Meeting Arbitrary QoS Constraints Using Dynamic Rate Shaping of Coded Digital Video

Alexandros Eleftheriadis and Dimitris Anastassiou

Department of Electrical Engineering
and Center for Telecommunications Research
Columbia University, New York, NY 10027, USA
{eleft,anastas}@ctr.columbia.edu

Abstract. We introduce the concept of *Dynamic Rate Shaping*, a technique to adapt the rate of compressed video bitstreams (MPEG-1, MPEG-2, H.261, as well as JPEG) to dynamically varying rate (and delay) constraints. The approach provides an interface (or filter) between the encoder and the network, with which the encoder's output can be perfectly matched to the network's quality of service characteristics. Since the presented algorithms do not require interaction with the encoder, they are fully applicable to precoded, stored video (as in, for example, video-on-demand systems). By providing decoupling of the encoder and the network, universal interoperability can be achieved. A set of low-complexity algorithms for dynamic rate shaping is presented, and both optimal and extremely fast designs are discussed. The latter are simple enough to allow software-based implementation. Experimental results are provided using actual MPEG-2 bitstreams.

1 Introduction

In applications of digital video communications there are many cases where control of the bit rate of video is needed, even after encoding has already taken place. One example is video-on-demand services, in which transmission of precoded material may occur over a wide variety of channels; multiresolution coding with too many layers would be undesirable, due to the loss in coding efficiency. Another example is transmission of real-time or precoded video material over channels with limited or no quality of service guarantees (e.g. CSMA/CD LANs). Although techniques have been developed to employ rate control for live sources based on network feedback [4, 6], no solution is currently available for prerecorded material. Similarly, consider a variable bit rate (VBR) video source that is fed to an ATM virtual circuit: due to the difficulties in modeling VBR video traffic, the traffic characterization used for admission control and policing will not necessarily match that of the source. Instead of dropping vital information at the source or in internal network nodes, an operation that would manipulate the bitstream so that it complies with what the network can deliver would be an extremely useful proactive measure against resource exhaustion.

Another environment that could potentially benefit from such post-encoding rate control operation would be multipoint communication with mobile hosts:

since the mobile link is typically of much lower bandwidth than wired ones, by reducing the video rate at the base-to-mobile link, wired participants would still be able to utilize the full bandwidth available to them without being compromised by the presence of wired ones. The same argument holds for heterogeneous (at least in terms of bandwidth) internetworks.

Finally, an environment that continuously grows in importance is that of general purpose computers. Due to the variety of network transport mechanisms that can be employed and the potential use of video for non-communication applications, it is most likely that general-purpose (transport-independent) video codecs will be used. It is desirable, then, to provide a mechanism that can gracefully interface the codec with the particular transport facilities used, if any.

In all the above cases, the common theme is the need to manipulate the coded bitstream so that it complies with the bandwidth availability of the underlying communication resources. In general, this manipulation is performed at the transmitting host, just above the transport layer, and interfaces the coded video bitstream with the transport service.

We refer to this rate manipulation operation as *Dynamic Rate Shaping* (DRS). The term dynamic refers to the possibility that rate constraints are time-varying, while shaping is used instead of rate control to: 1) differentiate with classical encoder rate control in which the variable rate of an entropy-coded bitstream is matched to a fixed channel rate, and 2) to more accurately capture the posterior (with respect to coding) nature of the operation. Note that DRS is quite different from traffic shaping (e.g. in DRS the traffic's average rate can change). Also, DRS can be used in new types of hybrid guaranteed/best-effort services, such as the ones described in [2].

In order for rate shaping to be viable it has to be implementable with reasonable complexity and yield acceptable visual quality. With respect to complexity, the straightforward approach of decoding the video bitstream and recoding it at the target rate would be obviously unacceptable; the delay incurred would also be an important deterrent. Hence algorithms of complexity less than that of a cascaded decoder and encoder will be sought. In terms of quality, it should be noted that recoding does not necessarily yield optimal conversion; in fact, since an optimal encoder (in an operational rate-distortion sense) is impractical due to its complexity, recoding can only serve as an indicator of an acceptable quality range. As will be shown, regular recoding can be quite lacking in terms of quality, with DRS providing significantly superior results.

We present a set of algorithms that solve the problem of dynamic rate shaping for—possibly motion-compensated—block-based transform coders, including MPEG-1, MPEG-2, H.261, and JPEG. After formulating the DRS problem in an operational rate-distortion context, we derive both optimal and fast approximate algorithm. The latter are shown to perform within 0.5 dB of the optimal ones (a typically non-perceptible difference), hence providing a very good tradeoff between algorithmic complexity and visual quality. The complexity of the optimal algorithm is shown to be less than that of an encoder, while for the fast approximations it is shown to be significantly less than that of a decoder's. While the approach is applicable to any motion-compensated block-based transform codec, the MPEG-2 [1] draft international standard is used for all simulation results presented.

The structure of the paper is as follows. In Section 2 we formulate the problems of general and constrained dynamic rate shaping. In Section 3 we discuss optimal and fast approximate algorithms for constrained rate shaping of intra-coded pictures. In Section 4 we generalize the approach to tackle the mixed-mode (I, P, and B) coding case. Finally, in Section ?? we present some concluding remarks.

2 Dynamic Rate Shaping

We define rate shaping as an operation which, given an input video bitstream and a set of rate constraints, produces a video bitstream that complies with these constraints. For our purposes, both bitstreams are assumed to meet the same syntax specification, and we also assume that a—possibly motion-compensated—block-based transform coding scheme is used. This includes both MPEG-1 and MPEG-2, as well as H.261 and so-called "motion" JPEG. If the rate constraints are allowed to vary with time, the operation will be called dynamic rate shaping. Throughout the paper we assume that MPEG-2 is used as the video coding syntax. For the benefit of the non-expert reader, in the following section we briefly review MPEG's main characteristics; an overview can be found in [7], while the actual standard is detailed in [1].

2.1 MPEG-2 Overview

The algorithmic foundation of MPEG is motion-compensated, block-based transform coding (H.261 falls in the same category). Each picture (either frame or field for interlaced sources) is decomposed into a hierarchical structure consisting of blocks, macroblocks, and slices (see Fig. 1). A block is an 8×8 array of pixels, and is the unit for transform coding. A macroblock is an array of 2×2 luminance blocks (the YUV format is used), together with the corresponding blocks of the chrominance components (an additional 2 to 8 blocks depending on the chroma format used). Macroblocks are the units of motion compensation and quantizer selection, as discussed below. A horizontal strip of macroblocks forms a slice, which is the unit for bitstream resynchronization (several recursively computed quantities are reset at the beginning of a slice).

Each block is transformed using the 2-D Discrete Cosine Transform (DCT), and is subsequently quantized. Quantization is the sole source of quality loss in MPEG, and of course a major source of compression efficiency. The quantized coefficients are converted to a one-dimensional string using a zig-zag pattern (Fig. 1) and then run-length encoded. Run-length codes jointly encode the number of consecutive zero DCT coefficients as well as the value of the next non-zero coefficient. The motivation behind this approach is that typical pictures contain

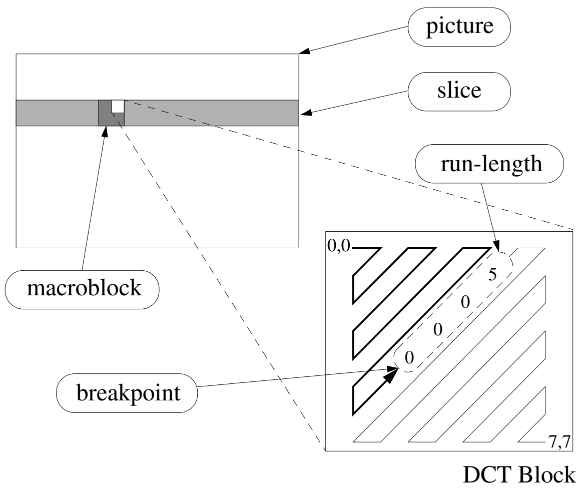

Fig. 1. MPEG picture structure and DCT coefficient zig-zag scanning pattern.

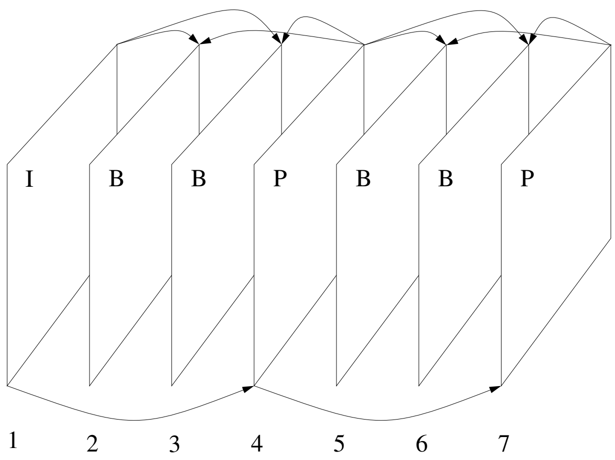

Fig. 2. MPEG sequence structure and motion-compensated prediction reference pictures.

large sequences of zeros in the zig-zag pattern after quantization, and hence run-length coding can very efficiently represent them.

There are three types of pictures in a video sequence: I, P, and B. I or intra pictures are individually coded, and are fully self-contained. P pictures are predicted from the previous I or P picture, while B (or bidirectional) pictures are interpolated from the closest past and future I or P pictures. Fig. 2 shows a typical pattern, including the pictures used as prediction references. Prediction is motion-compensated: the encoder finds the best match of each macroblock in the past or future picture, within a prespecified range. The displacement(s), or motion vector(s), is (are) sent as side information to the decoder.

In order to increase the coding efficiency, MPEG relies heavily on entropy coding. Huffman codes (variable length codewords) are used to represent the various bitstream quantities (run-length codes, motion vectors, etc.). As a result, the output of an MPEG encoder is inherently a variable rate bitstream: the ratio of bits per pixel varies from one block to the next. In order to construct a constant bit rate bitstream (when needed), rate control is used. This is achieved by connecting a buffer to the output of the encoder, that is emptied at a constant rate (the channel rate). The buffer's occupancy is fed back to the encoder, and is used to control the selection of the quantizer for the current macroblock. High buffer occupancy leads to more coarsely quantized coefficients, and hence less bits per block, and vice versa. Through this self-regulation technique one can achieve a constant output rate; clever design is needed in order to avoid buffer overflows and underflows (if required by the channel), and to allocate bits so that the best possible image quality is achieved.

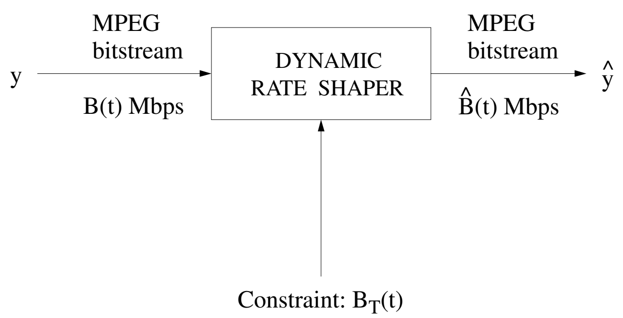

Fig. 3. Operation of a dynamic rate shaper.

2.2 DRS Problem Definition

The rate shaping operation is depicted in Fig. 3. Note that no communication path exists between the rate shaper and the source of the input bitstream, which ensures that no access to the encoder is necessary. Of particular interest is the source of the rate constraints $B_T(t)$. In the simplest of cases, $B_T(t)$ may be just a constant and known a priori, e.g. the bandwidth of a circuit-switched connection. It is also possible that $B_T(t)$ has a well (a priori) known statistical characterization, e.g. a policing function. Finally, another alternative is that $B_T(t)$ is generated by the network over which the output bitstream is transmitted; this could be potentially provided by the network management layer, or may be the result of end-to-end bandwidth availability estimates (as in [4, 6]). The objective of a rate shaping algorithm is to minimize the conversion distortion, i.e.:

$$\min_{\hat{B}(t) \leq B_T(t)} \{||y - \hat{y}||\} \tag{1}$$

Note that no assumption is made on the rate properties of the input bitstream, which can indeed by arbitrary. The attainable rate variation ($\hat{B}/B$) is in practice limited, and depends primarily on the number of B pictures of the bitstream and the original rate $B(t)$.

Assuming that MPEG-2 (or, more generally, a motion-compensated block-based transform coding technique) is used to generate the input bitstream and decode the output one, there are two fundamental ways to reduce the rate: 1) modifying the quantized transform coefficients by employing coarser quantization, and 2) eliminating transform coefficients. In general, both schemes can be used to perform rate shaping; requantization, however, leads to recoding-like algorithms which are not amenable to fast implementation and, as we will see, do not perform as well as selective-transmission ones. Consequently, in the rest of this paper we only consider selective-transmission based algorithms, and more specifically we address the particular case of truncation (a set of DCT coefficients at the end of each block is eliminated). This approach will be referred to as *constrained* dynamic rate shaping.

The number of DCT run-length codes within each block which will be kept will be called the *breakpoint* (Fig. 1). Assuming use of MPEG, and to avoid certain syntax complications [1], we require that at least one DCT coefficient will remain in each block. Consequently, breakpoint values will range from 1 to 64.

3 Rate Shaping of Intra-Coded Pictures

In intra-picture rate shaping, there is no temporal dependence between pictures. Consequently, the shaping error will simply consist of the DCT coefficients that are dropped. It can then be shown that the DRS problem can be expressed as follows:

$$\min_{\hat{B}(t) \leq B_T(t)} \{||y - \hat{y}||\} \Longleftrightarrow \min_{\sum_{i=1}^{N} R_i(B_i) \leq B_T(t)} \left\{ \sum_{i=1}^{N} D_i(b_i) \right\} \tag{2}$$

with

$$D_i(b_i) \equiv \sum_{k \geq b_i} [E^i(k)]^2 \tag{3}$$

[1] These include recoding the coded block patterns, and reexecuting DC prediction loops.

where $b_i \in \{1, \ldots, 64\}$ is the breakpoint value for block i (run-length codes from b_i and up will be eliminated), N is the number of blocks considered, $E^i(k)$ is the value of the DCT coefficient of the k-th run in the i-th block, and $R_i(b_i)$ denotes the rate required for coding block i using a breakpoint value of b_i.

This constrained minimization problem can be converted to an unconstrained one using Lagrange multipliers: instead of minimizing $\sum_i D_i(b_i)$ given $\sum_i R_i(b_i)$, we minimize:

$$\min \left\{ \sum_{i=1}^{N} D_i(b_i) + \lambda \sum_{i=1}^{N} R_i(b_i) \right\} \tag{4}$$

Note that the two problems are not equivalent; for some value of λ, however, which our algorithm will have to find, their solutions become identical [8].

The unconstrained minimization problem can be solved using an iterative bisection algorithm (on λ), which at each step k separately minimizes $D_i(b_i) + \lambda R_i(b_i)$ for each block. A similar algorithmic approach but in a different context has been used in [3, 5, 8]. A short description of the complete algorithm is as follows. We denote by $R_i^*(\lambda)$ and $D_i^*(\lambda)$ the optimal rate and distortion respectively for block i for that particular λ (i.e. they minimize $D_i + \lambda R_i$). We also denote by $b_i^*(\lambda)$ the breakpoint value that achieves this optimum.

Lagrangian Optimization Algorithm

Step 1: Initialization
Set $\lambda_l = 0$ and $\lambda_u = \infty$. If the inequality:

$$\sum_{i=1}^{N} R_i^*(\lambda_u) \leq R_{\text{budget}} \leq \sum_{i=1}^{N} R_i^*(\lambda_l) \tag{5}$$

holds as an equality for either side, an exact solution has been found. If the above does not hold at all, then the problem is infeasible (this can happen if the target rate $\hat{B}$ is too small). Otherwise go to Step 2. Note that these two initial λ's correspond to the minimum and maximum possible breakpoint values (the former minimizes distortion, while the latter minimizes the rate).

Step 2: Bisection and Pruning
Compute:

$$\lambda_{\text{next}} := \left| \frac{\sum_{i=1}^{N} [D_i^*(\lambda_u) - D_i^*(\lambda_l)]}{\sum_{i=1}^{N} [R_i^*(\lambda_u) - R_i^*(\lambda_l)]} \right| \tag{6}$$

and find $R_i^*(\lambda_{\text{next}})$ and $D_i^*(\lambda_{\text{next}})$ such that $b_i^*(\lambda_u) \leq b_i^*(\lambda_{\text{next}}) \leq b_i^*(\lambda_l)$.

Step 3: Convergence Test
If

$$\sum_{i=1}^{N} R_i^*(\lambda_{\text{next}}) = \sum_{i=1}^{N} R_i^*(\lambda_u) \quad \text{or} \quad \sum_{i=1}^{N} R_i^*(\lambda_{\text{next}}) = \sum_{i=1}^{N} R_i^*(\lambda_l) \tag{7}$$

then stop; the solution is $b_i^*(\lambda_u)$, $i = 1, \ldots, N$. If

$$\sum_{i=1}^{N} R_i^*(\lambda_{\text{next}}) > R_{\text{budget}} \tag{8}$$

then $\lambda_l := \lambda_{\text{next}}$, else $\lambda_u := \lambda_{\text{next}}$.

The bisection algorithm operates on the convex hull of the $R(D)$ curve of each slice. Consequently, points which lie above that, and hence are not $R(D)$ optimal, are not considered by the algorithm. One can easily verify that actual $R(D)$ curves from real sequences are to a significant degree convex (i.e. only a few points are above the convex hull), particularly for P and B pictures. In some cases, if the $R(D)$ curve of a slice is sufficiently misbehaved, the bisection algorithm can be set off track, with a resulting underutilization of the target bit budget. In order to mitigate this effect, and also to speed up operation, each iteration considers a continuously shrinking interval of possible breakpoint values ("pruning"). This will result in convergence of the algorithm to a much smaller set of non-convex points. The computational overhead of the algorithm is small, and convergence is achieved within 8–10 iterations.

The collection of necessary data in (2) requires only parsing of the bitstream up to inverse quantization of the DCT coefficients. Since this represents a small fraction of the complete decoding process, the algorithm has complexity less than that of a decoder. The window N in which the algorithm operates is a design parameter. Since rate shaping is performed on top of encoding (although not necessarily at the same time), it is desirable to minimize the additional delay introduced by the extra processing step. A plausible selection is then a single picture (frame or field). The target bit budget R_{budget} of each picture can be set to: $R_{\text{budget}} = (B_T/B)R - R_o$, where R is the size (in bits) of the currently processed picture, and R_o is the number of bits spent for coding components of the bitstream that are not subject to rate shaping. R is immediately available after the complete picture has been parsed. Allocated bits that are left over from one picture are carried over to the subsequent picture.

Since a full resolution picture (704×480) may contain up to 15,840 blocks (for a 4:4:4 format), the processing required within each iteration in order to find the breakpoint value that minimizes $D_i(b_i) + \lambda R_i(b_i)$ can be significant. Consequently, it is worth examining *clustering* approaches, in which a common breakpoint value is selected for a set of macroblocks. We refer to such algorithms as $C(n)$, where n is the number of sequential macroblocks contained in each cluster. An additional benefit of clustering is that the distortion can be defined on only the luminance part of the signal, hence greatly simplifying the implementation. Clustering, of course, will degrade performance; for example, the C(44) algorithm reduces the quality by about 2 dB, but at a substantial decrease in complexity.

4 Mixed-Mode Rate Shaping

When all types of picture coding types are used (I, P, and B) the problem is significantly more complex. The decoding process for the original and the rate shaped signal can be described by $P_i = \mathcal{M}_i(P_{i-1}) + e_i$ and $\hat{P}_i = \mathcal{M}_i(\hat{P}_{i-1}) + \hat{e}_i$, where P_i denotes the i-th decoded picture (in coding order), $\hat{P}_i$ denotes the rate shaped decoded picture, $\mathcal{M}_i(\cdot)$ denotes the motion compensation operator for picture i, and e_i and $\hat{e}_i$ denote the coded original and rate shaped prediction errors respectively. The first picture is assumed to be intra-coded, and hence $P_0 = e_0$ and $\hat{P}_0 = \hat{e}_0$. Although, for simplicity, a single reference picture is shown above for motion compensation, the expression can be trivially extended to cover the general case (which includes B-pictures).

We can then rewrite (1) as:

$$\min_{\sum_{i=1}^{N} R_i(b_i) \leq B_T} \left\| \sum_{p=1}^{M} \mathcal{M}_i(P_{i-1}) - \mathcal{M}_i(\hat{P}_{i-1}) + e_i - \hat{e}_i \right\| \tag{9}$$

where M is the number of pictures over which optimization takes place. Note that in general $\mathcal{M}_i(P_{i-1}) - \mathcal{M}_i(\hat{P}_{i-1}) \neq \mathcal{M}_i(P_{i-1} - \hat{P}_{i-1})$, i.e. motion compensation is a non-linear operation, because it involves integer arithmetic with truncation away from zero.

From (9) we observe that, in contrast with the intra-only case, optimization involves the accumulated error $a_i \equiv \mathcal{M}_i(P_{i-1}) - \mathcal{M}_i(\hat{P}_{i-1})$. Furthermore, due to the error accumulation process, rate shaping decisions made for a given picture will have an effect in the quality and partitioning decisions of subsequent pictures. As a result, an optimal algorithm for (9) would have to examine a complete group of pictures (I-to-I), since breakpoint decisions at the initial I-picture may affect even the last B or P picture. Not only the computational overhead would be extremely high, but the delay would be unacceptable as well.

An attractive alternative algorithm is one that solves (9) on a picture basis, and where only the error accumulated from past pictures is taken into account; this algorithm will be referred to as *causally optimal*. Note that in order to accurately compute a_i, two prediction loops have to be maintained (one for a decoder that receives the complete signal, and one for a decoder that receives only partition 0). This is because of the nonlinearity of motion compensation, which involves integer arithmetic with truncation away from zero. With the penalty of some lack in arithmetic accuracy, these two loops can be collapsed together.

The causally optimal problem can be formulated as follows:

$$\min_{\sum_{i=1}^{N} R_i(b_i) \leq B_T} \left\{ \sum_{i=1}^{N} \hat{D}_i(b_i) \right\} \tag{10}$$

with

$$\hat{D}_i(b_i) \equiv \sum_{k} A^i(k)^2 + \sum_{k \geq b_i} 2A^i(\mathcal{I}(k))E^i(k) + E^i(k)^2 \tag{11}$$

where N is such that a complete picture is covered, $A^i(k)$ is the k-th DCT coefficient (in zig-zag scan order) of the of the i-th block of the accumulated error a_i, and $\mathcal{I}(\cdot)$ maps run/length positions from the prediction error $E^i(\cdot)$ to actual zig-zag scan positions. This minimization problem can be solved using the Lagrangian multiplier approach of Section 3, with this new definition for the distortion $\hat{D}$.

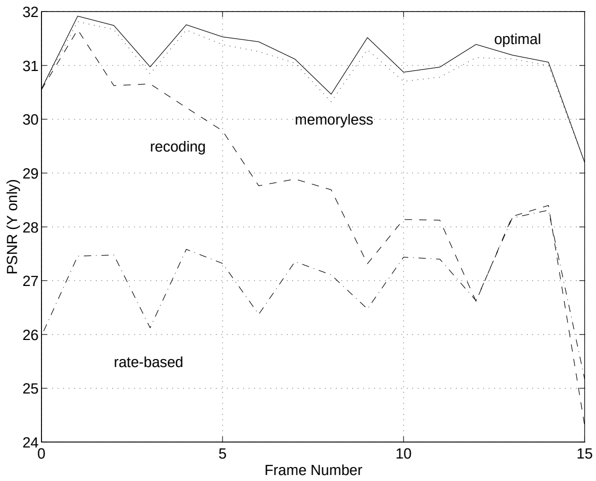

Fig. 4. Results of various rate shaping algorithms on the "Mobile" sequence, MPEG-2 coded at 4 Mbps and rate shaped at 3.2 Mbps.

An important issue in mixed-mode coding is the target bit budget that will be set for each picture. In a typical situation, I and P picture DCT coding requires a significant number of bits, while B picture sizes are dominated by header and motion vector coding bits. Consequently, B pictures provide much less flexibility for data partitioning. In order to accommodate this behavior, I and P pictures are assigned proportional bit budgets as in Section 3; for B pictures the same is done, except when the resulting bit budget is negative, in which case it is set to 0. The negative budget, however, is accounted for, so that the bits spent for the B picture are subtracted from the budget of the immediately following picture. Note that an optimal bit allocation for each picture would be a direct by-product of the optimal (non-causal) algorithm.

The complexity is solving 10 is significant, and can be shown to be between that of a decoder and an encoder. In order to examine the benefit of error accumulation tracking, one can apply the intra-only algorithm of Section 3 to the mixed-mode case, since the only difference is the accumulated error term a_i. Surprisingly, the results of this *memoryless* mixed-mode partitioning algorithm are almost identical. Fig. 4 shows the relevant PSNR values for the "Mobile" sequence; the difference is in general less than 0.1 dB and the curves can hardly be

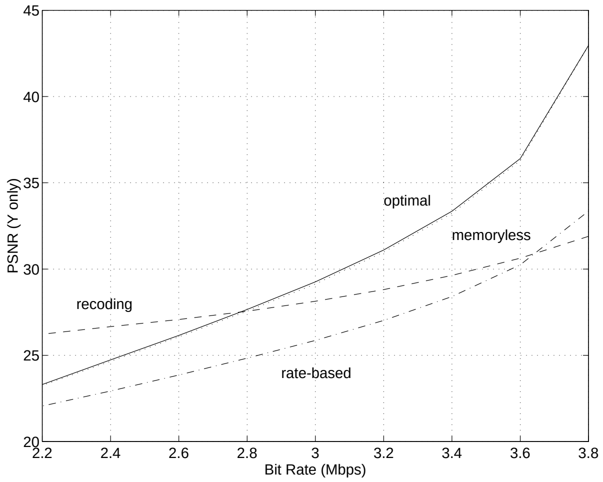

Fig. 5. Results of various rate shaping algorithms on the "Mobile" sequence, MPEG-2 coded at 4 Mbps and rate shaped at various different target bitrates.

distinguished. It turns out that this holds for a wide range of bit rates (Fig. 5), although the difference increases slightly to 0.2-0.3 dB. This is a very important result, as it implies that we can dispense completely with the error accumulation calculation and its associated computational complexity, for a minimal cost in performance: the quality degradation between the causally optimal and memoryless algorithms will be perceptually insignificant, across the spectrum of cluster sizes and partition rates.

For comparison purposes, we also examine the performance of a purely rate-based optimization algorithm. Breakpoint selection here is performed proportionally to the number of bits used to originally code each block. Fig. 4 depicts the results obtained on the "Mobile" sequence, coded at 4 Mbps and rate shaped at 3.2 Mbps, while Fig. 5 shows average PSNR values for a wide spectrum of rates. Fixed input and output rates have been selected here for simplicity; similar results can be obtained for more complex rate characteristics. All algorithms (except from recoding) are based on $C(1)$ clustering; i.e. breakpoint selection is performed on a macroblock basis. It is important to note that regular recoding gives inferior results to both the optimal and memoryless algorithms for a wide range of rates, while the latter two can hardly be distinguished.

5 Concluding Remarks

The concept of Dynamic Rate Shaping was introduced as an adaptation mechanism between coded video rate characteristics and transport service capabilities, and analyzed in an operational rate-distortion context. For the case of constrained DRS, an optimal algorithm based on Lagrangian multipliers was derived for intra-only coding. For the mixed-mode case (I, P, and B pictures) the optimal algorithm was shown to possess significantly high complexity and delay, and a causally optimal algorithm was introduced. It was then shown that a memoryless version of the algorithm performs almost identically, hence significantly simplifying the implementation complexity with no compromise in terms of quality. It was also shown that this DRS approach can outperform regular recoding for a wide range of rates. Due to its relative simplicity, the algorithm (particularly clustered versions, e.g. C(4)) can be implemented in software and operate in real-time on high-end general purpose CPUs.

References

1. Generic Coding of Moving Pictures and Associated Audio (MPEG-2). ITU-T Draft Recommendation H.262, ISO/IEC 13818-2 Draft International Standard, 1994.
2. A. Campbell. A Dynamic QoS Management Scheme for Adaptive Hierarchically Coded Flows. In *Proceedings of the 5th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV-95)*, April 1995.
3. A. Eleftheriadis and D. Anastassiou. Optimal Data Partitioning of MPEG-2 Coded Video. In *Proceedings of the 1st IEEE International Conference on Image Processing*, pages I.273–I.277, August 1993.
4. A. Eleftheriadis, S. Pejhan, and D. Anastassiou. Architecture and Algorithms of the Xphone Multimedia Communication System. *ACM/Springer Verlag Multimedia Systems Journal*, 2(2):89–100, August 1994.
5. K. Ramchandran and M. Vetterli. Rate-distortion optimal fast thresholding with complete JPEG/MPEG decoder compatibility. In *Proceedings of the Picture Coding Symposium '93*, March 1993.
6. H. Kanakia, P. P. Mishra, and A. Reibman. An Adaptive Congestion Control Scheme for Real-Time Packet Video Transport. In *Proceedings of the ACM SIGCOMM '94 Conference*, pages 20–31, September 1993.
7. Didier LeGall. MPEG: A Video Compression Standard for Multimedia Applications. *Communications of the ACM*, 34(4):46–58, April 1991.
8. Y. Shoham and A. Gersho. Efficient bit allocation for an arbitrary set of quantizers. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(9):1445–1453, 1988.