 we experimentally compare it to our algorithm.

3 Bit-Allocation Algorithms

In this section, we describe our new interframe bit allocation methods. In sections 3.1 and 3.2, we present the MultiStage and Ratio algorithms. Section 3.3 describes how these algorithms can be modified to satisfy buffer underflow constraints. Finally, in section 3.4 we propose "Pseudo I-frames"—a simple method for dividing a video into interframe allocated portions and constant-rate portions.

3.1 The MultiStage Algorithm

The MultiStage algorithm is so named because it alternates between two distinct stages—the Constant Quality stage and the Target Rate stage. At the beginning of the process, the Constant Quality stage encodes every frame of the video to an initial PSNR which is a parameter of the algorithm. In our experiments, we chose an initial PSNR of 40.00 dB. Let $v_i, i = 1, \ldots, n$ be the total number of bits allocated to frame i, and let $T = \sum_{i=1}^{n} v_i$ be the total number of bits allocated to all frames. We next enter the Target Rate stage, in which we encode each frame i using $v_i' = (B/T)v_i$

bits[1], where B is our total bit budget. Our new allocation is proportional to the previous allocation, but uses exactly B bits. Thus frames that need more bits to maintain a constant quality will be assigned more bits. Now, let $p_i, i = 1, \ldots, n$ be the PSNR of frame i after this encoding. We next repeat the process, except that in the Constant Quality stage, we encode every frame to $(\sum_{i=1}^{n} p_i)/n$ (i.e., the average PSNR of all the frames after the Target Rate stage encoding). Note that a variant of this algorithm could instead code to the average MSE.

The two stages continue to alternate, with each stage providing feedback to the next stage, until one of two termination conditions occurs. After each iteration of the Constant Quality stage, the quality of every frame will be approximately the same ($p_1 \approx p_2 \approx \ldots \approx p_n$). This is our objective, so we check to see if $|T - B|/B \leq \varepsilon$, i.e., if the total allocation is within a factor ε of the total bit budget. If this check passes, then we terminate with the allocation found by the Constant Quality stage. In our experiments, we set $\varepsilon = 0.01$ (1%). The other termination condition occurs after the Target Rate stage. At the end of this stage, the total allocation will be the bit budget ($T = B$). Thus, we check if the quality of each frame is close to constant. We measure this using the variance of the PSNR's. If the variance of the frame PSNRs,

$$\sum_{i=1}^{n} \frac{p_i^2}{n} - \left(\sum_{i=1}^{n} \frac{p_i}{n} \right)^2 \leq \delta \,,$$

where δ is a parameter to the algorithm, then we terminate with the exact target total allocation and a nearly constant quality. In our experiments, we set $\delta = 0.1$. In all of the experiments described in this work, the second condition was always satisfied before the first condition—thus we always achieved the exact target bit budget and a nearly constant quality.

Note that there is no guarantee that one of these termination conditions will ever be met. However, in all of our experiments with δ and ϵ set as above, the algorithm terminated after two full iterations, where a full iteration consists of one iteration of the Constant Quality stage and one iteration of the Target Rate stage. Thus, four passes over the data were required. Note that we expect the MultiStage algorithm to converge faster then the bisection algorithm ([12]) because a plausible target distortion is chosen at the beginning of each Constant Quality stage.

Also note that, if we are in a situation where buffer underflow is a concern, this algorithm does not guarantee that the buffer underflow avoidance constraint will be satisfied. In section 3.3 we suggest a simple modification to the algorithm which ensures that underflow is avoided.

3.2 The Ratio Algorithm

The Ratio algorithm is designed to be a simple, fast algorithm for situations where encoding time is a concern. It gives better performance than constant-rate allocation, but is still fast and simple to implement. The algorithm is based on the empirical

[1]Since we can not have fractional allocations, we need to round off the values of v_i'. We can track how much we have gained or lost by rounding, and use that to choose whether to round up or down.

observation that the number of bits necessary to encode a frame to a given PSNR using GTV is generally approximately proportional to the norm (the square root of the sum of the squares of the coefficients) of the residual.

The Ratio algorithm begins by making a quick pass over the video, calculating the norm of each residual assuming *no* transform and a perfect encoding of the previous frame. This pass will be significantly faster than a regular pass over the video, because at each frame we only need to run the motion compensation algorithm using that frame and the previous frame. On average, this takes about 60% as long as a normal pass. We do not need to compute the transform because the norm is preserved under orthogonal transforms, and we do not need to encode or decode because we are assuming a perfect reference frame. However, because of this assumption, these residuals will be approximations.

We use these approximate residuals to compute an allocation. Let r_i be the norm of the residual of frame i, and let $R = \sum_{i=1}^{n} r_i$. Originally, we let our allocations be $v_i = r_i B/R$. However, we noticed that this occasionally led to extremely large allocations at scene changes, which forced other frames to receive very few bits. To correct for this, we added upper and lower bounds. For any i, if $r_i > 5R/n$ we set $r_i = 5R/n$, and if $r_i < 0.5R/n$, we set $r_i = 0.5R/n$. We then recompute our allocations using these bounded norms.

Like the MultiStage algorithm, the Ratio algorithm does not guarantee that the buffer underflow constraint will be satisfied. This is addressed in the next section.

3.3 Buffer Control

In sections 3.1 and 3.2, we mentioned that the MultiStage and Ratio algorithms may both fail to satisfy the buffer underflow constraint. Recall from section 1.2 that we assume the buffer is initially preloaded by delaying playback by some delay D. The initial buffer occupancy will then be $O_0 = DC$, where C is the rate of the channel. The buffer occupancy after frame i will be

$$O_i = O_{i-1} + (C/h) - v_i = DC + i(C/h) - \sum_{j=1}^{i} v_j$$

where h is the frame rate. Buffer underflow occurs when $O_i < 0$ after some frame i.

Our buffer control modification is based on the observation that a buffer underflow of u bits at frame j (i.e., $O_j = -u$) can be fixed by removing a total of u bits from some combination of the previous and current frames F_1 through F_j. Our modification divides the u bits which must be removed between the j frames in proportion to their original allocations. In other words, if $B_j = \sum_{i=1}^{j} v_i$ then our new allocations are

$$v_i' = \frac{v_i(B_j - u)}{B_j}, i = 1, \ldots, j\ .$$

We then add the removed bits to the frames after F_j, once again in proportion to their original allocations. Our new allocations for frames F_{j+1} through F_n are

$$v_i' = \frac{v_i(B - B_j + u)}{B - B_j}, i = j+1, \ldots, n\ .$$

3.4 "Pseudo I-Frames"

When encoding time is an issue, it may be desirable to only run bit allocation algorithms on small portions of the video. Graphs of the PSNR during constant-rate coding (such as Figure 1a), often contain a small number of large "dips" (at the stars in Figure 1a). The rest of the video, however, is generally of fairly constant quality. The dips typically occur when there is a large residual to encode, due to poor prediction such as at a scene change. Because these frames perform similarly to I-frames, we refer to them as "pseudo I-frames."

The pseudo I-frame method encodes the video at the constant rate B/n, until it finds a pseudo I-frame. When a pseudo I-frame is found at some frame j, we treat the G frames $F_j, \ldots, F_{j+G-1}$ as an independent Group of Pictures (GOP), and run a bit allocation algorithm on the group with a bit budget of $B_{GOP} = B(G/n)$. Once the allocation completes, we resume constant bit rate coding at frame F_{j+G}. We can identify pseudo I-frames by simply looking at the norm of the residual. If the norm is above some tunable threshold z, we declare the frame to be a pseudo I-frame.

4 Results

This section describes the results of the experiments we ran to test our techniques. All tests had a channel rate of 100 kbps and a frame rate of 30 fps. We used the GTV coder [14] and two QCIF (176×144) videos, News and Trevor. The News video contains three scene changes which only impact a portion of the frame. The Trevor video contains a single scene change which affects the entire frame.

In Figure 2 we compare the results of running the MultiStage and Ratio algorithms with the results of Yang and Hemami's algorithm and constant rate allocation. We chose a GOP size of 30 frames for Yang and Hemami's algorithm, as in [18]. In both experiments, the MultiStage algorithm results in the most consistent PSNR and the highest minimum PSNR. The Ratio algorithm was second in both categories. The MultiStage and Ratio algorithms also avoid the PSNR dips at the scene changes.

In Figure 3 we investigate the results of combining the Pseudo I-frame method with the MultiStage algorithm. We use a GOP size of 30. The graphs show that the combined method results in less PSNR consistency. We are no longer able to trade off some of the quality in easy to code scenes to improve the quality of harder to code scenes, because the method no longer performs global allocation. However, the coding process is more computationally efficient and we still avoid some of the dips at scene changes.

In Figure 4 we show the results of adding buffer control to the MultiStage algorithm. We did not see buffer underflow for every video, so we only examine Trevor, which experienced buffer underflow with the MultiStage algorithm. We assumed a one second (30 frame) delay. We see that the effects of buffer control were negligible in the Trevor video. The quality prior to the frame where underflow occurs without buffer control is slightly degraded and the quality after that frame is slightly improved. These differences, however, are small. In some of the other experiments we

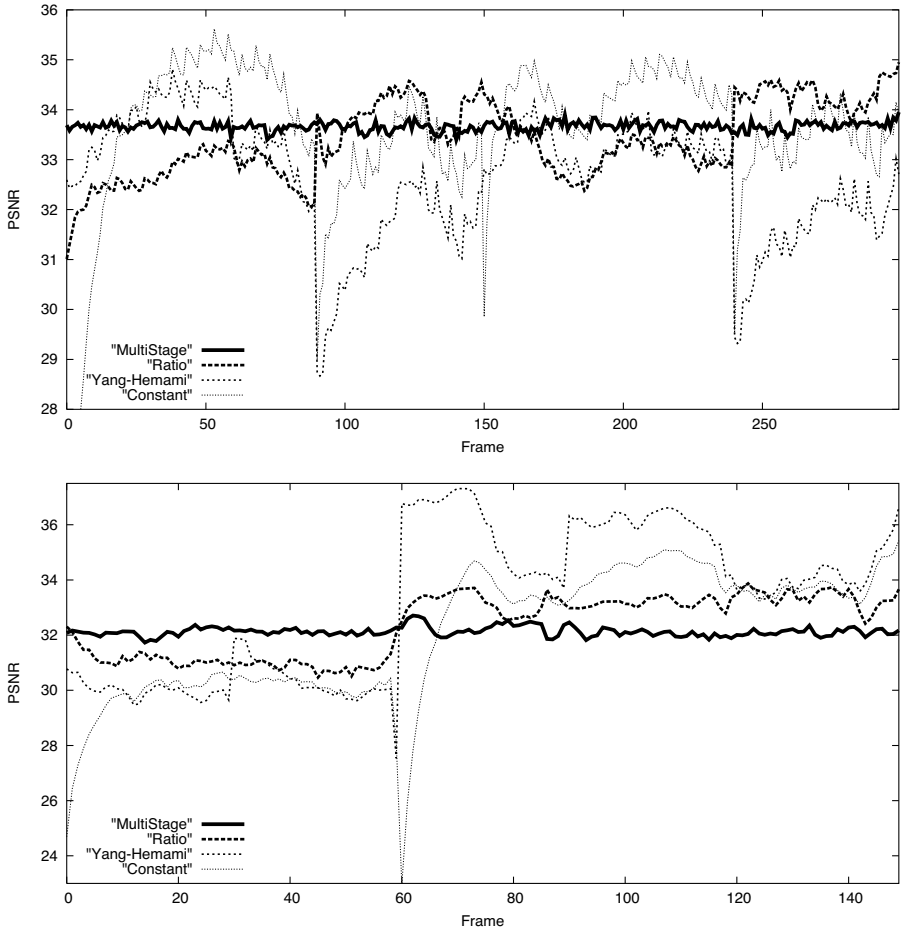

Figure 2: **Algorithms** A comparison of the MultiStage, Ratio, Yang-Hemami, and Constant Rate algorithms for the videos News (top) and Trevor (bottom).

ran (described in [11]), though, buffer control had a more significant impact. This occurred in videos whose first scene was significantly more difficult to code than the rest of the scenes. This is the worst case scenario because a disproportionate number of the bits must be allocated to the beginning of the video in order to achieve consistent quality. Stacking the bits at the beginning of the video leads to a large, early underflow because the buffer is quickly depleted before it has a chance to fill up.

The following table summarizes our results:

Video	Method	Minimum PSNR	Variance	Delay
News	MultiStage	33.40	0.01	0.22 sec
10 sec	MultiStage, Pseudo I	32.02	0.66	GOP
	Yang-Hemami	28.67	1.48	GOP
	Ratio	31.01	0.58	0.14 sec
	Constant	22.97	2.58	0
Trevor	MultiStage	31.73	0.03	1.22 sec
5 sec	MultiStage, Pseudo I	28.06	3.90	GOP
	Yang-Hemami	27.54	7.51	GOP
	Ratio	30.46	1.27	0.59 sec
	Constant	22.93	5.69	0

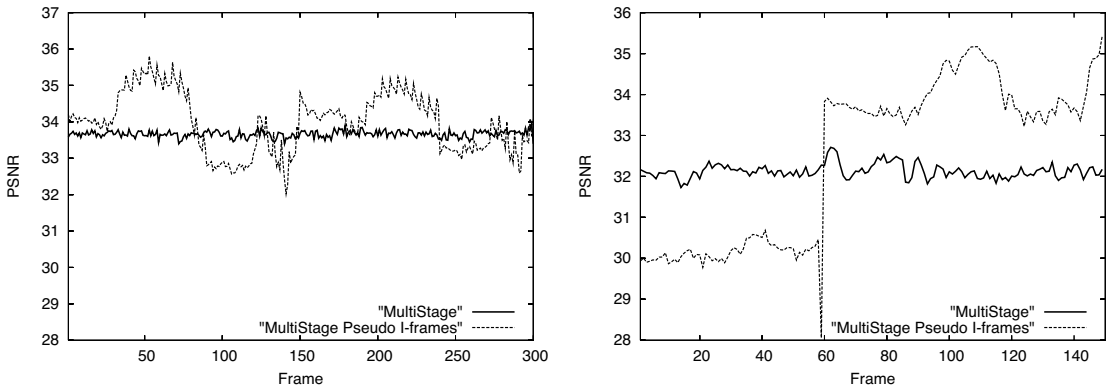

Figure 3: **Pseudo I-frames.** A comparison of running the MultiStage algorithm only after pseudo I-frames with running it on the whole video, on News (left) and Trevor (right).

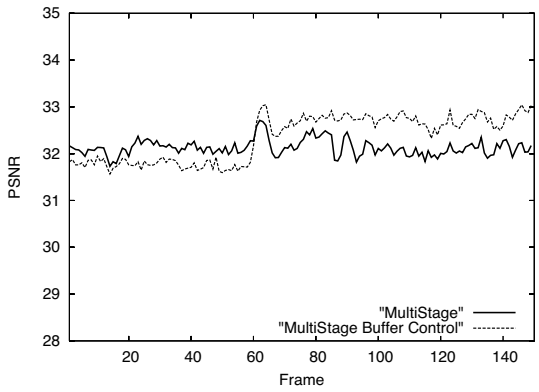

Figure 4: **Buffer** The above graph shows the effect of adding buffer control to the MultiStage algorithm on Trevor. The x and y axes indicate frame number and PSNR, respectively.

The variance column measures the variance from the mean of the frame PSNR's. The minimum PSNR column lists the minimum frame PSNR. We also measure the delay necessary to avoid underflow. For the Yang-Hemami and Pseudo I-frame methods, one GOP (1 second in our experiments) is always sufficient.

5 Conclusions

We proposed two new global bit allocation methods for embedded video coders, the MultiStage algorithm and the Ratio algorithm. We compared our algorithms to an algorithm due to Yang and Hemami [18] and to constant rate bit allocation. We showed that on the videos tested, the MultiStage algorithm had the lowest variance and the highest minimum PSNR. The Ratio algorithm was second in both categories.

We also proposed a method to determine when to run bit allocation algorithms, and when to simply use constant rate bit allocation. We called our approach the "Pseudo I-frame" method. We combined it with the MultiStage algorithm and showed that it did not minimize the PSNR variance as well as the MultiStage algorithm on its own. However, it saved time and it still managed to eliminate the scene change PSNR dips.

References

[1] MPEG-2 Test Model 5. *ISO-IEC AVC-491*, Apr. 1993.

[2] P. Cheng, J. Li, and C.-C. Kuo. Rate control for an embedded wavelet video coder. *IEEE Trans. Circuits and Systems for Video Technology*, 7(4):696–702, 1997.

[3] M. Effros and P. A. Chou. Weighted universal bit allocation: optimal multiple quantization matrix coding. In *Proc. ICASSP*, pages 2343–46, 1995.

[4] D. T. Hoang, E. L. Linzer, and J. S. Vitter. Lexicographic bit allocation for MPEG video. *J. Visual Communication and Image Representation*, 8(4):384, 1997.

[5] E. Hong, R. E. Ladner, and E. A. Riskin. Group testing for image compression using alternative transforms. *Signal Proc: Image Communications*, 18:561–74, Aug. 2003.

[6] S.-Y. Lee and A. Ortega. Optimal rate control for video transmission over VBR channels based on a hybrid MMAX/MMSE criterion. In *Proc. ICME*, pages 93–96, 2002.

[7] J. Lin and A. Ortega. Bit-rate control using piecewise approximated rate-distortion characteristics. *IEEE Trans. Circuits and Systems for Video Tech.*, 8(4):446–59, 1998.

[8] A. E. Mohr. Bit allocation in sub-linear time and the multiple-choice knapsack problem. In *IEEE Data Compression Conference*, pages 352–361, Apr. 2002.

[9] K. Ramchandran, A. Ortega, and M. Vetterli. Bit allocation for dependent quantization with applications to multiresolution and MPEG video coders. *IEEE Trans. Image Processing*, 3(5):533–545, Sept. 1994.

[10] K. Ramchandran and M. Vetterli. Best wavelet packet bases in a rate-distortion sense. *IEEE Trans. Image Processing*, June 1993.

[11] M. F. Ringenburg. Global MINMAX interframe bit allocation for embedded video coding. At "http://www.cs.washington.edu/homes/miker/bitalloc-quals.pdf", Oct. 2003.

[12] G. M. Schuster, G. Melnikov, and A. K. Katsaggelos. A review of the minimum maximum criterion for optimal bit allocation among dependent quantizers. *IEEE Trans. Multimedia*, 1(1):3–17, Mar. 1999.

[13] Y. Sermadevi and S. Hemami. Linear programming optimization for video coding under multiple constraints. In *IEEE Data Compression Conference*, pages 53–62, 2003.

[14] G. Shavit, M. F. Ringenburg, J. West, R. E. Ladner, and E. A. Riskin. Group testing for video compression. In *IEEE Data Compression Conference*, Mar. 2004.

[15] Y. Shoham and A. Gersho. Efficient bit allocation for an arbitrary set of quantizers. *IEEE Trans. Signal Proc.*, 36:1445–1453, Sept. 1988.

[16] K. Uz, J. M. Shapiro, and M. Czigler. Optimal bit allocation in the presence of quantizer feedback. In *Proc. ICASSP*, pages 385–388, Apr. 1993.

[17] P. H. Westerink, J. Biemond, and D. E. Boekee. An optimal bit allocation algorithm for sub-band coding. In *Proc. ICASSP*, pages 757–760, 1988.

[18] Y. Yang and S. S. Hemami. Minmax frame rate control using a rate-distortion optimized wavelet coder. In *IEEE Int. Conf. Image Processing*, Kobe, Japan, Oct. 1999.