MultiStage: A MINMAX Bit Allocation Algorithm for Video Coders

Neva Cherniavsky, Gidon Shavit, Michael F. Ringenburg, Richard E. Ladner, *Senior Member, IEEE*, and Eve A. Riskin, *Senior Member, IEEE*

***Abstract*—Most bit allocation algorithms for video are geared toward optimizing the average frame distortion. However, video sequences optimized this way may exhibit sudden changes in distortion, or "flicker," which can significantly affect the perceived quality of the sequence. An alternative approach is to minimize the *maximum* frame distortion, which aims to produce a constant-quality sequence, thus avoiding the flicker problem.**

In this work, we present a new algorithm for constant-quality video, called *MultiStage*. We first show how MultiStage works for an embedded bit plane coder, and we then demonstrate that it can be applied to traditional quantization-based coders, such as H.263 and H.264, in conjunction with a novel single-frame block-level rate-distortion optimization algorithm based on multiple-choice knapsack. We show that *MultiStage* achieves very good results, both in terms of maximum distortion and average distortion.

***Index Terms*—Constant quality, H.263, H.264, rate control, rate-distortion optimization (RDO), variable bit rate (VBR), video coding.**

I. Introduction

Many applications, such as DVD players and streaming Internet video, use variable bit rate compressed video streams which can be encoded offline. Since these streams are compressed once and then played back many times, speed of encoding is less important than the quality achieved at a given bit rate. Interframe bit allocation determines the number of bits to allocate to each frame given a total bit budget B for the entire video. Some algorithms attempt to minimize the average distortion of the video's frames (e.g., [1]); other algorithms attempt to minimize the maximum distortion of any frame (e.g., [2]); and still others work to satisfy buffer constraints (e.g., MPEG TM 5 [3]).

Manuscript received February 25, 2006; revised August 14, 2006. This work was supported in part by the National Science Foundation (NSF) under Grants CCR-0104800 and CCF-0514353, in part by the Army Research Office (ARO) under Grant DAAD1919-02-1-0309, and in part by the National Institute of Health (NIH) under Grant 5 R01 GM 068878-02. The work of N. Cherniavsky was supported in part by a NSF graduate research fellowship. Preliminary versions of this work appeared in IEEE Data Compression Conference (DCC)'04 and IEEE Data Compression Conference (DCC)'05. This paper was recommended by Associate Editor M. Comer.

N. Cherniavsky and R. E. Ladner are with the Department of Computer Science and Engineering, University of Washington, Seattle, WA 98195-2350 USA (e-mail: nchernia@cs.washington.edu; ladner@cs.washington.edu).

G. Shavit is with Medio Systems, Seattle, WA 98101 USA (e-mail: g_shavit@yahoo.com)

M. F. Ringenburg is with Cray, Inc., Seattle, WA 98104 USA.

E. A. Riskin is with the Department of Electrical Engineering, University of Washington, Seattle, WA 98195-2500 USA (e-mail: riskin@ee.washington.edu).

Digital Object Identifier 10.1109/TCSVT.2006.887135

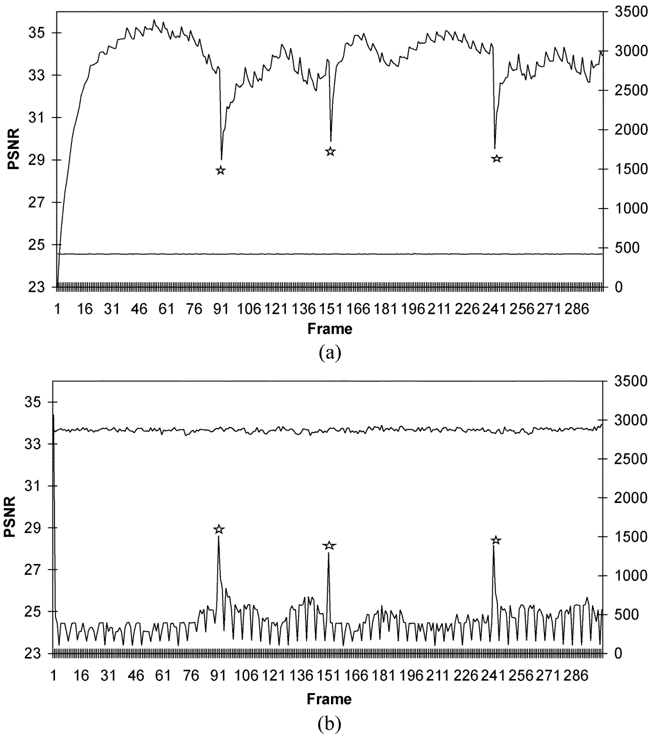


Fig. 1. Bit rate and quality (PSNR) of each frame of the News video at 100 kbps and 30 fps after coding with GTV and applying (a) constant rate bit allocation and (b) our new MultiStage algorithm. The top line tracks the PSNR of each frame and the bottom line tracks the number of bytes allocated to each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

Embedded video coders allow for 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 frame, where h is the frame rate. Unfortunately, a constant bit rate does not usually result in constant quality, as can be seen in Fig. 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." In this paper we propose MultiStage, a global interframe bit allocation algorithm that achieves nearly constant video quality [see Fig. 1(b)]. 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 first apply the algorithm using the University of Washington's Group Testing for Video (GTV) coder [4], an embedded video coder. We then use MultiStage on the quantization-based coders H.263 and H.264.

In the next section, we discuss the background of the bit allocation problem and related work. In Section III, we describe the MultiStage algorithm and its application to the three different video coders. Section IV contains our experimental results and Section V, the conclusion and suggestions for future research.

II. Background and Related Work

In this section, we formalize the interframe bit allocation problem, give an overview of standard video compression techniques and describe the GTV coder, H.263, and H.264.

A. 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. (Another objective might be to prevent buffer overflow, but since memory is inexpensive, we can assume our buffer is big enough that overflow is not a problem.) 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. To prevent underflow, we introduce a start-up delay by preloading the buffer. The initial buffer occupancy O_0 will then be the delay D times the rate of the constant bit rate 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 buffer occupancy

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

after frame i. Underflow is prevented if the constraints $O_i \geq 0$ for all $i \geq 0$ are satisfied. 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.

The frame level bit allocation problem has received a great deal of attention in the literature. Mohr [5] 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 [6]–[9].

More recent work examined the bit allocation problem in the context of video compression. When frames are quantized independently, it can be optimally solved using Shoham and Gersho's Lagrangian relaxation [7]. Most video coders, however, employ some form of predictive coding, resulting in interframe dependencies. A naïve Lagrangian approach to the dependent problem would be exponentially complex in the number of frames, which is prohibitive.

Several authors attempt to reduce complexity through pruning heuristics [10], by assuming finite memory and using dynamic programming [11], or through gradient descent [12]. In all of the above, the complexity is heavily dependent (at least quadratic and often exponential) on the prediction depth. Others (e.g., [13]) rely on a model-based approach to come up with a closed-form solution based on inexpensive preanalysis; however, models simple enough for analysis tend to be inaccurate. Another critical parameter is the number of operating rate-distortion (R-D) points that is considered for each frame.

Almost all the work in this area has attempted to optimize the average (or total) distortion of the sequence (MINAVE). While this is typically easier to analyze, especially in the independent case, it is not necessarily the best measure for perceptual quality. A MINAVE optimized video sequence may exhibit sudden variations in fidelity, or "flicker," which can be distracting.

Two alternative optimization goals, aimed at addressing this issue, are minimizing the maximum distortion (MINMAX) or distortion variation (MINVAR or MDV) objectives. Both metrics reduce quality variation between frames [14] but there is no general agreement in the research community regarding which is better. We target MINMAX because it is uniquely defined and easy to minimize. On the other hand, MINVAR can be defined in multiple ways (for example, as total variation or maximum frame-to-frame variation) and it is not immediately obvious how to minimize such a target.

Among those who target MINMAX, Schuster *et al.* use bisection to find the minimum distortion [11], while Wang and Woods use a similar but model-based approach [15] and Lee and Ortega iteratively modify the frame-level quantization parameter (QP) of the currently maximum-distortion frame [16]. Xie and Zeng [17] allocate bits based on the scene complexity ratio while Chen and Ngan [18] modify the distortion criteria and try to reduce buffer delay. Hoang *et al.* [19] provide a framework for lexicographic optimization, which is related to MINMAX, while others attempt to optimize the MINVAR objective [20], [21]. Many of the above are one-pass or real-time methods; we are able to achieve closer to constant quality since our algorithm is offline.

We measure quality in terms of peak signal-to-noise ratio (PSNR). While this is quite common in the compression literature, PSNR is an imperfect measure, since frames may have widely varying characteristics. For example, at a scene change, equalizing PSNR may not correspond with the perception of quality, due to the way the visual system processes abrupt change. Our algorithm would work equally well with metrics that measure perceptual quality, such as those proposed by the VQEG in [22]. The result would be an equalization of quality across frames, where quality is measured perceptually instead of only objectively.

B. Video Compression and Coders

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 block 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 GTV coder [4] is based on the Group Testing DCT Image coder [23]. 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 *et al.*[23]. 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.

Recent work investigates bit allocation methods for embedded video coders. Cheng *et al.* [1] 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 [2] 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. Their initial estimation stage inspired the MultiStage algorithm's constant quality stage (described in Section III). In Section IV-A, we experimentally compare MultiStage to their algorithm.

In quantization-based coders, it is not straightforward to precisely control the bit allocation per frame. Instead, the frames are divided into macroblocks and a QP is specified for each macroblock. A lower QP results in higher quality at the expense of more bits. To apply our algorithm to the H.263 coder, we need to implement a block-level bit allocation algorithm that allows coding frames to precise rate and distortion targets, in an R-D optimal or near-optimal fashion. The problem has been well-studied in the literature (e.g., [7], [24]), mainly for the MPEG-2 coder [25]. Most use either the Lagrange multiplier method [7], [26], dynamic programming [24], or gradient descent [12]. However, all of those assume that macroblocks are coded *independently* of each other. While this holds for MPEG-2, it does not for H.263, where the QPs of consecutive macroblocks may not differ by more than 2.

This dependency led us to cast the block-level bit allocation problem as a variation on the multiple choice knapsack problem (MCKS) (see e.g., [27]), which takes into account the limit on QP variation [5]. We then solved the problem using dynamic programming. For H.264, the QPs can be unrestricted, so we can use a more efficient dynamic programming setup to achieve good rate distortion with constant quality.

The current research implementation of H.264 used in the JM reference software includes a rate-control algorithm that also performs R-D optimization. The algorithm is optimized for average distortion and uses a second-order rate-distortion model [28], [29]. A pre- and post-encoding stage are necessary to ensure bit targets are met. In Section IV-C, we use JM 10.2 as a benchmark to compare against MultiStage.

```
MultiStage($B$, $\delta$, $\epsilon$)
  1) Target rate stage: code each frame to rate $B/n$ ($n =$ number of frames)
  2) Let $p_i$ be the PSNR achieved for frame $i$ in step 1
  3) While $\max_i(p_i) - \min_i(p_i) > \delta$:
      a) set $P \leftarrow \frac{1}{n}\sum_{i=1}^{n} p_i$ (average PSNR)
      b) Constant quality stage: code all frames to PSNR $P$
      c) Let $r_i$ be the bit rate achieved for frame $i$ in step 3b, and $R = \sum_{i=1}^{n} r_i$
      d) If $|R - B|/B < \epsilon$, terminate
      e) Target rate stage: code each frame $i$ to rate $(B/R) \cdot r_i$
      f) Let $p_i$ be the PSNR achieved for frame $i$ in step 3e
```

Fig. 2. Our version of the MultiStage algorithm. The parameters are B: total bit budget, δ: target PSNR range, and ϵ: target rate range.

III. MultiStage Algorithm

The MultiStage algorithm [30], [31] is so named because it alternates between two distinct stages—the *constant quality* stage and the *target rate* stage. In the constant quality stage, all frames are coded to the same distortion. In the target rate stage, the sequence is coded to the exact total bit budget. There are two possible termination conditions: either the distortion after the rate stage is close to constant or the rate after the distortion stage is close to the target rate. The algorithm, slightly modified from the one presented in [30], is shown in Fig. 2.

The target for each constant quality stage is the average PSNR from the previous target rate stage. Let r_i be the rate achieved for frame i in the last constant quality stage, and $R = \sum_i r_i$ the total rate. If B is the total target rate, then the target rate stage assigns to the ith frame $r_i(B/R)$ bits (which would work perfectly if R-D curves were linear). Note that if MultiStage terminates at step 3c, the result is a constant-quality encoding. Otherwise, the rate target will be met precisely but there will be a slight variation in the quality.

Although originally designed for embedded coders, the only explicit requirement MultiStage has from the underlying coder is that it allow coding a frame to precisely the required rate or distortion. In fact, almost any coder can do that, including H.263 and the new H.264 standard [32]–[34] with the help of an appropriate single-frame bit allocation scheme.

This algorithm is not optimized for speed; indeed, since MultiStage consists of multiple passes of compression, it will always be slower than one pass methods. As noted in the Introduction, there are many offline applications that require excellent quality and do not need efficient encoding time.

A. Buffer Control

Note that if we are in a situation where buffer underflow is a concern, the MultiStage algorithm does not guarantee that the buffer underflow avoidance constraint will be satisfied. Recall from Section II-A 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 is given by (1). 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

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

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

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

B. Applying MultiStage to Video Coders

We first apply MultiStage to an embedded video coder, GTV. Embedded video coders allow the bit allocation algorithm to specify an exact size for each frame. Thus, the implementation of MultiStage is straightforward.

It is more challenging to apply MultiStage to the quantization-based coders, especially H.263. While many authors have addressed block-level bit allocation, most have worked with MPEG-1/2 coders. While H.263 is very similar to MPEG-2 in the way frames are coded, there are a few differences that affect the way bit allocation works. One such difference is the way H.263 codes QPs.

Unlike MPEG-2, H.263 uses differential coding for QPs. The QP of the first macroblock is coded first as a full 5-bit value, but later QPs are coded by their difference (delta) from the previous block (in raster order). Furthermore, the magnitude of this difference (called DQUANT) may not exceed 2 (i.e., $\mathrm{DQUANT} \leq 2$), and it is somewhat cheaper (by 2 bits) to code a DQUANT of zero than nonzero.

Unfortunately, this renders most bit allocation algorithms in the literature inaccurate, since they assume that QPs may be picked in any combination. This includes the Shoham–Gersho method [7] and its model-based derivatives, as well as gradient-descent. The method of Wiegand *et al.* [24] can handle this constraint, but they chose not to implement this. Ribas–Corbera and Lei [26] choose a QP for each block in raster order, and then force it into the valid range. Their method is now implemented as TMN8 in the H.263 standard.

We based our solution on a different approach. As Mohr observed in [5], block-level bit allocation can be viewed as an instance of the MCKS. This is a well-known generalization of the classic knapsack problem (see e.g., [27] ch. 11): given N boxes, each containing K items, each with integer value and cost, and an integer C, pick exactly one item from each box to maximize total value, while keeping total cost under C.

If blocks are coded independently, there is an obvious mapping from bit allocation to MCKS: each macroblock is a box, and each QP setting is an item, whose cost is the bit-rate and whose value is the negated distortion, measured as the total squared error. The cost limit C is set to the total bit budget. An optimal choice of items will correspond to an optimal choice of QPs. The problem can be solved using dynamic programming. We define $V(i, c)$ to be the optimal value achievable using boxes $1, \ldots, i$ with total cost *exactly* c. That yields the following recurrence:

$$V(i, c) = \max_k \left(V(i-1, c - c_{ik}) + v_{ik}\right) \tag{2}$$

where v_{ik} and c_{ik} are the value and cost of item k in box i, respectively. The global optimal solution will have value $opt = \max_c V(N, c)$. Once we know *opt*, we can trace back in the usual way to reconstruct the optimal set of choices. The time complexity for this algorithm is $O(NKC)$.

Of course, in the case of H.263, we must find a way to preserve the DQUANT constraint. We begin by adding that constraint to MCKS, in the form of an additional parameter Δ, which is the maximum difference between choices from consecutive boxes; we call this new problem Δ-*constrained MCKS*. To solve it, we need to modify our goal function somewhat. We now define $V(i, c, k)$ to be the optimal value achievable using boxes $1, \ldots, i$, with total cost c, *and* choosing item k from box i. This yields the following recurrence:

$$V(i, c, k) = \max_{|\delta| \leq \Delta} \left(V(i-1, c - c_{ik}, k - \delta) + v_{ik}\right). \tag{3}$$

The value of DQUANT is represented by Δ. This recurrence can be computed for all pertinent values in time $O(NKC\Delta)$, and the solution reconstructed, once again, by tracing back through the computed table. In our case, of course, $\Delta = 2$.

Finally, to reduce the complexity, we introduced a simple heuristic. In practice, the different QPs in a single frame tend to be close—within a range of 6–8 values. Taking advantage of this, we restricted the QP search to a narrow range. Given a target rate R, let Q_l be the highest QP for which coding all blocks to Q_l will achieve rate higher than R, and let $Q_h = Q_l + 1$. We find Q_l and Q_h by search, and then restrict the search to the range $[Q_l - D, Q_h + D]$, where D is a user-specified parameter. If that range exceeds the range of valid QPs, we use the appropriate (highest or lowest) $2D$-value range containing Q_l and Q_h. We did not apply this restriction to the initial I-frame, where QPs tend to vary more.

Typically, consecutive frames will have similar R-D characteristics, and so we start our search for Q_l with the value found in the previous frame. In our experiments, only rarely (as in scene changes) did the value change by more than 3. The heuristic then gains on two fronts: first, by not having to generate the entire R-D curves of every block; and second, by reducing the size of the Δ-constrained MCKS problem that has to be solved. Our experiments show that for a range of 8 QPs, the sacrifice in optimality was negligible (usually 0, and never more than

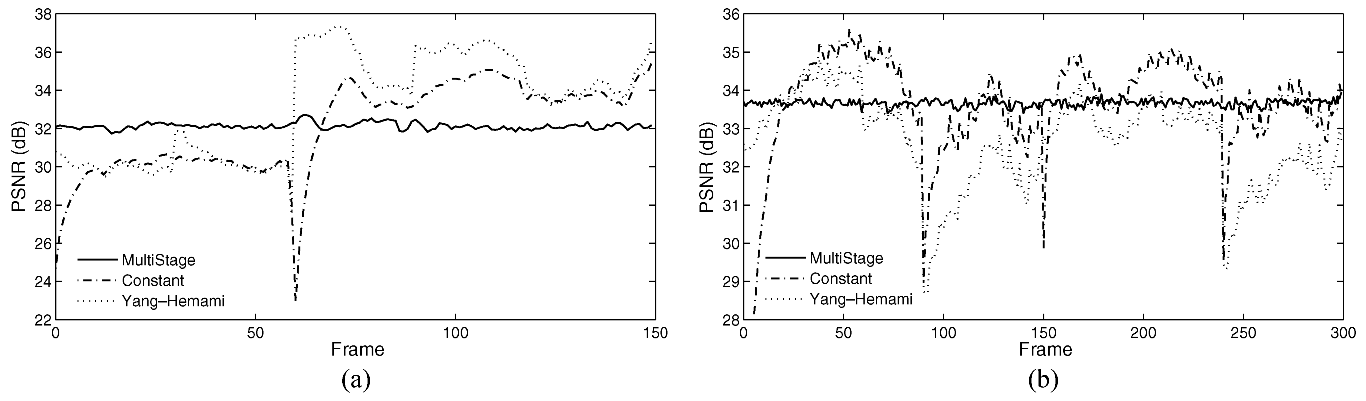


Fig. 3. Comparison of the MultiStage, Constant Rate, and Yang–Hemami algorithms for the videos Trevor and News, using the embedded coder. (a) Trevor. (b) News.

0.1 dB), and the speedup gain was more than 3-fold compared to the full-range optimal solution.

With the emergence of the new H.264 standard [35], it is natural to study how MultiStage may be applied to it. H.264 does not have the same DQUANT requirement as H.263, so we are free to simply run an unconstrained version of the block-level bit allocation. Thus, our dynamic program is given by (2). The values returned by the dynamic program will be the optimal QPs for each macroblock in the frame. However, the original reason for the DQUANT constraint was to save bits in the header of the frame. We implemented both the Δ-constrained version of the dynamic program (3) with $\Delta = 2$ and the simpler unconstrained version, and found that they produced essentially the same PSNR given the bit rate, with the average slightly higher in the Δ-constrained version. Thus, it appears that constraining the change in QPs between macroblocks in H.264 does not significantly improve PSNR.

Note that the JM implementation of H.264 has its own rate control algorithm that allows for R-D optimization. However, it is somewhat strict in that it does not allow the bit rate to vary much between frames. As the next section shows, we perform much better than JM under the MINMAX criterion, while still maintaining a constant bit rate.

IV. Results

In this section, we present our results for MultiStage on the embedded coder, GTV, H.263, and H.264. We note that MultiStage is a multiple pass algorithm and as such is much slower than one pass methods. This is not a concern of ours, because MultiStage is designed for offline applications where quality is much more important than speed. MultiStage usually converges in 3–5 iterations.

A. MultiStage With an Embedded Coder

For the embedded coder, all tests had a channel rate of 100 kbps and a frame rate of 30 fps. We used the GTV coder [4] 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 Fig. 3 we compare the results of running the MultiStage algorithm with the results of Yang and Hemami's algorithm and

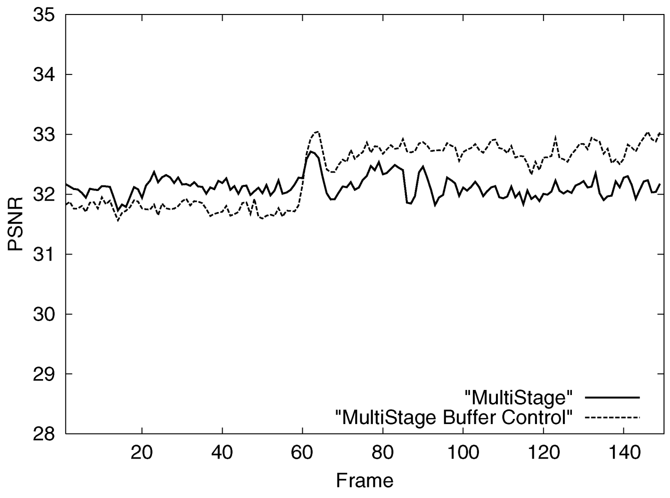


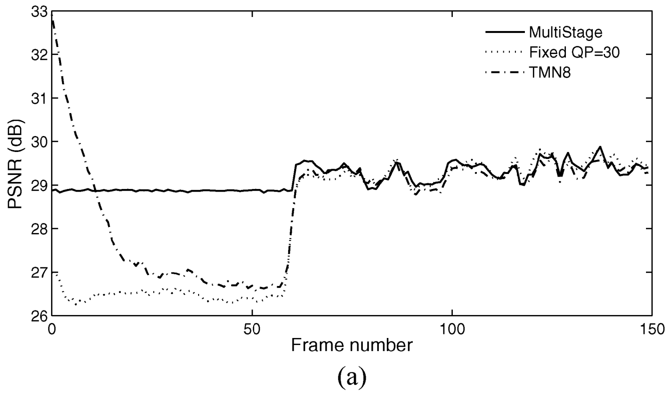
Fig. 4. Effect of adding buffer control to the MultiStage algorithm on Trevor.

TABLE I
Minimum PSNR, Average PSNR, PSNR Variance, and Delay, for Three Embedded Coder Methods at 100 kbps

Video	Method	PSNR			Buffer Delay
		Min	Avg	Variance	
News	MultiStage	33.40	33.66	0.01	0.22 sec
10 sec	Yang-Hemami	28.67	32.62	1.48	GOP ≈ 1 sec
	Constant	22.97	33.52	2.58	0
Trevor	MultiStage	31.73	32.13	0.03	1.22 sec
5 sec	Yang-Hemami	27.54	33.21	7.51	GOP ≈ 1 sec
	Constant	22.93	32.00	5.69	0

constant rate allocation. We chose a GOP size of 30 frames for Yang and Hemami's algorithm, as in [2]. In both experiments, the MultiStage algorithm results in the most consistent PSNR and the highest minimum PSNR. The MultiStage algorithm also avoids the PSNR dips at the scene changes.

In Fig. 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



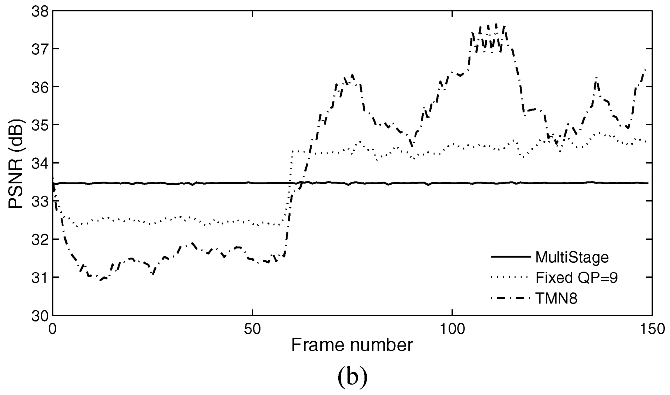
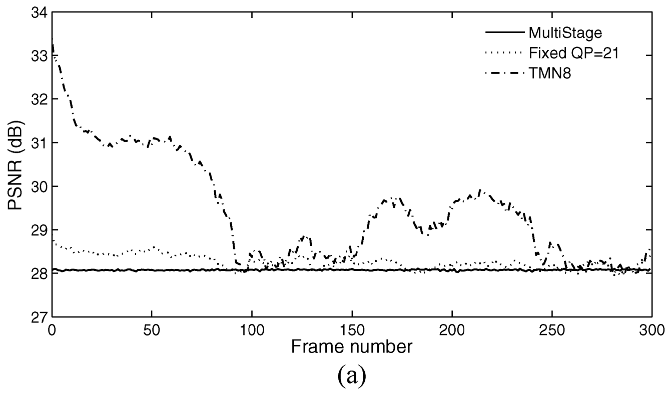


Fig. 5. Comparison of frame-by-frame PSNR between fixed QP, MultiStage, and TMN8, for Trevor at bit rates 25 kbps and 100 kbps. (a) 25 kbps. (b) 100 kbps.



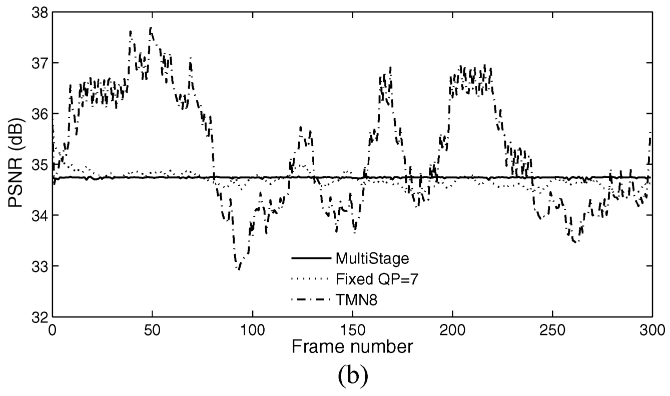


Fig. 6. Comparison of frame-by-frame PSNR between fixed QP, MultiStage, and TMN8, for News at bit rates 25 kbps and 100 kbps. (a) 25 kbps. (b) 100 kbps.

degraded and the quality after that frame is slightly improved. These differences, however, are small. In some of the other experiments we ran (described in [36]), 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.

Table I summarizes our results. MultiStage has the highest minimum PSNR and the lowest PSNR variance of the three methods, at a cost of some delay. For the Yang–Hemami method, one GOP (1 s in our experiments) is always sufficient.

B. MultiStage in H.263

Measuring the success of our block-level allocation algorithm against H.263 is challenging. On the one hand, the default H.263 block-level bit allocation is very unreliable and can miss the target by a large margin. On the other hand, most algorithms published in the literature do not maintain the DQUANT constraint, and hence are not comparable to ours. We therefore compare our algorithm to the fixed-QP allocation, where all macroblocks are quantized with the same QP.

We compared the performance of MultiStage against the default TMN8 rate control algorithm (based on [26]) implemented in the Telenor H.263 coder [37], and against fixed-QP coding. (The online TMN5 algorithm implemented in the coder is not comparable, since it attempts to maintain strict buffer constraints, which our algorithm does not.) We choose a target rate and code with MultiStage and TMN8 to that target. The block-level MCKS optimizer was set to use an 8-QP range. In order to hit the target with fixed QP, we code the video at a fixed QP over the entire range of possible QPs. We then pick the QP that is closest to our target rate.

Figs. 5 and 6 compares frame-by-frame PSNR for MultiStage, TMN8, and fixed QP. Results on more sequences and bit rates can be found in [31]. Clearly, MultiStage achieves very high stability, typically coding all frames to within 0.3–0.4 dB of each other, while the sequences coded with TMN8 exhibit very wide fluctuations. Coding with a fixed QP works very well within a scene, but different scenes can have very different R-D characteristics, as can be seen in the Trevor graphs (Fig. 5) around frame 50. Since MultiStage is a global algorithm, it can equalize the quality of the two scenes.

C. MultiStage in H.264

We measure the success of MultiStage against the default rate control included in the JM 10.2 implementation of H.264, and against a fixed QP version. We use the Δ-constrained version of MultiStage, since it gives a slight advantage in PSNR.

Fig. 7 compares frame-by-frame PSNR for MultiStage, JM 10.2, and fixed QP on 5 s of the video sequence Trevor. Two different bit rates are shown. Again, MultiStage achieves very high stability, typically coding all frames to within 0.1 dB of

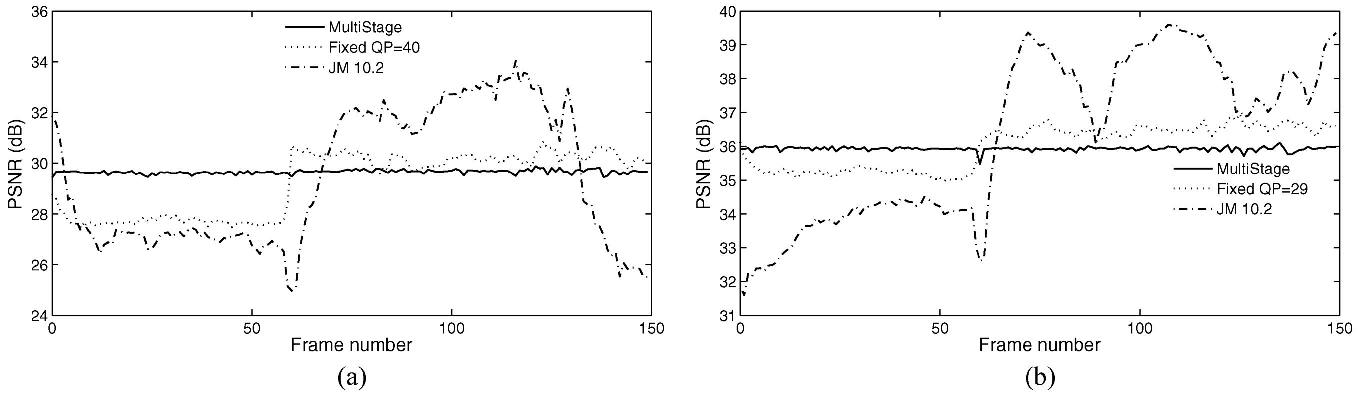


Fig. 7. Comparison of frame-by-frame PSNR between MultiStage, fixed QP, and JM 10.2, for the sequence Trevor at bit rates 25 kbps and 100 kbps. (a) 25 kbps; (b) 100 kbps.

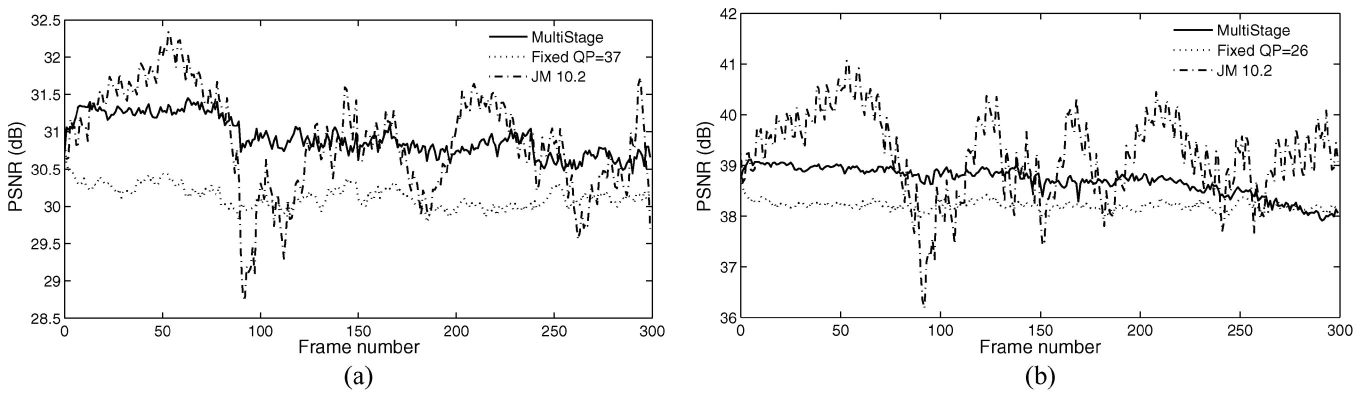


Fig. 8. Comparison of frame-by-frame PSNR between MultiStage, fixed QP, and JM 10.2, for the sequence News at bit rates 25 kbps and 100 kbps. (a) 25 kbps. (b) 100 kbps.

TABLE II
VIDEO SEQUENCE AKIYO, 10 S, AT FOUR DIFFERENT BIT RATES

Target rate	Method	Rate (% error)	PSNR			Buffer delay (s)
			Min	Avg	Var	
25 kbps	MultiStage	27.4 (9.4%)	38.8	38.9	0.00	1.32
	Fixed QP=28	24.7 (-1.1%)	38.2	38.5	0.02	0.73
	JM 10.2	25.0 (0.1%)	37.4	39.0	0.68	0.73
50 kbps	MultiStage	51.0 (2.0%)	41.5	42.0	0.05	1.15
	Fixed QP=24	44.3 (-11.4%)	41.0	41.3	0.01	0.49
	JM 10.2	50.0 (0.0%)	40.6	42.4	0.81	0.47
75 kbps	MultiStage	75.4 (0.5%)	43.4	43.9	0.05	1.00
	Fixed QP=21	69.7 (-7.1%)	43.2	43.5	0.01	0.41
	JM 10.2	75.0 (0.0%)	42.3	44.4	0.83	0.40
100 kbps	MultiStage	100.9 (0.9%)	45.3	45.4	0.00	0.78
	Fixed QP=19	94.6 (-5.4%)	44.8	45.1	0.01	0.41
	JM 10.2	100.0 (0.0%)	43.7	45.8	0.79	0.35

TABLE III
VIDEO SEQUENCE FOREMAN, 10 S, AT FOUR DIFFERENT BIT RATES

Target rate	Method	Rate (% error)	PSNR			Buffer delay(s)
			Min	Avg	Var	
25 kbps	MultiStage	32.6 (30.3%)	28.0	28.4	0.01	3.58
	Fixed QP=42	24.3 (-3.0%)	25.3	26.9	0.80	0.94
	JM 10.2	25.1 (0.3%)	23.5	27.7	3.03	0.64
50 kbps	MultiStage	54.8 (9.5%)	31.1	31.5	0.01	1.17
	Fixed QP=36	46.9 (-6.3%)	28.9	30.7	0.79	0.71
	JM 10.2	50.0 (0.1%)	27.1	31.5	3.24	0.19
75 kbps	MultiStage	77.8 (3.7%)	33.1	33.5	0.01	1.17
	Fixed QP=32	74.3 (-0.9%)	31.6	33.3	0.85	1.15
	JM 10.2	75.0 (0.0%)	29.5	33.8	2.91	0.17
100 kbps	MultiStage	101.7 (1.74%)	34.7	35.0	0.00	0.91
	Fixed QP=30	96.4 (-3.7%)	33.1	34.7	0.74	0.85
	JM 10.2	100.0 (0.0%)	31.7	35.3	2.72	0.14

each other, while the sequences coded with JM 10.2 vary widely. Fixed QP achieves similar results to those for H.263. Fig. 8 shows the results for 10 s of the video sequence News. The performance of MultiStage is even more striking on this sequence; the curve for quality at 100 kbps is almost a straight line. Fixed QP also shows a very flat curve, but the average PSNR is lower, because this method cannot hit the target rate precisely and thus wastes some bits.

Tables II–V give more information about the performance of MultiStage. MultiStage is compared with JM 10.2 and fixed QP on four different video sequences at four different rates. One advantage of the embedded coders over the quantization-based coders is the ability to precisely control rate. Applying MultiStage to H.264 results in higher error rates than with the embedded coder, particularly at low bit rates. Compared with JM 10.2 and fixed QP, the minimum PSNR is always highest with MultiStage and the variance is almost always lowest. Note also that the average PSNR is comparable with JM 10.2. The delay, calculated according to the method in Section III-A, is lowest with JM 10.2. This is because JM 10.2 devotes less total bits in the beginning of the sequence than either of the other methods (so it may be flexible later on). The effect is most striking in Foreman, which has the most variance overall due to scene changes.

TABLE IV
VIDEO SEQUENCE NEWS, 10 s, AT FOUR DIFFERENT BIT RATES

Target rate	Method	Rate (% error)	PSNR			Buffer
			Min	Avg	Var	delay(s)
25 kbps	MultiStage	27.9(11.5%)	30.6	30.9	0.00	1.17
	Fixed QP=37	24.8 (-0.7%)	29.8	30.1	0.02	0.47
	JM 10.2	25.2 (0.8%)	28.7	30.9	0.47	0.48
50 kbps	MultiStage	51.4 (2.9%)	33.9	34.7	0.10	0.70
	Fixed QP=32	44.8 (-10.4%)	33.3	33.5	0.01	0.36
	JM 10.2	50.1 (0.3%)	32.4	34.9	0.56	0.35
75 kbps	MultiStage	75.7 (1.0%)	36.3	37.0	0.11	0.55
	Fixed QP=29	65.5 (-12.7%)	35.6	35.9	0.01	0.30
	JM 10.2	75.2 (0.3%)	34.9	37.3	0.62	0.30
100 kbps	MultiStage	100.4 (0.4%)	38.6	38.7	0.00	0.29
	Fixed QP=26	95.2 (-4.8%)	38.0	38.2	0.01	0.45
	JM 10.2	100.2 (0.2%)	36.2	39.2	0.69	0.29

TABLE V
VIDEO SEQUENCE TREVOR, 5 s, AT FOUR DIFFERENT BIT RATES

Target rate	Method	Rate (% error)	PSNR			Buffer
			Min	Avg	Var	delay(s)
25 kbps	MultiStage	30.4 (21.7%)	29.5	29.6	0.00	2.31
	Fixed QP=40	24.6 (-1.7%)	27.5	29.2	1.55	0.95
	JM 10.2	25.5 (2.1%)	25.0	29.4	7.31	0.60
50 kbps	MultiStage	53.8 (7.7%)	32.1	32.6	0.01	1.61
	Fixed QP=35	45.8 (-8.3%)	30.8	32.2	0.95	0.70
	JM 10.2	50.2 (0.4%)	28.0	32.7	5.26	0.25
75 kbps	MultiStage	76.4 (1.8%)	34.0	34.4	0.01	1.29
	Fixed QP=32	67.0 (-10.7%)	32.7	33.9	0.57	0.59
	JM 10.2	75.2 (0.3%)	31.1	34.8	5.59	0.15
100 kbps	MultiStage	100.6 (0.6%)	35.5	35.9	0.00	1.17
	Fixed QP=29	99.6 (-0.4%)	35.0	36.0	0.41	0.84
	JM 10.2	99.8 (0.2%)	31.6	36.3	5.88	0.10

V. CONCLUSIONS AND FUTURE WORK

As our results demonstrate, the MultiStage algorithm is an effective approach for near-constant-quality coding of quantization-based video. Clearly, it can be applied only for offline applications (e.g., DVD coding). On the other hand, it achieves *global stability* with time complexity essentially linear in the sequence length. The only other algorithm we know of that does that is Schuster *et al.*'s bisection method from [11], but MultiStage seems to converge faster, especially at medium to high rates.

We have also introduced a novel approach for optimal block-level rate allocation for H.263, which maintains the DQUANT constraint. Our solution, based on a dynamic programming solution to a modified multiple-choice knapsack problem, is guaranteed to be optimal, as long as the rate and distortion of each block depend only on its QP. Our algorithm does not rely on the quantizers being convex or even monotonic (and, in fact, in H.263 they are not always so).

There are many interesting directions for possible future work on the algorithms presented here. Some of the more important ones include the following.

Buffer management: The current work does not address the issue of channel bandwidth and buffer constraints. It is not clear how to interpret the MINMAX criterion in the presence of such constraints. Lee and Ortega claim in [16], that once buffer constraints become tight for some portion of the sequence coded to MINMAX distortion, the question becomes how to allocate bits to the rest of the sequence. However, this is not always the case; a constant-quality allocation of bits for the tight portion might not be optimal globally.

Block mode selection: H.263 allows macroblocks in P-frames to be coded either in either P (predicted) or I (intra) mode. In our implementation we forced all blocks to be encoded as P blocks, which is suboptimal. Wiegand *et al.* showed in [24] how mode selection can be performed, and it seems that the MCKS framework should be able to support that task.

REFERENCES

[1] P. Cheng, J. Li, and C.-C. Kuo, "Rate control for an embedded wavelet video coder," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 4, pp. 696–702, Aug. 1997.

[2] Y. Yang and S. S. Hemami, "Minmax frame rate control using a rate-distortion optimized wavelet coder," in *IEEE Int. Conf. Image Process.*, Oct. 1999, pp. 551–555.

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

[4] G. Shavit, M. F. Ringenburg, J. West, R. E. Ladner, and E. A. Riskin, "Group testing for video compression," in *Proc. IEEE DCC*, Mar. 2004, pp. 212–221.

[5] A. E. Mohr, "Bit allocation in sublinear time and the multiple choice knapsack problem," in *Proc. IEEE DCC*, Mar. 2002, pp. 352–361.

[6] K. Ramchandran and M. Vetterli, "Best wavelet packet bases in a rate-distortion sense," *IEEE Trans. Image Process.*, vol. 2, no. 2, pp. 160–175, Jun. 1993.

[7] Y. Shoham and A. Gersho, "Efficient bit allocation for an arbitrary set of quantizers," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 36, no. 9, pp. 1445–1453, Sep. 1988.

[8] P. H. Westerink, J. Biemond, and D. E. Boekee, "An optimal bit allocation algorithm for subband coding," in *Proc. ICASSP*, 1988, pp. 757–760.

[9] M. Effros and P. A. Chou, "Weighted universal bit allocation: optimal multiple quantization matrix coding," in *Proc. ICASSP*, 1995, pp. 2343–2346.

[10] K. Ramchandran and A. Ortega, "Bit allocation for dependent quantization with applications to multiresolution and MPEG video coders," *IEEE Trans. Image Process.*, vol. 3, no. 9, pp. 533–545, Sep. 1994.

[11] G. M. Schuster, G. Melnikov, and A. K. Kastaggelos, "A review of the minimum maximum criterion for optimal bit allocation among dependent quantizers," *IEEE Trans. Multimedia*, vol. 1, no. 1, pp. 3–17, Mar. 1999.

[12] Y. Sermadevi and S. S. Hemami, "Efficient bit allocation for dependent video coding," in *Proc. IEEE DCC*, Mar. 2004, pp. 232–241.

[13] J. Ribas-Corbera and S. M. Lei, "A frame-layer bit allocation for H.263," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 7, pp. 1154–1158, Oct. 2000.

[14] D. W. Lin, M.-H. Wang, and J.-J. Chen, "Optimal delayed-coding of video sequences subject to a buffer-size constraint," in *Proc. SPIE Visual Commun. Image Process.*, Nov. 1993, pp. 223–234.

[15] K. Wang and J. W. Woods, "MPEG motion picture coding with long-term constraint on distortion variation," in *Proc. SPIE: Image Video Commun. Process.*, Mar. 2005, pp. 284–296.

[16] S.-Y. Lee and A. Ortega, "Optimal rate control for video transmission over VBR channels based on a hybrid MMAX/MMSE criterion," in *Proc. IEEE ICME*, Aug. 2002, pp. 93–96.

[17] B. Wie and W. Zeng, "A sequence-based rate control framework for consistent quality real-time video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 1, pp. 56–71, Jan. 2006.

[18] Z. Chen and K. N. Ngan, "Distortion variation minimization in real-time video coding," *Signal Process. Image Commun.*, vol. 21, no. 4, pp. 273–279, Apr. 2006.

[19] D. T. Hoang, J. S. Vitter, and E. L. Linzer, "Lexicographic bit allocation for MPEG video coding," in *Proc. IEEE DCC*, Mar. 1997, pp. 101–110.

[20] J. Lin and A. Ortega, "Bit-rate control using piecewise approximated rate-distortion characteristics," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 4, pp. 446–459, Aug. 1998.

[21] Y. Yang and S. S. Hemami, "Rate control for VBR video over ATM: simplification and implementation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 9, pp. 1045–1058, Sep. 2001.

[22] P. J. Corriveau, A. A. Webster, A. M. Rohaly, and J. M. Libert, "Video quality experts group: the quest for valid objective methods," in *Proc. SPIE*, Jun. 2000, vol. 3959, pp. 129–139.

[23] E. Hong, R. E. Ladner, and E. A. Riskin, "Group testing for image compression using alternative transforms," *Signal Process.: Image Commun.*, vol. 18, no. 7, pp. 561–574, Aug. 2003.
[24] 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 Trans. Circuits Syst. Video Technol.*, vol. 6, no. 2, pp. 182–190, Apr. 1996.
[25] *The MPEG-2 Int. Standard*, ISO/IEC 13818-2, 1996.
[26] J. Ribas-Corbera and S.-M. Lei, "Rate control in DCT video coding for low-delay communications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 1, pp. 172–185, Feb. 1999.
[27] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. New York: Springer-Verlag, 2004.
[28] H. J. Lee, T. H. Chiang, and Y. Q. Zhang, "Scalable rate control for MPEG-4 video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 6, pp. 878–894, Sep. 2000.
[29] A. Vetro, H. Sun, and Y. Wang, "MPEG-4 rate control for multiple video objects," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 1, pp. 186–199, Feb. 1999.
[30] M. F. Ringenburg, R. E. Ladner, and E. A. Riskin, "Global MINMAX interframe bit allocation for embedded video coding," in *Proc. IEEE DCC*, Mar. 2004, pp. 222–231.
[31] G. Shavit, R. E. Ladner, and E. A. Riskin, "MINMAX bit allocation for quantization-based video coders," in *Proc. IEEE DCC*, Mar. 2005, pp. 299–308.
[32] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.
[33] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro, "H.264/AVC baseline profile decoder complexity analysis," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 704–716, Jul. 2003.
[34] A. K. Luthra, G. J. Sullivan, and T. Wiegand, "Introduction to the special issue on the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 557–558, Jul. 2003.
[35] *Draft ITU Rec. Final Draft Int. Standard of Joint Video Specification*, ITU Rec. H.264/ISO/IEC 14 496-10 AVC, Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, JVT-G050, 2003.
[36] M. F. Ringenburg, Global MINMAX interframe bit allocation for embedded video coding Oct. 2003 [Online]. Available: http://www.cs.washington.edu/homes/miker/bitalloc-quals.pdf
[37] Signal Process. and Multimedia Group, Univ. British Colombia, TMN H.263+ encoder/decoder, ver. 3.0 Sep. 1997 [Online]. Available: http://spmg.ece.ubc.ca

Neva Cherniavsky received the B.S. degree from Tufts University, Medford, MA, in 2001 and the M.S. degree in computer science from the University of Washington, Seattle, in 2004, where she is currently working toward her Ph.D. degree.

Her research interests include video compression, assistive technology, machine learning, and auction theory.

Gidon Shavit graduated from Tel Aviv University, Tel Aviv, Israel, and received the M.S. degree in computer science from the University of Washington, Seattle, where he wrote his thesis on the use of alternative transforms and motion-compensation in bit plane coding video compression.

He is currently with Medio Systems, Inc., Seattle, WA.

Michael F. Ringenburg received the B.A. degree in mathematics in 1999 and the M.S. degree in computer science in 2001, both from Dartmouth College, Hanover, NH. He is currently a graduate student in the Department of Computer Science and Engineering, University of Washington, where he has conducted research into video compression, software security, and programming languages.

He is currently on leave from the University of Washington, and is working as a software engineer for Cray, Inc., Seattle, WA.

Richard E. Ladner (M'87–SM'91) received the B.S. degree from St. Mary's College of California, in 1965 and the Ph.D. in mathematics from the University of California, Berkeley, in 1971.

He is currently the Boeing Professor in Computer Science and Engineering, University of Washington, Seattle. In addition to his appointment in the Department of Computer Science and Engineering (CSE), he is an Adjunct Professor in the Department of Electrical Engineering and in the Department of Linguistics. He has a number of research interests, most of them in theoretical computer science. He is currently investigating design and analysis of algorithms, cache performance of algorithms, network algorithms for media-on-demand, data compression algorithms, and technology for disabled persons. He has continuing interests in automata based computational complexity theory and distributed computing. He has supervised or co-supervised 17 students on their Ph.D. dissertations and six on their M.S. theses. He has supervised numerous undergraduate research projects and coordinates the Undergraduate Research Seminar in CSE. Since 1994, as part of the DO-IT Project, he has held a one week summer workshop for disabled high school students encouraging them to pursue college programs and careers in science, mathematics, and engineering. He has served as an Area Editor for the *Journal of the Association of Computing Machinery*, Editor for SIAM *Journal on Computing*, and an Associate Editor of the *Journal of Computer and System Sciences*. He is currently on the Editorial Board for *Theory of Computing Systems.* He has served as Pacific Region Representative on the Council of the Association of Computing Machinery (ACM). He is currently Chair of the ACM Special Interest Group in Algorithms and Computation Theory (SIGACT).

Prof. Ladner was a Guggenheim Fellow in 1985-1986 and a Fulbright Scholar in 1993. He is a Fellow of the ACM. He is a recipient of the 2004 Presidential Award for Excellence in Science, Mathematics, and Engineering Mentoring (PAESMEM).

Eve A. Riskin (M'96–SM'02) received the B.S. degree in electrical engineering from the Massachusstes Institute of Technology, Cambridge, and the M.S. degree in electrical engineering, the M.S. degree in operations research, and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA.

Since 1990, she has been in the Electrical Engineering Department, University of Washington, Seattle, where she is currently Associate Dean of Academic Affairs in the College of Engineering and Professor of Electrical Engineering. Her research interests include image compression and image processing, with a focus on developing video compression algorithms to allow for cell-phone transmission of American Sign Language.

Dr. Riskin was awarded a National Science Foundation Young Investigator Award, a Sloan Research Fellowship, the 2006 WEPAN University Change Agent Award, and the 2006 Hewlett-Packard Harriett B. Rigas Award.