Two-pass MPEG-2 variable-bit-rate encoding

by P. H. Westerink
R. Rajagopalan
C. A. Gonzales

Many MPEG-2 encoding applications are real-time; this implies that the video signal must be encoded with no significant lookahead. However, there exist non-real-time applications that do enable us to first analyze a video sequence entirely, and, using the analysis results, to optimize a second encoding pass of the same data. One example of such an application is the digital video disk (DVD), which is designed to facilitate a variable-bit-rate (VBR) output stream. In that case, it is possible to let the MPEG-2 encoder produce a video sequence with a constant visual quality over time. This is in contrast to constant-bit-rate (CBR) systems, where the rate is constant but the visual quality varies with the coding difficulty. This paper describes a two-pass encoding system that has as its objective to produce an optimized VBR data stream in a second pass. In a first pass, the video sequence is encoded with CBR, while statistics concerning coding complexity are gathered. Next, the first-pass data is processed to prepare the control parameters for the second pass, which performs the actual VBR compression. In this off-line processing stage, we determine the target number of bits for each picture in the sequence, such that we realize the VBR objective. This means that the available bits are appropriately distributed over the different video segments such that constant visual quality is obtained. To be able to quantify the constant visual quality, perceptual experiments are described and a practical model is fitted to them. Exceptional cases such as scene changes and fades are detected and dealt with appropriately. We also ensure that the second-pass compression process does not violate the decoder buffer boundaries. Finally, the encoding is performed again, but now under control of the processed first-pass data. During the running of this second pass, a run-time bit-production control mechanism monitors the accuracy and validity of the first-pass data, correcting errors in prediction and observing the buffer boundaries. Results are compared to CBR operation.

1. Introduction

The primary application that prompted the development of two-pass VBR algorithms is the digital video disk (DVD). This is a new storage device that has been standardized for video to use the MPEG-2 video coding standard [1]. The storage capacity is such that a full two-hour video can be recorded at an average bit rate of 4 Mb/s. The system output bit rate can support up to approximately 10 Mb/s. By using output buffering, a DVD essentially has a variable output bit rate from 0 up to this

0018-8646/99/$5.00 © 1999 IBM

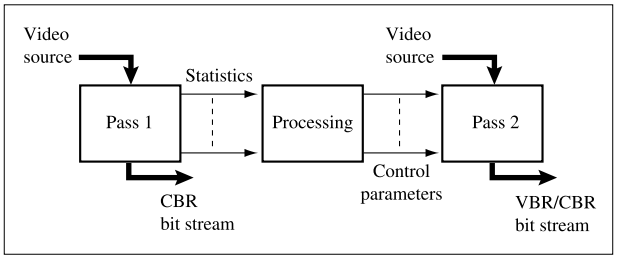

Figure 1

Schematic overview of a two-pass MPEG-2 compression system.

maximum of 10 Mb/s. Thus, whereas most conventional MPEG-2 coding systems are designed for a constant bit rate (CBR), an MPEG-2 encoder for DVD can be designed for a variable bit rate (VBR). CBR systems typically produce a constant-bit-rate stream, but inevitably with a corresponding variable picture quality. VBR, however, has the potential to produce constant picture quality throughout an entire video sequence. Such constant quality can be obtained by appropriately distributing the total available bits over the different video segments.

To assign bit rates to specific segments of a video sequence such that we attain constant visual quality, it is optimal to have knowledge of the characteristics of the entire video. This can only be done by playing the whole sequence and gathering certain statistics over time. Thus, a VBR system for DVD, in which we wish to distribute the available bits optimally over the video, is essentially a multiple-pass system. A one-pass VBR system can be designed as well, but it will always be suboptimal.

Brief examples of two-pass VBR methods can be found in [2] and [3]. In [2], a visual quality measure is defined for each macroblock as the product of a nominal quantization scale and some perceptual factor, which is not further specified. The bit allocation for the macroblocks in the second pass is done by lexicographically ordering all of the blocks over the entire sequence, but constrained by the buffer and channel requirements. This makes the method computationally quite intensive, especially for longer sequences. The method described in [3] is largely based upon the mean-squared error, although some perceptual measure can be incorporated by using the spatial activity.

This paper describes a complete practical two-pass MPEG-2 encoding system that can be tuned to produce a VBR stream in a second pass. In Section 2, we describe the rate-control principles as they apply to MPEG-2 CBR and VBR encoding. In Section 3, a definition of constant visual quality for VBR is given, and perceptual experiments are described that establish a practical formulation. The VBR principle is set forth in Section 4, where the first-pass data is processed to produce second-pass bit-production targets, and tuned to yield a VBR video with a constant visual quality. A buffer-underflow-prevention algorithm is also described. Section 5 deals with the actual second pass, where the run-time buffer- and rate-control issues are addressed. Section 6 revisits the first-pass gathering of statistics. Data processing methods are described that refine the two-pass system to produce a better quality, primarily in problem areas such as scene changes and fades. Finally, in Section 7 practical results are given, and conclusions are drawn in Section 8.

2. Two-pass encoding principle

- *Two-pass system description*

A schematic overview of a two-pass system is shown in **Figure 1**. The video data is run through a first pass by operating the video encoder with conventional CBR control. This pass outputs certain statistics of the input video data, and these statistics are next processed. The processing is done off-line, as it does not require the video data itself. However, it does require all of the video statistics, and the processing can therefore not be performed until the whole first pass through the video has finished. The task of the processing stage is to compute the parameter settings for the second pass. Finally, using the prepared control parameters, the second pass is run to produce an output which is the desired and optimized CBR or VBR bitstream.

- *MPEG-2 encoder rate control*

The particular picture quality and rate of an MPEG-2 encoder is achieved by selecting a specific quantizer scale $Q_{i,m}$ for each macroblock m in picture i. This value is calculated for each macroblock by combining a picture-global quantization scale Q_i with a perceptual factor $p_{i,m}$:

$$Q_{i,m} = p_{i,m} Q_i \,. \tag{1}$$

This perceptual modulation factor should be set such that regardless of the picture-global quantization scale Q_i, each encoded macroblock m in this picture i will appear visually to have the same quality. A common measure for $p_{i,m}$ is based upon the degree of busyness in a macroblock. This relies on the notion that "flat" areas visually cannot tolerate the same magnitude of coding error as busy areas.

The rate of the MPEG-2 encoder can therefore be controlled only indirectly, by setting the global quantization scale Q_i. The bits per picture b_i and quantization scale Q_i are linked via a bit-production model,

$$b_i = f(Q_i). \tag{2}$$

In general, Equation (2) is a decreasing function. An example is a logarithmic function, such as the rate-distortion curve for a memoryless source where the samples have a Gaussian probability density function. The bit-production model as defined in MPEG-4 for rate-control experiments [4] is

$$b_i = \left(\frac{c_{1,i}}{Q_i} + \frac{c_{2,i}}{Q_i^2} \right) S_i , \tag{3}$$

where S_i is the sum of absolute differences (SAD), i.e., the prediction error, and $c_{1,i}$ and $c_{2,i}$ are the model parameters for picture i. However, the problem with this kind of model is that it has two degrees of freedom. The model therefore needs at least two distinct points on the curve in order to estimate its parameters. A model that has only one parameter is suggested in the MPEG-2 test model 5 [5]:

$$b_i(Q_i) = \frac{c_i}{Q_i} . \tag{4}$$

Here c_i is the bit-production model parameter for picture i. Measurements have shown that this is actually a fairly accurate model for a large variety of scenes.

- *MPEG-2 decoder buffer model for CBR*

The buffer fullness model for CBR is shown in **Figure 2**. We describe the state of the buffer just before picture $i + 1$ is removed from the buffer by the recurrence

$$\begin{aligned} B_0 &= B_{\text{init}} , \\ B_{i+1} &= B_i - b_i + RT_i . \end{aligned} \tag{5}$$

To prevent underflow and overflow, the buffer levels must always lie within the range

$$b_i \le B_i \le B_{\max} . \tag{6}$$

- *MPEG-2 decoder buffer model for VBR*

In VBR, the bits may enter the decoder buffer at any bit rate, but with a given maximum $R_{\max}$. In the MPEG-2 standard [1], this is modeled with bits always entering the buffer at a constant peak rate $R_{\max}$ until the buffer is full, at which point the bit rate is temporarily set to zero until the next picture is removed from the buffer. The buffer fullness model for this process is depicted in **Figure 3**. We describe the state of the buffer just before picture $i + 1$ is removed from the buffer by the recurrence

$$\begin{aligned} B_0 &= B_{\max} , \\ B_{i+1} &= \min(B_{\max}, B_i - b_i + R_{\max} T_i). \end{aligned} \tag{7}$$

Here B_0 is the initial buffer fullness, which, following the MPEG-2 standard [1], we set to the buffer size $B_{\max}$; T_i is the time it takes to display picture i. Note that because of a possible repeat-first-field situation, this time may vary from picture to picture. To prevent the buffer from underflowing, all of picture i must have arrived in the buffer before it is removed; that is,

$$B_i \ge b_i . \tag{8}$$

We see that in contrast to the case for CBR, there is no buffer overflow constraint.

When running VBR in a second pass, we must follow the buffer model according to Equation (7). Further, we can disable padding, because we cannot obtain an overflow. Moreover, in the VBR scenario we wish to take advantage of any scene that would require padding in the pure CBR case, because we can transfer bits from such scenes to more complicated ones. Finally, as a final decoder buffer overflow prevention measure in CBR, P- and B-macroblocks may be forced to be coded as I-macroblocks (nonpredictive mode) whenever overflow is impending; this can also be disabled for VBR.

- *Variable-bit-rate coding principle*

Given the total number of bits available for an entire video sequence, e.g., the total capacity of a DVD, we generally wish to encode so that quality is optimized for the video as a whole. For VBR, this is defined as the highest possible visual quality over the whole sequence. However, this goal may not be attainable when the maximum bit rate $R_{\max}$ is insufficient, resulting in a

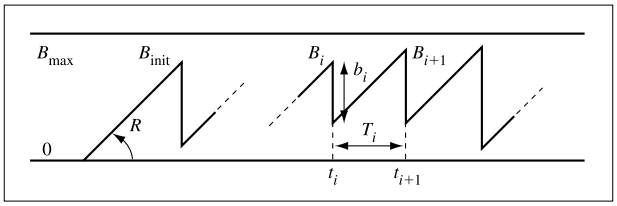

Figure 2

CBR modeling of buffer fullness.

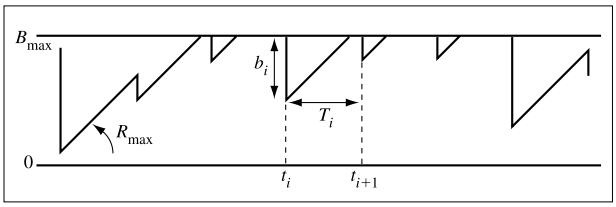

Figure 3

DVD VBR modeling of buffer fullness.

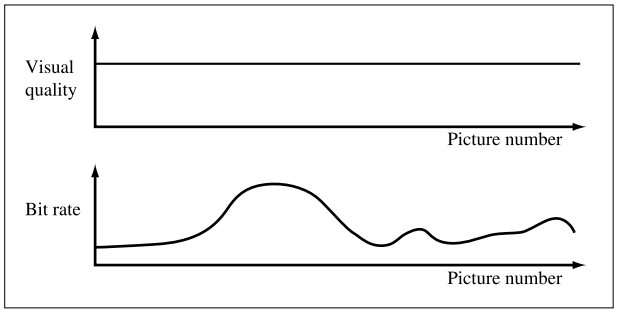

Figure 4

Constant visual quality and variable bit rate for a long video sequence.

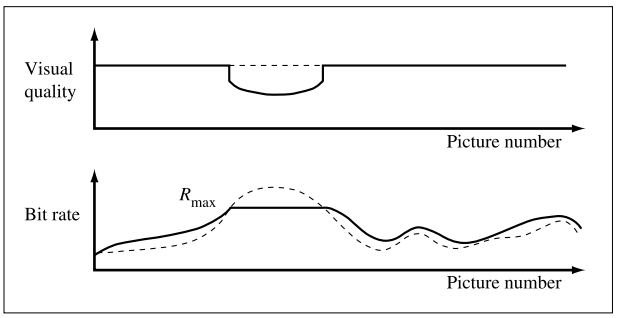

Figure 5

Visual quality and bit rate for a video sequence in which constant quality (distortion) cannot be maintained because of the buffer underflow constraint.

decoder buffer underflow. For example, suppose we deliberately impose a constant quality on a video sequence. This results in a certain bit rate per picture that is generally not constant, as illustrated in **Figure 4**. In this situation we have reached our goal if during the sequence we never exceed the maximum bit rate R_{max} and we have used the exact number of bits available. However, if the bit rate does exceed this maximum rate during some segment in time, the buffer underflow constraint is violated and the constant-quality solution is not viable. In that case we must lower the quality for that brief segment of pictures for which we would have $R > R_{max}$, as shown in **Figure 5**. We are forced to restrict the rate to $R = R_{max}$ for the "troublesome" segment, thereby necessarily lowering the visual quality to some extent. Note that we can spend the excess bits that become available on the rest of the sequence, effectively increasing the quality there.

3. Constant visual quality for VBR

- *Scene statistics*

In the first pass we run the data through the encoder while using the regular real-time constant-bit-rate control algorithm. The rate at which this is run is determined by the total available number of bits and by the duration of the video. Typically, we expect this to be in the range of 4 to 5 Mb/s. We then gather various numerical values for each picture.

The rate-control algorithm sets a picture-global quantization scale in the first pass for each picture. This value is very important, since, together with the number of bits produced, it determines our bit-production model of Equation (4). Besides the quantization scale and the number of bits per picture, other available data include the picture type, the number of intracoded macroblocks, a measure of the spatial activity, and a measure of the temporal activity.

The spatial activity of a picture is the average of the macroblock spatial activities. For each macroblock, the average pixel luminance value is first calculated. Next, the absolute difference between each pixel value and this average is accumulated and averaged. This yields an absolute norm variance measure for each macroblock. The average over all macroblocks in the picture is then used as the spatial activity for that picture.

The temporal activity is based upon the motion vectors. For both predicted picture types, i.e., P (predicted) and B (bidirectionally predicted), for each macroblock the forward-prediction vector is used that yields the lowest prediction error. However, for different pictures motion vectors do not have the same scale, and this must be compensated for. For example, if we have two B-pictures between reference pictures, the P-picture motion vectors will have a scale of three pictures, that being the temporal distance between the P-picture and the last reference picture from which the motion vectors were estimated. For the B-pictures, the scale is the distance from the last reference picture, which is 1 for the first B-picture, while the second B-picture has a scale of 2. Finally, as was done with the spatial activity, an absolute norm variance is calculated for both the horizontal and the vertical direction of the motion vector for each macroblock. The sum of the two pseudovariances is the macroblock temporal activity, which is summed to obtain the picture temporal activity.

- *Visual perception experiments*

We have defined our two-pass objective as the obtaining of a perceptual constant quality throughout a long video sequence. In a constant-bit-rate scenario, because of complexity differences, certain scenes will visually have a

Figure 6

Typical key frames of each training set sequence.

higher quality than others. Therefore, in our definition of constant perceived quality, we take bits from the easier scenes and put them into the more complicated scenes, thus creating variable-bit-rate coding. The issue then becomes how to redistribute the bits among the various scenes in such a manner that the scenes appear *perceptually* to be of the same quality.

Experiments were carried out to determine constant visual quality across different scenes. A test set of nine scenes of five seconds each was used. To illustrate, a typical key frame of each scene is shown in **Figure 6**. The scenes were all different in characteristics, ranging from low to high motion activity, and low to high spatial complexity. The characteristics within each scene were stable (i.e., throughout a scene, the quantization scale, the corresponding average bit rate, the spatial activity, and the temporal activity all remained constant).

Each scene was individually coded at various bit rates by fixing the quantization scale for each bit rate. The range extended from a very low bit rate, with a very poor quality, up to a very high bit rate, with a quality virtually indistinguishable from the original. The characteristics of all of the input scenes, as coded at different fixed quantization scales and corresponding bit rates, are given in **Table 1**. The quantization scale at 4 Mb/s, Q_4, is obtained by interpolating Table 1 for each scene, using Equation (3) with its two parameters, $p_1 = c_{1,i}S_i$ and $p_2 = c_{2,i}S_i$, based upon two surrounding points. The data in Table 1 is typically the kind we obtain as statistics from a first-pass run at a constant bit rate such as 4 Mb/s. The scene characteristics Q_4, A_s, and A_t are also shown in **Figure 7**, sorted by increasing Q_4.

The objective of the perceptual experiments was next to compare the different scenes at the different bit rates, and to determine a particular level for each scene such that all scenes are perceived to have the same quality. The test setup consisted of two monitors side by side, each monitor playing a different scene. While one scene was kept the same, the other was varied in quantization scale. A test person was then asked whether this other scene was perceived as being of lesser, greater, or the same quality. To make the results more reliable, monitors were switched, three different test persons were used, and experiments were randomly repeated by showing each comparison (at least) twice.

In this way, a bit rate with corresponding quantization scale was determined for each scene, such that all scenes were perceived to be of equal quality. This experiment was done for two different levels of perceptual quality. These visual quality levels can respectively be described with the very subjective terms “fair” and “good.” The results are given in **Table 2**, sorted by Q_4. The results were

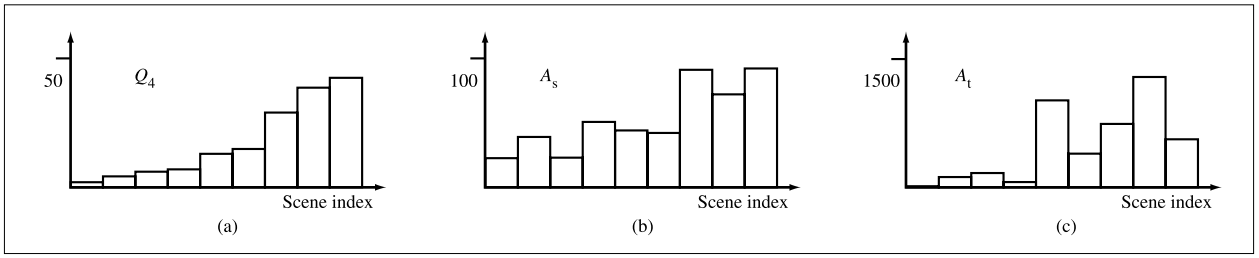

Figure 7

Measured characteristics of nine scenes showing (a) first-pass quantization scale at 4 Mb/s, Q_4; (b) spatial activity, A_s; and (c) temporal activity, A_t.

Table 1 Characteristics for nine different scenes, each coded with various fixed quantization scales. Q_4 is the quantization scale at 4 Mb/s, A_s is the spatial activity for each scene, and A_t is the temporal activity for each scene. The numeric column headings are the constant quantization scales, heading the corresponding bit rates. For example, coding the scene "lady" at a fixed quantization scale of 10 corresponds to an average bit rate of 2.8 Mb/s.

Name	Q_4	A_s	A_t	*2*	*4*	*6*	*8*	*10*	*12*	*16*	*20*	*24*	*32*	*48*
pac10	1.6	22	7	3.5	1.0	0.9	0.7	0.7	0.7	0.6	0.6	0.6	0.6	0.5
diagr	4.4	39	138	15	5.4	2.2	1.6	1.5	1.4	1.2	1.1	1.1	1.0	0.9
suzie	5.7	23	177	41	8.7	3.7	2.6	2.1	1.8	1.5	1.4	1.3	1.3	1.2
lady	6.9	51	49	35	11	4.6	3.3	2.8	2.3	1.7	1.5	1.3	1.2	1.0
tennis	13.2	43	1027	—	14	8.9	6.4	5.4	4.4	3.3	2.8	2.4	2.0	1.6
fount	15.1	43	418	24	12	8.4	6.4	5.6	4.8	3.8	3.2	2.8	2.2	1.7
flower	29.1	91	755	—	—	21	16	13	10	7.5	6.0	4.8	3.6	2.5
cheer	39.0	72	1316	—	36	23	18	15	12	9.1	7.4	6.2	4.7	3.4
mobile	42.9	92	568	—	—	31	22	19	16	11	9.0	7.5	5.3	3.6

Table 2 Perceptual measurement results: bit rates for each scene for which the perceptual quality was perceived to be the same. Results are given for two separate experiments.

	pac10	*diagr*	*suzie*	*lady*	*tennis*	*fount*	*flower*	*cheer*	*mobile*
Fair	0.65	1.53	2.12	1.73	2.76	2.79	4.85	7.36	8.98
Good	0.73	1.59	2.63	2.26	3.31	3.80	7.52	9.11	11.35

further perceptually verified by creating a composite video consisting of all nine scenes at those bit rates (quantization scales) found, and checking with test persons that there are no scenes that stand out as being of lower or higher quality than any of the others in the composite video.

The next step is to model the acquired training set data with a mathematical function, with which we can determine the second-pass quantization scale and bit rate for any scene, on the basis of first-pass statistics.

- *Reliability of the visual perception experiments*

By their very nature, visual perception experiments are subjective; comparing different scenes in general can best be referred to as "comparing apples and oranges." Different viewers have different impressions, look at different parts of the scene, and regard different coding artifacts as being different in nature and severity. Furthermore, viewers "learn" to look for certain artifacts, especially after seeing a particular scene more than once. As such, the denominations of "fair" and "good" in Table 2 suggest a larger visible difference than actually exists. In a comparison of "fair" and "good" quality for the same scene, the difference is usually visible, though not in all cases. For example, see the column for the "diagr" scene in Table 2, where we have a bit-rate difference of only 0.06 Mb/s. This difference is visible only after a long and close examination of the still pictures. Also, when comparing different scenes, it is even less obvious to viewers that one scene is clearly superior or inferior to the other. There seems to be an inherent limitation of the reliability of the performed perception measurements.

Other factors further influence the limited accuracy of the results of the perceptual experiments. For instance, the training set that was used is fairly small, and can never pretend to capture all possible scenes and situations. In that sense, the training set can never be large enough, as there will always be scenes that do not exactly fit the statistics of a training set. Note that this is generally the case with any situation where training is performed.

We must be aware of this limited reliability when fitting a mathematical model to the experimental results of Table 1 and Table 2. For example, experiments with different fitting models for long video sequences have shown that models that relatively take more bits out of the easier scenes yielded a lower overall visual quality. This is because these more "aggressive" models sometimes take too many bits from the easier scenes, producing too great a decrease in the relative quality of those scenes. Apparently, the overall visual quality of a video is rated not so much by the high quality of certain scenes, but more by the scenes of the lowest visual quality, the ones that might show obvious artifacts and thus stand out.

Therefore, we should be wary of finding a mathematical function that will aggressively remove bits from the easier scenes. It is better to fit a function somewhat conservatively, that is, to define a function that is biased toward a constant-bit-rate function. Of course, this does not mean that we should use a constant function, and thus realize CBR, since visual evaluation of experimental results clearly shows the advantages of VBR over CBR.

- *Modeling the experimental results*

Table 1 displays the characteristics of each scene n. They are, respectively, the quantization scale $Q_{1,n}$ for running the first pass at 4 Mb/s, the spatial activity $A_{s,n}$, and the temporal activity $A_{t,n}$. The objective of this section is to define a function that uses the characteristics of Table 2 to find the bit rates $R_{2,n}$ from Table 2, for each scene n:

$$R_{2,n} = f(Q_{1,n}, A_{s,n}, A_{t,n}, R_{2,\text{ave}}). \tag{9}$$

Here $R_{2,\text{ave}}$ is the average bit rate for the second pass, which determines the overall quality level on which we operate. For an average bit rate of about $R_{\text{good}} = 4.9$ Mb/s, a perfectly fitted function yields the results for "good" in Table 2, while the results for "fair" are for an average bit rate of $R_{\text{fair}} = 3.8$ Mb/s. These numerical values can be calculated as the averages for, respectively, the "good" and "fair" bit rates in Table 2. It turns out that we can reduce the dependency of the general function in Equation (9) on $R_{2,\text{ave}}$ to

$$R_{2,n} = R_{2,\text{ave}} f(Q_{1,n}, A_{s,n}, A_{t,n}). \tag{10}$$

It can be verified that Equation (10) models the data well by using

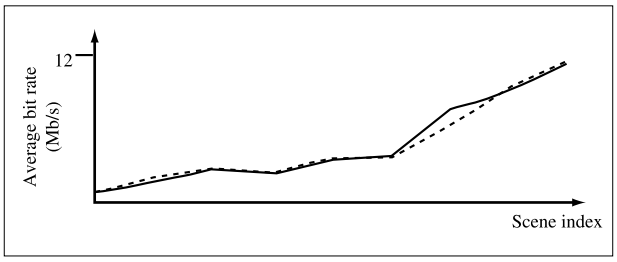

Figure 8

Scene bit rates for constant perceptual quality, at an average of 4.9 Mb/s (solid line) and at an average of 3.8 Mb/s scaled up by multiplication by 4.9/3.8 = 1.29 (dashed line).

$$\begin{cases} R_{n,\text{fair}} = R_{\text{fair}} f(Q_{1,n}, A_{s,n}, A_{t,n}) \\ R_{n,\text{good}} = R_{\text{good}} f(Q_{1,n}, A_{s,n}, A_{t,n}) \end{cases} \tag{11}$$

and by noting that if Equation (10) is correct, we must have

$$R_{n,\text{good}} = \left(\frac{R_{\text{good}}}{R_{\text{fair}}}\right) R_{n,\text{fair}}\,. \tag{12}$$

This relation is illustrated in **Figure 8**, where we have projected the data points for the lower bit rates $R_{n,\text{fair}}$ onto the data points for the higher bit rates $R_{n,\text{good}}$ by multiplying each point by the ratio of the average bit rates, 4.9/3.8. As can be seen, the match is very close. This is quite an important implication, as it substantially simplifies the assignment procedure for the second pass, especially with regard to meeting the total bit budget.

When comparing the plot in Figure 7(a) with the data in Figure 8, we see that a very large correlation exists between the quantization scale at 4 Mb/s Q_4 and the bit rates to be fitted to the data. On the other hand, very little resemblance can be found with the other two characteristics, A_s and A_t. We therefore concentrate on a function that uses only the quantization scale:

$$R_{2,n} = kf(Q_{1,n}), \tag{13}$$

where the parameter k is calculated from the constraint

$$\frac{1}{N}\sum_{n=1}^{N} R_{2,n} = R_{2,\text{ave}}\,. \tag{14}$$

The particular function we use is

$$R_{2,n} = k(Q_{1,n})^p\,. \tag{15}$$

One advantage of this function is that with the bit-production model from Equation (4) and the bit-budget constraint of Equation (14), this function is independent of the (constant) bit rate that was run in the first pass.

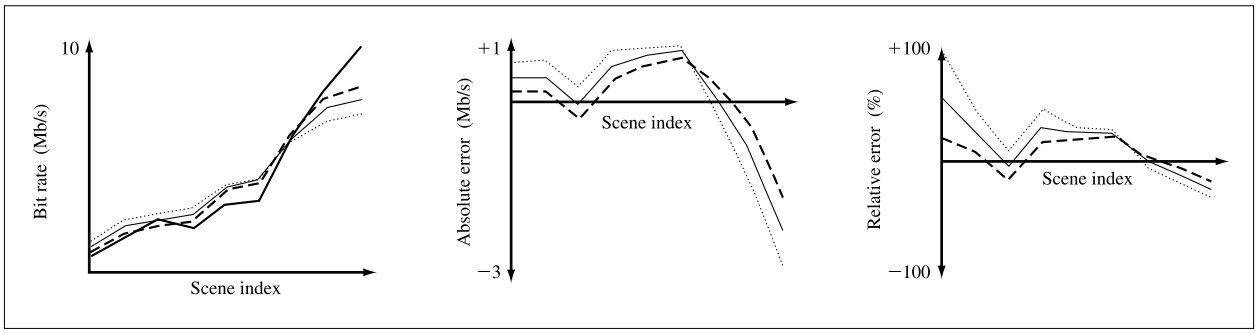

Figure 9

Fitting results for the nine test scenes, for three different values of the power p in Equation (15). The dotted line is for $p = 0.5$, the solid line for $p = 0.6$, and the dashed line for $p = 0.7$. The thicker line in the left plot is the bit rate that was fitted to the experimental data.

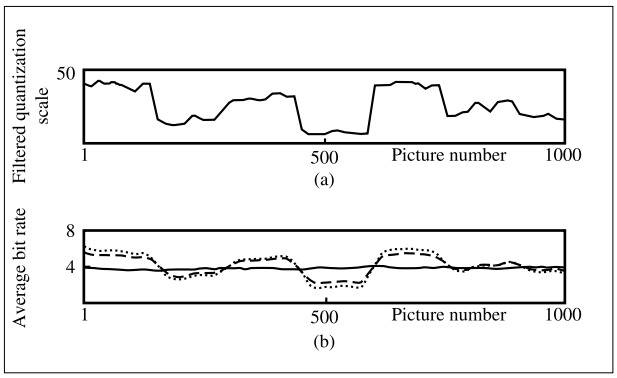

Figure 10

Filtered quantization scales (a) and average bit rates (b). The plot of average first-pass bit rates is the solid line for the fairly constant 4 Mb/s, the plot for power $p = 0.5$ is the dashed line, and $p = 0.7$ is the dotted line.

Furthermore, by choosing an appropriate value for the power p, we can control how "aggressive" we wish to make the fitting function. Some fitting results for various values of the power p are shown in **Figure 9**.

The bit rate that is fitted to the data is derived from Table 2, for an average bit rate of 4 Mb/s. This is the thicker line in the left plot. The middle plot in Figure 9 shows the fitting errors, which are defined as the differences between the fitted-to and the fitted data. Shown on the right are the relative fitting errors, which are the fitting errors divided by the rates that were fitted to. It can be seen that each value for the power p results in a certain tradeoff between allotting more or fewer bits to the easier scenes compared to the more complicated scenes. Note that any kind of fit will always have negative and positive errors, since the average bit rate of the fitted curves must have the same bit rate as the curve that was fitted to.

To further evaluate the effect of the power p (that is, to see how aggressive a function we can use), additional experiments were carried out. Some of the characteristics of the sequences that were used are plotted in **Figure 10**. Shown are the quantization scales (top) and the average bit rates (bottom), where the bit-rate curves are respectively for the first pass (run at 4 Mb/s), and for the second pass using powers of $p = 0.5$ and $p = 0.7$. The average bit rate is calculated as a running average over the pictures, with a window size of 45 pictures, which is exactly three groups of pictures (GOPs). The plot closest to the constant bit rate, the dashed curve, is the one with the lowest power, in this case $p = 0.5$. We see that the bit rates for powers of $p = 0.5$ and $p = 0.7$ are different, although visually the difference was not that apparent. However, when viewers were asked independently, the second-pass result for $p = 0.5$ was judged to be the better one, simply because the easiest scene looked slightly better (in this case, the segment of the video that was coded at the lowest bit rate in Figure 13, around picture number 500). Again, the overall quality seems to be decided by the easier scenes.

There still remains the question of what to choose for the power p. Choosing a value that is too high, such as $p = 0.7$, degrades the quality of the easier scenes too much, while a much smaller power, such as $p = 0.4$, may not fully exploit the VBR principle and may not enhance the quality of the difficult scenes as much as possible. The overall experience is that a power of $p = 0.5$–0.6 yields the best results.

4. Picture-by-picture control parameters for second-pass VBR

• *Picture target number of bits*

As described in the section on modeling experimental results, we use a function of the first-pass quantization scale to set a second-pass target bit rate. However, the measurements were performed for whole scenes and applied to bit rates, while in practice we wish to set a target number of bits for each individual picture. To that end, we note that to change the bit rate of a particular scene from a first-pass rate of $R_{1,n}$ to a second-pass rate of $R_{2,n}$, we should change the number of bits of each picture i in scene n proportionally:

$$b_{2,n,i} = \left(\frac{R_{2,n}}{R_{1,n}}\right) b_{1,n,i}\,. \tag{16}$$

With this we can rewrite Equation (15), but now for each picture i, and thus set the target number of bits $b_{2,i}$ for the second pass according to

$$b_{2,i} = k b_{1,i} (Q_{1,i})^p, \tag{17}$$

where the corresponding first-pass quantization scale is $Q_{1,n}$. Additionally, the target number of bits per picture must be such that the total bit budget B_{tot} is met:

$$B_{\text{tot}} = \sum_{i=1}^{N} b_{2,i}\,. \tag{18}$$

The bit budget is directly related to the second-pass average bit rate $R_{2,\text{ave}}$ via the total number of pictures N and the number of pictures per second f_{pic}:

$$\frac{1}{N} B_{\text{tot}} = \frac{R_{2,\text{ave}}}{f_{\text{pic}}}. \tag{19}$$

The factor k is thus calculated as

$$k = \frac{B_{\text{tot}}}{\sum_{i=1}^{N} b_{1,i} (Q_{1,i})^p}. \tag{20}$$

• *Allowing for biasing the individual picture target number-of-bits settings*

As stated in the previous section, we expect to obtain a constant perceptual quality for the entire sequence by setting the target number of bits for each picture according to Equations (17) and (20). However, after viewing the resulting video, we might conclude that certain scenes are not of the same perceptual quality as others, and should have their quality improved. Therefore, for such scenes we must be able to set a relatively higher or lower target number of bits. Another such example is that general video quality often benefits from biasing I-pictures over P-pictures and P-pictures over B-pictures. In such a case we wish to (de-) emphasize the target bits, on the basis of the picture type. To facilitate this, we introduce a weight w_i for each picture i:

$$b_{2,i} = w_i [k b_{1,i} f(Q_{1,i})]. \tag{21}$$

A weight larger than 1 results in a higher target number of bits, and thus a higher individual picture quality.

• *Buffer underflow protection*

Minimum multiplication factor for a segment of pictures

To prevent the buffer from underflowing, we consider segments of a certain number of pictures. A segment is usually a GOP, but this may be different in the vicinity of scene changes; see also Section 6. We assume that we have calculated the target number of bits b_i for each picture i in the segment. With this, and with the repeat-first-field information and the maximum bit rate $R_{\max}$, we can check whether the buffer will underflow for the segment. We can also calculate the buffer level at which the segment will end, that is, where the following segment will start. For each segment we can thus impose two conditions: no underflow, and to finish with at least a certain buffer fullness B_0. If either of the conditions is not met, each target is reduced proportionally. This is done by multiplying each target b_i in the segment by a common multiplication factor f.

In the following we calculate the maximum multiplication factor f, for which both conditions as described above are exactly met. If this factor is larger than 1, there is no need to take action; otherwise, the targets must be multiplied to meet the formulated conditions.

We assume that just before we remove the first picture $i = 0$ of the segment from the buffer, the fullness is B_0. Then, before we remove the next picture $i = 1$ from the buffer, we can have two situations:

$$\begin{aligned} &B_0 - f b_0 + r_0 \geq B_{\max} \qquad B_1 = B_{\max}\,, \\ &B_0 - f b_0 + r_0 < B_{\max} \qquad B_1 = B_0 + r_0 - f b_0\,. \end{aligned} \tag{22}$$

Here $r_0 = R_{\max} T_0$, being the maximum number of bits that enter the buffer in display time interval T_0. This interval is a variable to accommodate repeat-first-field situations. We have a similar situation for picture $i = 1$:

$$\begin{aligned} &f b_1 \leq r_1 - (B_{\max} - B_1) \qquad B_2 = B_{\max}\,, \\ &f b_1 > r_1 - (B_{\max} - B_1) \qquad B_2 = B_1 + r_1 - f b_1\,. \end{aligned} \tag{23}$$

By combining this with Equation (22), we obtain four possible cases for the buffer fullness B_2:

$$\begin{array}{ll}
(fb_0 \le r_0 - B_\Delta) \wedge (fb_1 \le r_1) & B_2 = B_{\max}, \\
(fb_0 \le r_0 - B_\Delta) \wedge (fb_1 > r_1) & B_2 = B_{\max} + r_1 - fb_1, \\
(fb_0 > r_0 - B_\Delta) \wedge (fb_0 + fb_1 \le r_0 + r_1 - B_\Delta) & B_2 = B_{\max}, \\
(fb_0 > r_0 - B_\Delta) \wedge (fb_0 + fb_1 > r_0 + r_1 - B_\Delta) & B_2 = B_0 + (r_0 + r_1) \\
 & \quad - (fb_0 + fb_1),
\end{array} \tag{24}$$

with $B_\Delta = B_{\max} - B_0$. In general, we have $n + 1$ possible values for the buffer fullness before picture n is removed:

$$\begin{cases} B_n = B_{\max} + R_{k,n-1} - fS_{k,n-1} & (k = 1, \cdots, n), \\ B_n = B_0 + R_{k,n-1} - fS_{k,n-1} & (k = 0), \end{cases} \tag{25}$$

where we have shortened the notation by using

$$R_{k,n} = \sum_{i=k}^{n} r_i \qquad S_{k,n} = \sum_{i=k}^{n} b_i . \tag{26}$$

Note that if there is no repeat first field, we have simply

$$R_{k,n} = (n + 1 - k) R_{\max} T, \tag{27}$$

where T is the fixed picture display time, the inverse display frequency.

Given the values for b_i, not all cases k in Equation (25) are possible. For example, in Equation (24), if

$$r_1 b_0 < (r_0 - B_\Delta) b_1 , \tag{28}$$

the third case from the top cannot occur, regardless of how we choose f. On the other hand, if the condition in Equation (28) is reversed, the second case from the top in Equation (24) is void. However, cases that are invalid for a particular set of b_i always result in higher values of f than other cases, because those are the ones where we have ignored the min-operator of Equation (7). And since we are looking for the worst case, the minimum multiplication factor f, we therefore need not explicitly exclude them. Thus, the conditions we have in Equation (24) are left out in Equation (25).

To prevent buffer underflow for a whole segment of N pictures, we must have that

$$B_n - fb_n \ge B_g \qquad (n = 0, \cdots, N - 1). \tag{29}$$

Here we have introduced a small guard band B_g to add some margin at the bottom of the buffer. Combining Equations (25) and (29), we thus obtain a multiplication factor for each case (k, n):

$$\begin{cases} f_{k,n} = \dfrac{R_{k,n-1} + (B_{\max} - B_g)}{S_{k,n}} & (n = 0, \cdots, N - 1) \\ & (k = 1, \cdots, n), \\ f_{k,n} = \dfrac{R_{k,n-1} + (B_0 - B_g)}{S_{k,n}} & (n = 0, \cdots, N - 1) \\ & (k = 0). \end{cases} \tag{30}$$

Note that the inclusion of fb_n from Equation (29) causes the index $n - 1$ for S in Equation (25) to change to n in Equation (30). As an additional condition, we wish to have a buffer fullness of at least B_0 at the start of the next segment:

$$B_N \ge B_0 . \tag{31}$$

Combining Equations (25) and (31), we thus obtain multiplication factors for the case of $n = N$:

$$\begin{cases} f_{k,n} = \dfrac{R_{k,n-1} + (B_{\max} - B_0)}{S_{k,n-1}} & (n = N) \\ & (k = 1, \cdots, n), \\ f_{k,n} = \dfrac{R_{k,n-1}}{S_{k,n-1}} & (n = N) \\ & (k = 0). \end{cases} \tag{32}$$

Finally, the segment multiplication factor is found with

$$f = \min (f_{k,n}) \qquad (n = 0, \cdots, N) \qquad (k = 0, \cdots, n), \tag{33}$$

where the multiplication factors $f_{k,n}$ for $n = 0, \cdots, N - 1$ are taken from Equation (30), and for $n = N$ from Equation (32).

Using the multiplication factors to adjust the picture targets

To control the buffer fullness, we first compute the multiplication factor f_k for each segment k, according to the previous section. After that, we may find that we have segments that have $f_k < 1$, which implies that the corresponding targets must be multiplied by this amount. For those segments, the number of bits is lowered, making bits available to be reassigned to other segments. An algorithm to multiply the targets for segments that have $f_k < 1$ and redistribute the freed-up bits proportionally is as follows:

1. Initialize. Zero the sum of bits to redistribute: $S_{red} = 0$. Zero the total sum of target bits of all segments that are still eligible to receive more bits: $S_{elig} = 0$.
2. Loop over all segments:
   - For every segment k for which $f_k < 1$, decrease the corresponding targets by multiplying them by f_k. Add the number of bits that become available to the sum of bits to redistribute S_{red}. The multiplication factor of this segment will be adjusted to exactly 1, which also indicates that it has been handled and cannot receive additional bits.
   - For the segments that have $f_k > 1$, add the picture targets to the total sum of target bits S_{elig}.
   - Skip segments that have $f_k = 1$.
3. If S_{red} equals 0, we have not found (any more) segments with $f_k < 1$: stop.
4. Calculate the redistribution multiplication factor $f_{red} = 1 + (S_{red}/S_{elig})$.
5. Loop over all segments:

- For the segments that have $f_k > 1$, multiply the targets by f_{red}. Adjust the segment multiplication factor f_k by dividing it by f_{red}.
- Skip segments that have $f_k = 1$ (after step 2, there should be no segments left with $f_k < 1$).

6. Since $f_{\text{red}} > 1$, step 5 for $f_k > 1$ could result in some segments to obtain $f_k < 1$. Return to step 1.

5. Run-time bit-production control in the second pass

The bit-production model of Equation (4) is only approximate; furthermore, we have only an estimate of the parameters c_i. Therefore, when running a second pass with the set targets, we obtain only an approximate bit production b_i. Consequently, we will probably not exactly meet the total bit budget B_{tot}. We therefore need some kind of control mechanism that monitors the actual bit production and adjusts the quantizer setting according to any mismatch with the expected and targeted bit production.

Just before removing picture k from the decoder buffer, we can determine the targeted ideal total number of bits produced so far by using

$$B_{k,\text{ideal}} = \sum_{i=0}^{k-1} b_{i,\text{ideal}}\,, \tag{34}$$

where $b_{i,\text{ideal}}$ equals the target number of bits set for the second-pass CBR or VBR. Then, with the true number of bits produced so far being B_k, the accumulated bit-production error is

$$\Delta_k = B_k - B_{k,\text{ideal}}\,. \tag{35}$$

Note that this error is positive if we have produced too many bits. In practice, this bit-production error is calculated recursively by updating the previous bit-production error Δ_{k-1} with the target number of bits $b_{k-1,\text{ideal}}$ and the actually produced number of bits b_{k-1} for picture $k - 1$:

$$\Delta_k = \Delta_{k-1} + b_{k-1} - b_{k-1,\text{ideal}}\,. \tag{36}$$

On start-up we set the condition that there is no production mismatch for the first picture $k = 0$:

$$\Delta_0 = 0. \tag{37}$$

We control the error by adjusting the ideal target number of bits $b_{k,\text{ideal}}$, and set the new target quantization scale for the picture k to

$$\tilde{Q}_k = \frac{c_k}{b_{k,\text{ideal}} - \alpha_k \Delta_k}, \tag{38}$$

where the denominator of Equation (38) effectively represents the adjusted target number of bits we set for picture k. The parameter α_k determines the rate of control by compensating the ideal bit budget for picture k with a portion of the error. By using

$$c_k = Q_{k,\text{ideal}} b_{k,\text{ideal}}\,, \tag{39}$$

Equation (38) can be rewritten as

$$Q_k = \left(\frac{1}{1 - \dfrac{\alpha_k}{b_{k,\text{ideal}}}\Delta_k}\right) Q_{k,\text{ideal}}\,. \tag{40}$$

For different pictures to have the same adjustment in quantization scale for the same Δ_k, we make α_k proportional to $b_{k,\text{ideal}}$. This means that each picture corrects for the bit-production error, but proportionally to its own targeted size. We thus set

$$\alpha_k = \alpha b_{k,\text{ideal}}\,, \tag{41}$$

yielding

$$Q_k = \left(\frac{1}{1 - \alpha\Delta_k}\right) Q_{k,\text{ideal}}\,. \tag{42}$$

Of course, we still have

$$Q_{\min} \le Q \le Q_{\max}\,, \tag{43}$$

which guarantees that the values stay within bounds. Finally, we choose α such that an accumulated bit-production error of, for example, one full VBV buffer size $B_{\max}$ results in halving the target bit count. This means that we set

$$\alpha = \frac{1}{2B_{\max}}. \tag{44}$$

Note that this method is *not* symmetric with respect to the sign of the bit-production error, since an underproduction of $B_{\max}$ bits would result in multiplying the target bit count by only 3/2.

6. Refinements

- *First-pass parameter "filtering"*

As was described in Section 3, the picture target bits for the second pass can be based directly upon the first-pass quantization scale settings. However, at first-pass run time, these values were set on the basis of very limited knowledge of the future, and are not necessarily the best that could have been set. Particularly in special situations, such as immediately following scene changes or during fades, when the scene characteristics are changing in an unpredictable manner, the first-pass rate control can temporarily become unstable. In certain rare circumstances, the first-pass quantization scale settings can even be

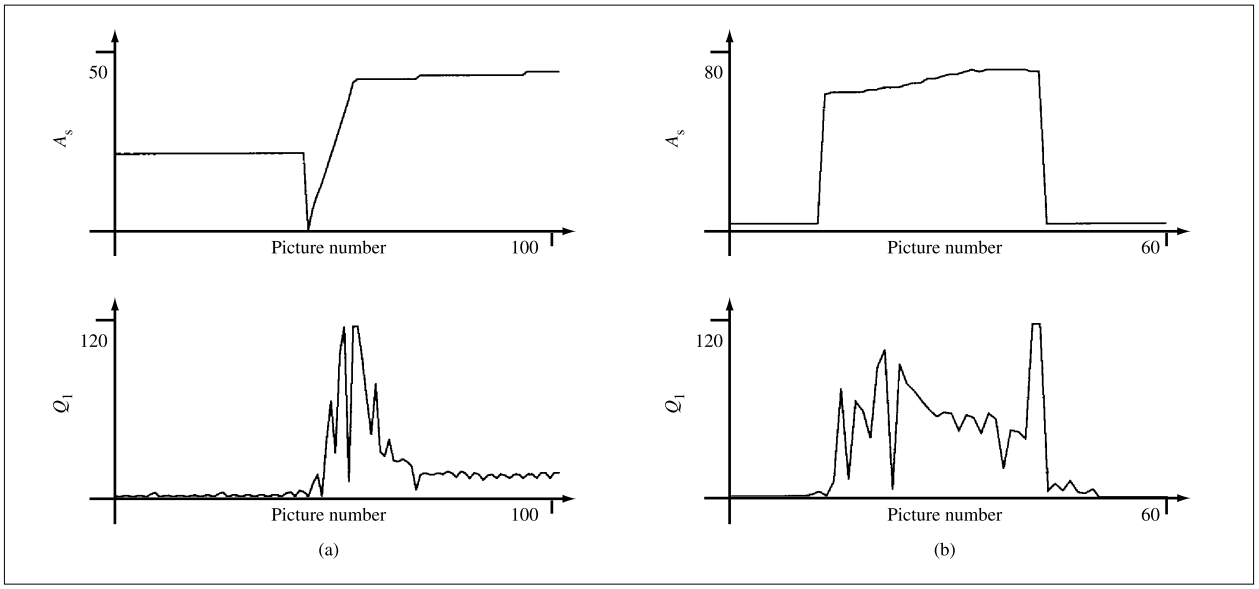

Figure 11

Average picture activity A_s (top) and first-pass quantization scale settings Q_1 (bottom) during a fade (a) and scene changes (b).

so far off that poor visual quality is the result. An example of this is shown in **Figure 11**, together with the average activity, to illustrate the change in scene characteristics. In this case we are dealing respectively with a fade from black and a scene change. Clearly, the quantization scale values are much too high, and the resulting picture quality during the fade and after the scene change is very poor. This is verified by looking at the corresponding video clips. Therefore, in the second pass we should not base our targets directly on these values.

After running the first pass we know the characteristics of each picture in the entire sequence. Had we known these in the first pass, we would not have set the erroneous quantization scale values as we did in the examples of Figure 11. Therefore, if we run a second-pass CBR, we can produce quantization scale settings that we can use to set the second-pass quantization scale and avoid the previously mentioned problems. In other words, we can use the first-pass information to calculate the quantization scale settings as if we are going to run a second-pass CBR. We can then use those values to set the second-pass VBR bit targets. By doing that, we essentially correct for the first-pass errors. This results in cleaned-up, or filtered, first-pass quantization scales suitable for reliably computing the second-pass quantization scale settings.

The second-pass CBR algorithm we use is identical to the one that was executed for the first pass. A constant quantization scale is set for each GOP according to the number of bits available for each GOP. However, assuming that we now know where the scene cuts lie, we should not set a constant quantization scale across one. Instead, we break up a GOP that contains a scene cut in the middle; see also the next subsection. So, in general, we divide a sequence into time intervals, each with a certain number of pictures. As mentioned, such an interval is usually a GOP, starting with an I-picture (in coding order), but this may be different, for example, around scene changes. We then set a constant quantization scale for each interval such that the available number of bits for that interval is exactly spent. For this we use the bit-production model parameters as they were collected from the first pass.

Using the first-pass bit-production model parameters, the interval quantization scale is set as follows. If the average bit rate is R_{ave} bits per second, and the picture rate is f_{pic} pictures per second, the average number of bits available per picture is

$$b_{\text{ave}} = \frac{R_{\text{ave}}}{f_{\text{pic}}}. \tag{45}$$

Suppose that interval I has N_I pictures, taking repeat-first-field into account. Then the number of bits available for this interval is

$$b_I = N_I b_{\text{ave}}. \tag{46}$$

When the bit-production model is used to relate this to a quantization scale, the total number of bits spent in this interval is also equal to the sum of bits over the pictures i in the interval:

$$b_I = \sum_{i=1}^{N_I} b_i = \sum_{i=1}^{N_I} \frac{c_i}{Q_i}. \tag{47}$$

Now, setting a constant quantization scale for this interval, we have

$$Q_I = \frac{1}{N_I b_{\text{ave}}} \sum_{i=1}^{N_I} c_i. \tag{48}$$

Finally, the corrected values for the number of bits and for the quantization scale for each picture i are

$$\begin{cases} \bar{b}_{1,i} = \dfrac{c_i}{Q_I}, \\ \bar{Q}_{1,i} = Q_I, \end{cases} \tag{49}$$

where Q_I is calculated as in Equation (48).

An example of the resulting second-pass CBR quantization scale targets is shown in **Figure 12**. It can be seen that the quantization scale values are now "more reasonable." To test the impact, we ran a second-pass CBR with these targets. As a result, the visual video quality remains stable during the problem areas, for example immediately following a scene change. This shows that we can more reliably use these quantization scale target settings to compute our second-pass VBR, instead of the raw, unprocessed first-pass results.

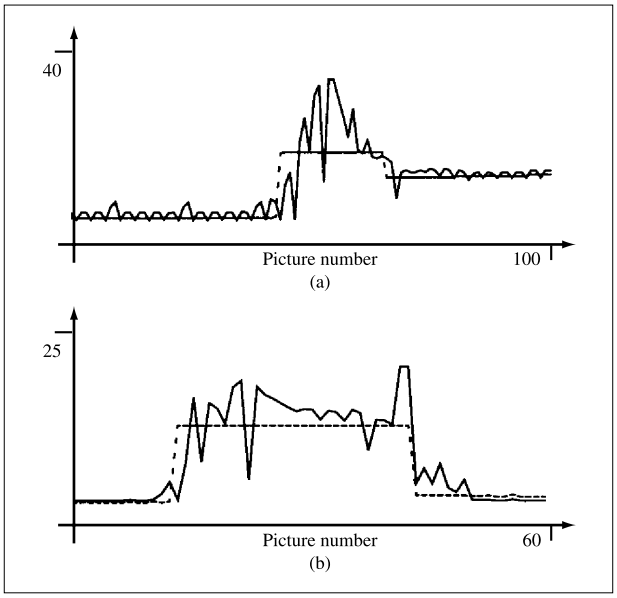

Figure 12

Target quantization scale settings with second-pass CBR quantization scale correction (dashed line), and without (solid line). Both correspond to the same examples as in Figure 11: (a) Fade; (b) scene changes.

• *Scene changes*

To detect scene changes, we use the average picture activity, such as that shown at the top of Figure 11. It is assumed that the data is in display order. A simple threshold function then detects scene changes by comparing the average activity of a picture with that of the previous one, in display order. Most scene changes that are obvious to human observers show a fairly large change in spatial activity, of the order of 40 and greater (see for example the top right plot in Figure 11, where the change is approximately 60). It is for these "obvious" scene changes that scene-change detection is most needed. Small changes in the spatial activity generally do not require scene-change detection, because the first-pass rate control does not become as unstable as with large changes. Such a relatively small change in spatial activity may be caused by a scene change, but also by motion (uncovering new background) and other changes in scene content. Currently, the scene-change detection threshold is set at 20, which was found to detect most scene changes, while the ones that are not detected with this threshold are generally not significant enough to require detection. Overall, the second-pass VBR results were not very sensitive to the exact value of this threshold, where varying the value from 15 to 25 gave almost identical results.

After detecting the scene changes, we determine the intervals for which we will set a constant target quantization scale to filter the first-pass quantization scales, as described in the preceding subsection. Normally an interval is equivalent to a GOP, but if a scene change happens to fall in the middle of a GOP, this breaks the GOP into two parts. We combine the first part of this split-up GOP with the previous GOP, and the second part with the next GOP. Thus, instead of having three GOPs as intervals, a scene change in the middle of a GOP yields only two intervals, which are both longer than a GOP. In doing this, special care is taken when two scene changes are very close to each other. Note that this operation is performed on the data in display order.

• *Fades*

Quite often, the transition between two different scenes is realized not with an abrupt scene cut, but with a fade. This may be done directly between two scenes, or the old scene may first be faded out toward a blank scene, followed by a period of blank pictures, and completed by

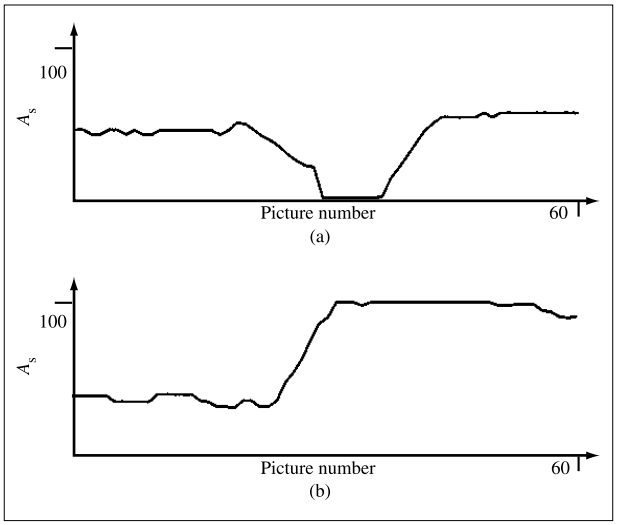

Figure 13

Spatial activities for a sequence of 60 pictures for two typical cases of a fade: (a) Fade-out to almost black, followed by fade-in to another scene. (b) Fade from one scene directly into another.

Table 3 Measured fade properties for the examples shown in Figure 13.

	Length	*Height*	*Slope*
Fade-out to almost black—Figure 13(a)	9	28	3.1
Fade-in from almost black—Figure 13(a)	8	52	6.5
Fade between two scenes—Figure 13(b)	10	75	7.5

a fade-in from the blank to the new scene. During fades, MPEG-2 encoders may have problems maintaining coding efficiency, especially if the fade is not very gradual and the pictures in the fade differ too much for efficient prediction. Practical motion estimators are generally not designed to match blocks with different luminances, though features may be similar. An estimator therefore has difficulty identifying the correct motion vectors during fades, resulting in large prediction errors and random motion fields. These random motion fields consume large quantities of bits. Especially for B-pictures, this may be disproportionally large compared to the total number of bits for the picture. Even if the correct motion vector is found, the prediction errors may be larger than normal, resulting in a need for more bits to encode at a certain quality.

It is possible to improve coding efficiency during fades in a second pass. One measure is to force zero motion vectors everywhere. We can thus free bits from coding the motion vectors, and use them for coding the prediction errors more accurately. Note that if there is very little motion, which is often the case at the beginning and end of a scene, we actually are using the "correct" motion vectors, which may help to reduce the visual artifacts. Setting motion vectors to zero is quite a drastic measure, so we wish to be sure that we apply it only to cases of clearly problematic fades.

Just as with scene-change detection, fades are identified using the picture spatial activities. Two examples of the activities during and around fades are shown in **Figure 13**. The plot shown in part (a) is for a fade-out to almost black, followed by a fade-in to another scene. The short, "almost black" scene was an all-blank scene, but with a small logo superimposed. Hence, the activity is not entirely zero. The plot shown in part (b) depicts a fade directly from one scene into the other. We thus formulate a fade as an increasing or decreasing sequence of spatial activities. Further, we can define certain properties of such a sequence, such as the length of the fade sequence, the height (the absolute spatial activity difference between beginning and end), and the ratio of these two, which is the slope.

Detecting a fade is implemented by first determining sequences of spatial activities for which the values keep on increasing or decreasing. This is done across all pictures in the entire video. After identifying a sequence of pictures with increasing or decreasing spatial activities, the properties of length, height, and slope are calculated. Each of these properties is then tested against a threshold. The thresholds were set such that not every incline (or decline) in spatial activity is detected as a fade. Only "strong" fades are to be detected.

The actual thresholds for the length, height, and slope were found by observing many long video sequences that contained several "true" fades, but also many effects that could trigger a false and unnecessary detection. A fade length of two pictures is by definition identical to a scene change, while a length of three pictures was generally found not to be a fade, but a detectable scene change or a change of scene content. Only when four or more pictures exhibited increasing or decreasing spatial activity behavior did the corresponding video sequence almost always show a fade. Further, it turns out that a height less than approximately 15 often does not correspond to a fade. For example, a camera pan may be the reason for this behavior. The same argument holds for the slope, which is the height divided by the length: A very long fade with a relatively small height is often not a fade but some other gradual effect. In summary, by observing many video sequences that contained fades and other effects, the thresholds that were found to reasonably extract only the relevant fades were a minimum length of four pictures, a minimum height of 15, and a minimum slope (height divided by length) of 2.5.

For example, the spatial activities from Figure 13 yielded the fade properties shown in **Table 3**. It can be seen that all three fades, the fade-out and fade-in of part (a) and the fade of part (b), were detected with these settings.

Experiments were carried out to determine the effect of forcing motion vectors to zero. Data for such an example of a fade is shown in **Figure 14(a)** for a scene that was slowly panning while fading out to black. The PSNR has been computed for conditions both with and without forced zero motion vectors. In both cases, the number of bits for each frame remained roughly the same, and for some frames the case of forced zero motion vectors even used fewer bits. The improvement in PSNR with the use of forced zero motion vectors is plotted in **Figure 14(b).** It can be seen that forcing the motion vectors to zero has a positive effect on the PSNR. Furthermore, visual inspection of the results shows that blocking effects are present when the estimated motion vectors are used, while these disappear when the vectors are forced to zero. This effect of improved PSNR, combined with an improved visual quality, has been observed in all fades that are detected and have the motion vectors forced to zero.

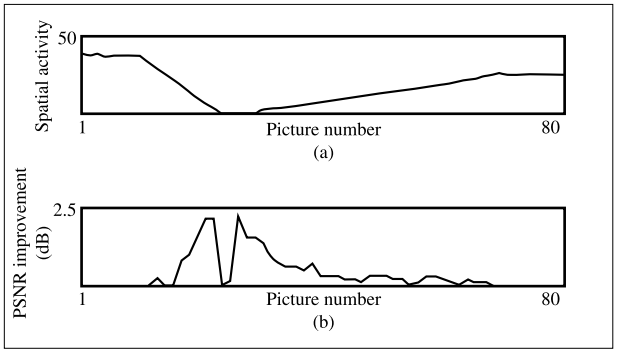

Figure 14

Effect of forcing motion vectors to zero during a fade: (a) Plot of spatial activities exhibiting the typical behavior of a fade-out followed by a fade-in. The fade-in is not steep enough to be detected, so no zero motion vectors were forced there. (b) Plot of improvement in PSNR with the use of forced zero motion vectors instead of estimated motion vectors.

7. System description

A schematic diagram of the MPEG-2 encoder–decoder system is shown in **Figure 15**. The MPEG-2 encoder consists of the IBM three-chip set, integrated on a PCI adapter card. Similarly, the MPEG-2 decoder is configured on another PCI card. Both the encoder and decoder cards are commercially available. The encoder takes its source video data from D1 tape, whereas the decoder outputs the decoded video to a monitor for display. The whole system is under the control of a personal computer. A special tape control manages starting and stopping of the D1 tape by using time codes to ensure synchronization of the first and the second passes.

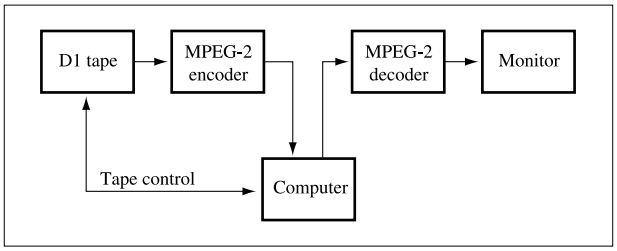

Figure 15

MPEG-2 encoder–decoder system.

A custom-made graphical user interface (GUI) provides full user access to the encoder–decoder system. The main screen of this GUI is shown in **Figure 16**. Among other things, the time-code control can be seen on the right of this window, where time codes for start and stop of the D1 tape can be entered. The video setup is done in a subwindow, which comes up when the "video" button on the main screen is clicked. This subscreen is shown in **Figure 17**. Various high-level MPEG-2 video parameters can be set there, such as the picture coding structure and the GOP size and structure. Low-level, more detailed MPEG-2 parameters can be set in yet another subwindow, which is opened by clicking on "Advanced." Also selectable in the video subwindow of Figure 17 is the video source, which can be composite, component (S-video), or D1. The system can take input from a 525-line system (NTSC) or a 625-line system (PAL). Finally, in

Figure 16

Main screen of the graphical user interface to control the IBM encoding system.

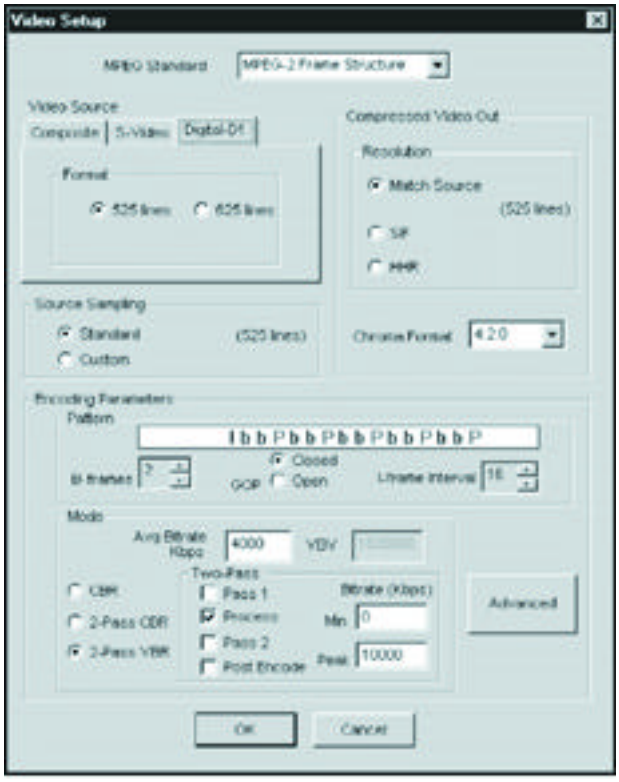

Figure 17

Video subscreen of the graphical user interface of the IBM encoding system.

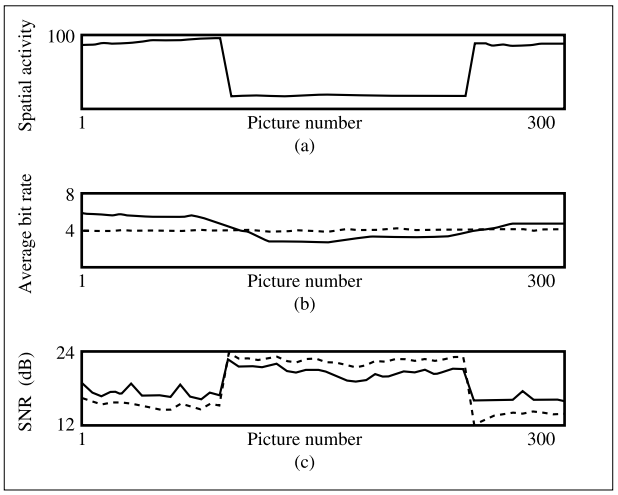

Figure 18

Spatial activities (a), average bit rates (b) and SNR values for 300 pictures from a longer sequence (c). The average bit rates and SNR values are shown for both CBR (dashed lines) and VBR (solid lines).

Figure 17 the mode can be selected (CBR, two-pass CBR, or two-pass VBR) with the corresponding parameters, such as maximum bit rate for VBR and the overall average bit rate. For the two-pass modes, it can further be selected whether to run a first pass, preprocessing, or a second pass.

8. Conclusions

In this paper a complete system for two-pass encoding of MPEG-2 video is described. First, a pass is made over an entire video sequence while extracting relevant video characteristics. These include the number of bits for each picture, quantization scale, picture type, picture spatial activity, etc. A special preprocessing program then uses these first-pass statistics to prepare second-pass control parameters, such as the individual picture target number of bits. Finally, a second pass is made through the video sequence, but under control of the prepared parameters and a special real-time bit-production monitoring and adjustment algorithm. This second-pass control is on a picture-by-picture basis.

Owing to the limited capabilities of the first-pass CBR, some segments of the raw statistics were found not suitable for immediate use by the preprocessing. Instead, a prefiltering operation is required, including scene change and fade detection, and recalculation of the first-pass quantization scales. It has been shown that this filtering significantly aided the parameter preparation for second-pass VBR, especially in certain situations, such as immediately following scene cuts.

Methods have been developed and implemented to ensure that an MPEG-2-compliant bitstream is generated. This implies that buffer underflow and overflow are anticipated and prevented at all times. Note that these measures take effect only when the bit rate is close to the maximum rate for VBR, and when the video sequence is extremely unpredictable and irregular.

The VBR objective has been defined as constant visual quality. This led to performing visual perception experiments in order to determine what constitutes such a constant quality for the human observer. By its very nature, this makes the final VBR optimization criterion subjective. However, it is possible to define certain qualitative methods that meet the VBR objective. The derived methods have been implemented, and many experiments have been run to verify the validity and performance on a variety of different video sequences.

Figure 18 shows an example of 300 pictures from a longer sequence when the second pass is run in CBR and in VBR mode. It can be seen that whereas the CBR runs at a constant bit rate, in VBR mode the bit rate may vary considerably. The total bit budget for both the CBR and VBR was the same, 4 Mb/s. The SNR values are also plotted in Figure 18, showing that the SNR takes the

opposite track from the bit rate, as expected, but that by no means does constant visual quality imply a constant SNR.

The general conclusion is that the second-pass VBR sequences visually appear to have a higher overall quality than the ones coded with CBR. For VBR to visually outperform CBR, a mix of "easy" scenes and "difficult" scenes is always required. If all scenes were the same (easy or difficult), the VBR results would be equal to those for CBR. The principle of VBR relies on taking bits from easy scenes and spending them on the difficult ones instead. In visual evaluations by different viewers, it was found that the visual quality of an entire video sequence is judged by the minimum quality across the whole sequence. This minimum quality is usually found in the easiest scenes, as coding artifacts are the most noticeable there. A good VBR performance is thus founded on removing bits safely from the easier scenes (i.e., without noticeably distorting their quality). These bits are then redistributed over the more difficult scenes, such that the entire video is perceived to have a constant quality.

References

1. "Information Technology—Generic Coding of Moving Pictures and Associated Audio Information: Video," First Edition, *ISO/IEC 13818-2*, May 1996.
2. D. T. Hoang, J. S. Vitter, and E. L. Linzer, "Lexicographic Bit Allocation for MPEG Video Coding," *Proceedings of the IEEE International Conference on Image Processing (ICIP '97)*, Santa Barbara, CA, October 26–29, 1997, Vol. 1, pp. 322–325.
3. L. Teixeira and H. Ribeiro, "Analysis of a Two Step MPEG Video System," *Proceedings of the IEEE International Conference on Image Processing (ICIP '97)*, Santa Barbara, CA, October 26–29, 1997, Vol. 1, pp. 350–352.
4. "Final Text of ISO/IEC FCD 14496-2: Visual," *ISO/IEC JTC1/SC29/WG11 N2202*, April 1998.
5. "Test Model 5," *ISO/IEC JTC1/SC29/WG11 N0400*, April 1993.

Received May 14, 1998; accepted for publication June 3, 1999

Peter H. Westerink *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (peterw@us.ibm.com).*

Dr. Westerink received his Ph.D. from the Delft University of Technology, Netherlands, in 1989. He then joined the AT&T Bell Laboratories, Murray Hill, New Jersey, where he was a member of an HDTV data compression algorithm development team. Leaving AT&T at the end of 1991, he joined the Matsushita Applied Research Laboratory in Burlington, New Jersey. This broadened his interest and scope toward practical studio and broadcast TV applications. The work involved the development of digital camera lens error-correction algorithms, edge detection, hierarchical shape coding, and segmentation of moving video. Also researched were methods for high-density field motion estimation for frame rate conversion, super slow motion, and interlaced-to-progressive conversion. In 1995 Dr. Westerink joined the IBM Thomas J. Watson Research Center to work on MPEG-2, supporting and innovating the functionality of the IBM MPEG-2 encoder system. Currently, his attention has shifted toward the MPEG-4 standard. His interests include video coding aspects, MPEG-4 systems, multiplexing, packet scheduling, rate control, and scene composition (VRML). He is currently the co-chair of one of the MPEG-4 subgroups, a member of the IEEE IMDSP committee, and the Finance Chair for the International Conference on Multimedia and Expo 2000. Dr. Westerink's general interests include video coding, rate control, computer graphics, Java multimedia and real-time issues, and interactive multimedia authoring and delivery systems.

Rajesh Rajagopalan *Lucent Technologies, 600 Mountain Avenue, Room 2A130, Murray Hill, New Jersey 07974.*

Dr. Rajagopalan received the Bachelor of Engineering degree in electronics and communication engineering in 1986 from the Regional Engineering College, Suratkal, India, and the Master of Science degree from the University of Texas at Austin in 1993 and the Ph.D. degree from the University of Illinois at Urbana–Champaign in 1996, both in electrical and computer engineering. He has been a Research Assistant in the Electrical Engineering departments of the University of Texas at Austin and the University of Illinois at Urbana–Champaign; a Research Associate at the Institute for Geophysics, Austin, Texas; a visiting student at Princeton University; and a Research Staff Member at the IBM Thomas J. Watson Research Center at various times from 1990 through 1997, working in the areas of signal, image, and video processing. He is currently a Member of Technical Staff at Lucent Technologies, working in the field of digital video. His areas of interest include video processing, distribution, and communications.

Cesar A. Gonzales *IBM Thomas J. Watson Research Center, Yorktown Heights, New York 10598 (butron @us.ibm.com).*

Dr. Gonzales is an IBM Fellow, Senior Manager for Multimedia Technologies at the IBM Thomas J. Watson Research Center, and manager of the development organization of the IBM Digital Video Product Group He is an expert in image and video processing and compression; his experience spans the development of algorithms, chip and system architectures, and multimedia applications. He is a co-inventor of various still-frame and motion video compression techniques that IBM contributed to the JPEG and MPEG international standards. He is the author of number of patents and publications related to these techniques. While at IBM, he has received multiple Outstanding Achievement awards, including a Corporate-level award for his contributions to IBM's MPEG products. In June

1998 Dr. Gonzales was named an IBM Fellow. He was also elected to serve in IBM's Academy of Technology, and has twice been named Master Inventor for his contributions to IBM's patent portfolio. Prior to coming to IBM, Dr. Gonzales worked on radar signal processing and physics of the ionosphere at the Arecibo Observatory in Puerto Rico. Dr. Gonzales is a Senior Member of the IEEE. He has served as an Associate Editor of the *IEEE Transactions for Circuits and Systems for Video Technology*. He has also served as the head of the U.S. delegation in the MPEG committee of the International Standards Organization. Dr. Gonzales received his B.S. and engineering degrees from the University of Engineering (UNI) in Peru and his Ph.D. from Cornell University, all in electrical engineering.