JOURNAL OF VISUAL COMMUNICATION AND IMAGE REPRESENTATION
Vol. 8, No. 1, March, pp. 51–67, 1997
ARTICLE NO. VC970329

Adaptive Piecewise Linear Bits Estimation Model for MPEG Based Video Coding

Jia-Bao Cheng and Hsueh-Ming Hang

Department of Electronics Engineering, National Chiao Tung University, Hsinchu, Taiwan 300, Republic of China

Received November 6, 1995; revised July 9, 1996

In many video compression applications, it is essential to control precisely the bit rate produced by the encoder. One critical element in a bits/buffer control algorithm is the bits model that predicts the number of compressed bits when a certain quantization stepsize is used. In this paper, we propose an adaptive piecewise linear bits estimation model with a tree structure. Each node in the tree is associated with a linear relationship between the compressed bits and the activity measure divided by stepsize. The parameters in this relationship are adjusted by the least mean squares algorithm. The effectiveness of this algorithm is demonstrated by an example of digital VCR application. Simulation results indicate that this bits model has a fast adaptation speed even during scene changes. Compared to the bits model derived from training data based on cluster analysis, the adaptive piecewise linear bits model achieves about the same high performance with a much lower complexity and high self-adaptativity. A particular advantage of a rate control scheme employing a bits model over the buffer-feedback rate control such as MPEG2 Test Model 5 is that it can control the bits of every microblock very precisely. © 1997 Academic Press

1. INTRODUCTION

In many video compression applications, it is essential to control precisely the bit rate produced by the encoder, for example, in the cases of constant rate video transmission and storage. An application described at the end of this paper is digital video cassette recording (DVCR). To meet the variable speed playback requirement and the VCR mechanical restrictions, the compressed data of specific frames have to be placed inside predefined areas. Therefore, the coded bits generated by the encoder for each picture frame have to match the pre-assigned bit number.

In a typical motion-compensated transform coding scheme such as *Reference Model 8* (RM8) used by CCITT expert group and *Simulation Model 3* (SM3) used by ISO motion picture expert group (MPEG) in defining video compression standards [1, 2], their output bit rate is controlled by adjusting quantizer stepsize to avoid buffer overflow and underflow. There are two popular types of rate control algorithms. In the first approach, the proper quantization stepsizes is selected based on the buffer status. That is, the quantization stepsize is set small when the buffer is nearly empty and the stepsize is set large when the buffer is nearly full. However, this type of control schemes such as RM8 and SM3 often result in non-uniform picture quality because (1) it does not have a bit budget plan that distributes bits appropriately for every image block and (2) it does not take image contents into consideration in adjusting stepsize. Without a bit budget plan we may produce too many bits at times; then, we have to increase stepsizes abruptly to reduce output bits. In the second case, smaller stepsizes should be used in the smooth image region and larger stepsizes in the texture region to match the sensitivity thresholding characteristics of human perception.

A modified version of the above scheme is to precompute a budget plan for each picture frame and/or every image block. Then, quantization stepsizes are "guessed" based on the buffer status to produce the desired bits. One example of this kind of schemes is *Test Model 5* (TM5) in the development of MPEG2 standards [3]. However, the lack of an explicit bits model leads to a number of bit-rate dependent parameters inside this scheme. Its bit control, bits produced for each block, is quite imprecise.

The second type of rate control approach employs an explicit bits model that describes the relationship between bits and quantization stepsize. This bits model is able to predict the compressed bits when a certain quantization stepsize is in use before the real quantization and variable word-length coding (VLC) operations are actually applied to. In order achieve a uniform perceptual picture quality, we preanalyze the entire image content and allocate bits accordingly. A bit budget plan is thus obtained. If the predicted bit number using a selected stepsize does not match the planned bit budget, the selected stepsize has to be altered until it matches. When the bits model is accurate, the actual coded bits number is close to the predicted one and the bit control can thus be quite precise. Consequently, we can encode the pictures according to the planned bit

1047-3203/97 $25.00

budget to produce nearly uniform image quality and, at the same time, buffer overflow and underflow would be eliminated.

What we have just described is a *one-pass* coding structure. That is, the bit budget or stepsize plan is precomputed and then the actual quantization and VLC operations are executed once only. In contrast, we may start with a rough guess of the stepsize and run quantization and VLC to check the output bits. If the result is not satisfactory, we modify the stepsize and rerun quantization and VLC again [4]. This *multiple-pass* structure may be used for off-line simulations; however, it is too expensive to be used in real-time machines.

There are two methods to construct a bits model. The first method is to derive a mathematical model analytically based on the information theory; one example of using this method is the model proposed by Chen and Hang [5]. The other method is to derive an experimental expression based on test data. Because the quantizer and VLC operations are nonlinear and the picture content is time-varying, it is rather difficult to come up with an accurate and self-adaptive bits estimation model relying only on theoretical analysis. We take the experimental approach in this paper. A few one-pass experimental models have been reported. Puri and Aravind [6] suggested a look-up table that records the average coded bits of the training pictures. A simple first-order polynomial model was proposed by Sun *et al.* [7]. Though both schemes perform reasonably well on controlling the output buffer level, their performance degrades on the test pictures with characteristics different from that of the training pictures.

Here, we propose an adaptive piecewise linear bits estimation model, whose structure is borrowed from the tree-structured piecewise linear filter in [8, 9]. Each node in the tree is associated with a linear relationship between the coded bits and the image activity measure divided by stepsize. The parameters in this relationship can be adjusted by the least mean squares (LMS) algorithm. Simulation results demonstrate that this bits model has a fast adaptation speed even during scene changes. In addition, compared to the nonadaptive bits model derived from data using cluster analysis (in Section 5, an improved version of [6]), the adaptive piecewise linear bits model has a much lower complexity but achieves about the same high performance. Also, the model derived using clustering analysis has difficulties in catching up the variation of data promptly.

This paper is organized as follows. The activity function used to measure the image complexity is developed in Section 2. Also included is the construction of the constant coefficient linear bits model. Section 3 describes the proposed adaptive piecewise linear bits model. Then, a simple bits allocation scheme is designed in Section 4 to test the effectiveness of the proposed model in a complete coding system. Section 5 presents computer simulation results which demonstrate the advantages of the proposed bits estimation model. A cluster analysis based approach is implemented for comparison. Also included is a simulation using TM5 (of MPEG2), which produces less desirable results. At the end, a brief summary in Section 6 concludes this paper.

2. ACTIVITY MEASURE AND CONSTANT LINEAR MODEL

The image activity function is a measurement of the image content complexity; a high activity value indicates a hard-to-compress image (block). Several types of activity functions have been proposed. Although the block variance seems to be a popular activity measure ([6]), it was reported that among the following four activity measures, the one using the AC coefficient absolute values is most accurate [7]. These four tested activity measures are:

1. the variance of all the DCT coefficients (same as the block variance),
2. the sum of all the DCT coefficient absolute values,
3. the variance of DCT coefficients without DC term, and
4. the sum of DCT coefficient absolute values without DC term.

Figure 1 is the experimental results of each measurement at a fixed quantization stepsize on a picture frame of *Flowergarden* sequence (CCIR601 4:2:2 format, 480 lines by 720 pels). Each point in these figures represents the coded bits and the activity value associated with a certain macroblock (MB, a 16×16 image block defined in MPEG). Note that the relationship between the coded bits and the activity value for the fourth measure is well approximated by a straight line. Hence, the fourth measure is adopted as the activity measure in the rest of the paper. Mathematically, it is defined

$$ACT = \sum |\text{AC coefficient}|.$$

On the other hand, given a fixed picture block, the coded bits for different quantization stepsizes are plotted in Fig. 2. The curve shows that the bit number is inversely proportional to the stepsize. Combining the above experimental results, we conclude that the number of coded bits per (macro)block is approximately proportional to the activity measure and is inversely proportional to the quantization stepsize. An empirical first-order bits model is thus derived [7]:

$$\widetilde{BIT} = m \frac{ACT}{Q} + n,$$

where $\widetilde{BIT}$ is the estimated coded bits, ACT is the activity measure, Q is the quantization stepsize, and m, n are two

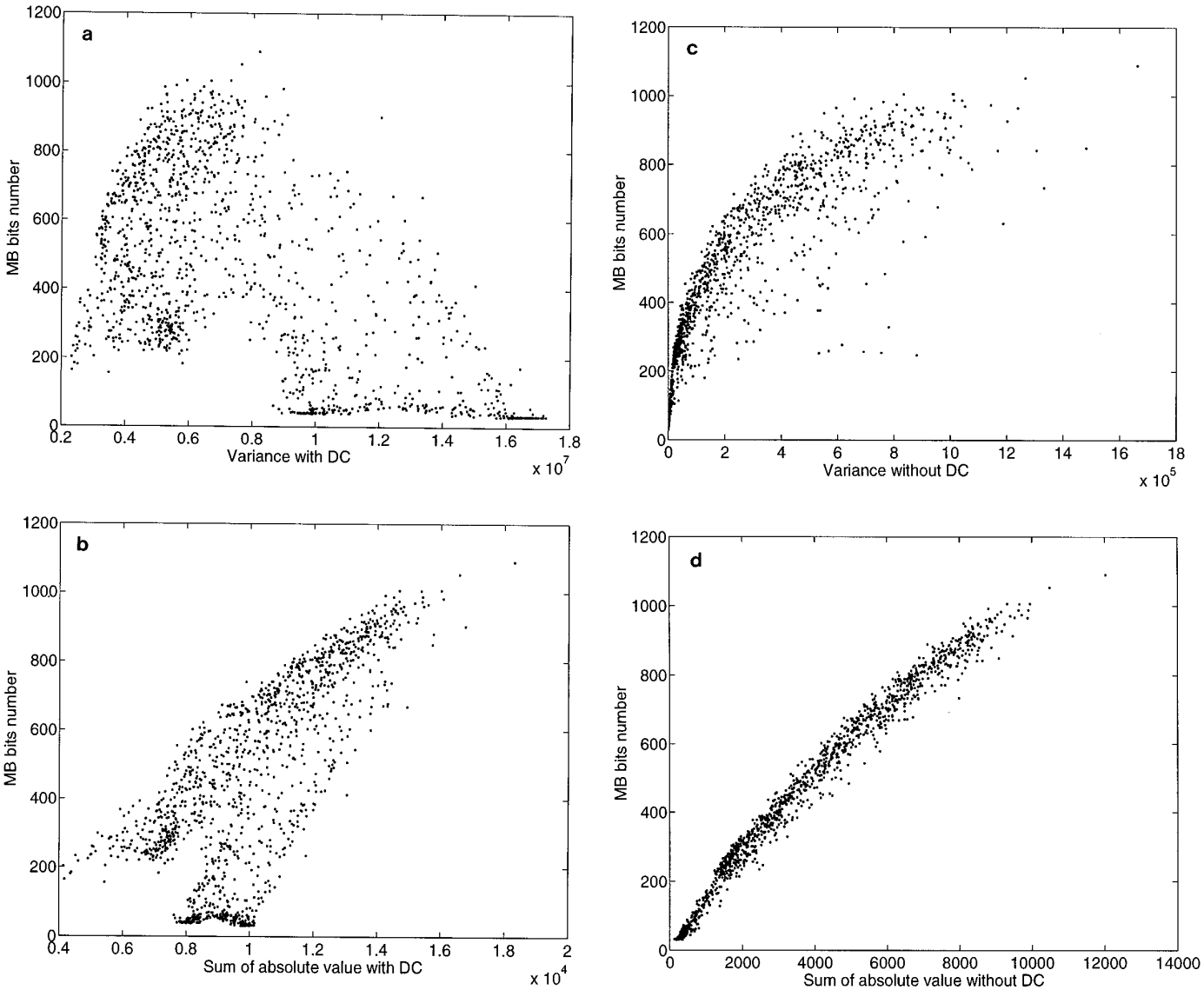

FIG. 1. (a) Coded bits versus variance with DC. (b) Coded bits versus absolute value with DC. (c) Coded bits versus variance without DC. (d) Coded bits versus absolute value without DC.

constants derived from training data to minimize a selected error criterion.

Typically, we choose m and n in (0) to minimize the mean square error, $E[(BIT - \widetilde{BIT})^2]$, based on the data triplets, (BIT, ACT, Q)'s. Each data triplet, (BIT, ACT, Q), represents the number of coded bits, the activity value, and the quantization stepsize of an image macroblock produced by the MPEG compression algorithm. For a test picture, the above minimization process has the

$$\min_{m,n} \sum_i \sum_j [BIT(j;i) - \widetilde{BIT}(j;i)]^2 = \min_{m,n} \sum_i \sum_j \left[BIT(j;i) - \left(m \frac{ACT(j)}{Q_i} + n \right) \right]^2, \quad (2)$$

where i is the index of legal quantization stepsizes and j is the index of macroblocks in one picture frame. According to the calculus of variation, (2) is equivalent to solving both

$$\frac{\partial \sum_i \sum_j (BIT - \widetilde{BIT})^2}{\partial m} = 0, \quad \frac{\partial \sum_i \sum_j (BIT - \widetilde{BIT})^2}{\partial n} = 0$$

under rather general conditions. Thus, m and n in (2) can be calculated by solving

$$\sum_{i,j} \left(\frac{ACT(j)}{Q_i} \right)^2 m + \sum_{i,j} \frac{ACT(j)}{Q_i} n = \sum_{i,j} BIT(j;i) \frac{ACT(j)}{Q_i}, \quad (3)$$

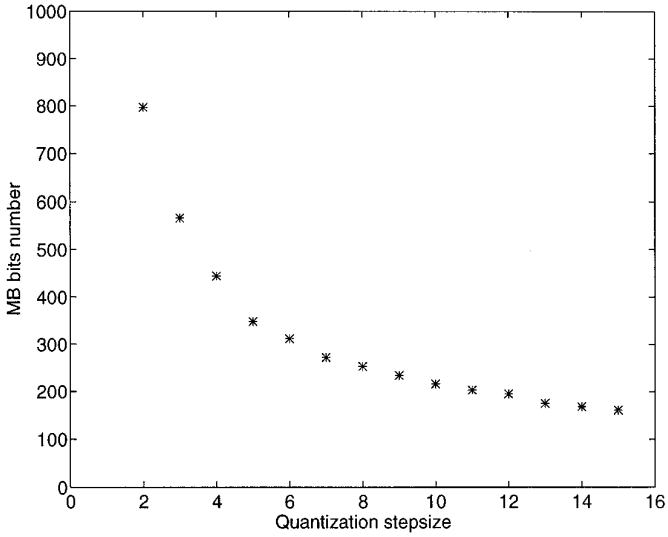

FIG. 2. Coded bits versus quantization stepsize.

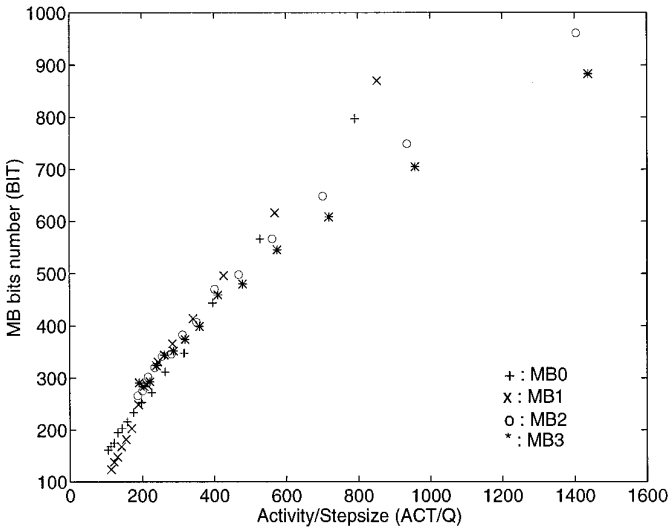

FIG. 4. Variation of the linear relationship for four neighboring macroblocks.

and

$$\sum_{i,j} \frac{ACT(j)}{Q_i} m + \sum_{i,j} n = \sum_{i,j} BIT(j;i). \qquad (4)$$

Figure 3 shows the fitness of the first-order bits estimation model for quantization stepsizes 2 to 10.

There are two drawbacks of this simple first-order model. One, the parameters, m and n, are picture-dependent and, two, the linear expression becomes less accurate when ACT/Q goes beyond a certain range. The first point can be observed from Fig. 4, where MB_i indicates the ith macroblock in an image frame. This problem becomes more serious for blocks belonging to pictures with different contents. The second point can be clearly observed from Fig. 3. These two drawbacks are often associated with a fixed empirical model derived from a set of training data. Either the model is tuned to a specific set of data so that it cannot be used for the other sets of data or the model is loosely fit into a large set of data so that it is inaccurate for a selected picture. Therefore, a more appropriate bits estimation model should be able to track the content variation of image sequence in a more precise manner. To improve the performance of the fixed first-order model, we need to develop an adaptive scheme that adjusts its parameters from time to time. In addition, instead of a single model that covers the entire range of data space, we partition the data space (ACT, Q) into segments and design appropriate parameter set for each segment separately.

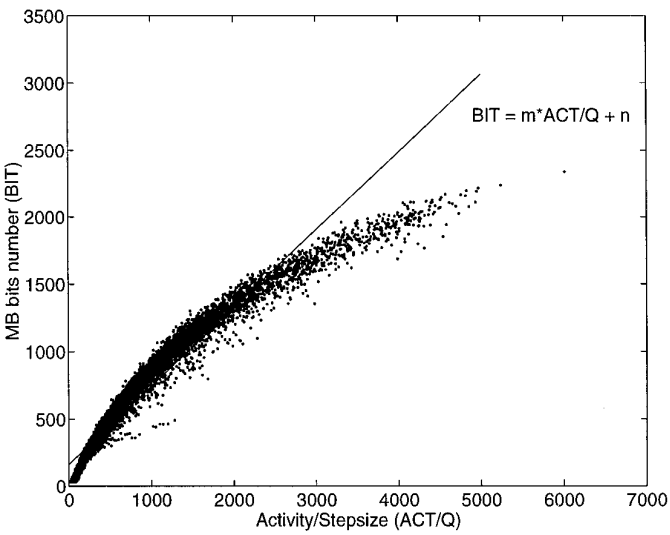

FIG. 3. First-order bit estimation model for Q = 2 to 10.

3. ADAPTIVE BITS ESTIMATION MODEL

In the previous section, we have discussed a linear bits estimation model and its weak points. In this section, an adaptive bits model is proposed based on the tree-structured piecewise linear filter structure. We first review the tree-structured filter which was designed for adaptive equalization in digital communication by Gelfand *et al.* [8, 9]. Our scheme for bits estimation application is then described.

3.1 Tree-Structured Piecewise Linear Filter

A nonlinear filter may be approximated by a piecewise linear filter in which the filter input domain is partitioned into disjoint regions and each linear filter operates only in its designated region. Figure 5 shows the structure of a piecewise linear filter [8, 9]. The input domain of this piecewise linear filter is divided into N regions denoted by

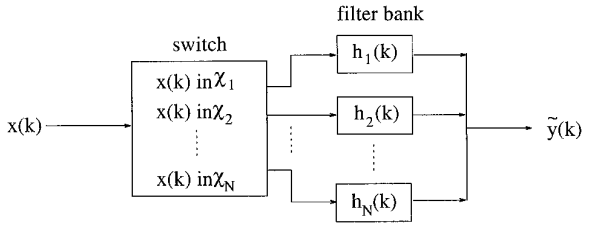

FIG. 5. Structure of a piecewise linear filter.

χ_i, $i = 1, 2, \ldots, N$. For each arriving input $x(k)$, the filter output $\tilde{y}(k)$ is the convolution of $x(k)$ and the impulse response $h_i(k)$, where the index i is the input region that $x(k)$ belongs to. If the input domain is partitioned using a sequential binary split, the entire piecewise linear filter can be organized as a tree. Each nonterminal tree node is associated with a linear filter and a threshold value; they are used to split the input domain. The filter coefficients and the threshold at each node are updated by the LMS algorithm. One advantage of this structure is that the standard linear filtering techniques can be employed at each node, for example, in finding the filter coefficients. Also, owing to its sequential and hierarchical partitioning structure, this approach is computationally efficient. It is reported that this filter structure has a good convergence speed compared to many other nonlinear adaptive filters. A typical example of the tree-structured piecewise linear filter is illustrated in Fig. 6a and explained below [8, 9].

To construct a tree-structured piecewise linear filter, we specify three elements for each node t in the *tree* T: a tap weight vector $\mathbf{c_t}$, an offset d_t, and a threshold θ_t.

Let $\mathbf{x}$ be an input data vector. Then node t is associated with a linear filter

$$\tilde{y}_t = \mathbf{c_t}'\mathbf{x} + d_t,$$

where $\tilde{y}_t$ is the filter output at node t. The final output of this piecewise linear filter is defined by

$$\tilde{y}_T = \tilde{y}_{t^*},$$

where t^* is the terminal node in the tree T obtained through the following process. We start from the root node and use the rule

$$\begin{aligned} &\tilde{y}_t > \theta_t, \quad \text{go to } r(t) \\ &\tilde{y}_t \le \theta_t, \quad \text{go to } l(t) \end{aligned} \tag{5}$$

where $r(t)$ is the right child stemming from node t and, similarly, $l(t)$, the left child. Therefore, each node in a tree corresponds to a filter with inputs restricted to a polygonal domain denoted by χ_t. In general, the filter output $\tilde{y}_t$ at node t is determined by the filter weight $\mathbf{c_t}$ and the offset d_t, whereas the domain χ_t is determined by the weights $\mathbf{c_s}$, offsets d_s, and thresholds θ_s at all the ancestor nodes s of node t.

The above point can be made more clear by examining the example shown in Fig. 6b [8, 9]. Note that there are three piecewise linear filters corresponding to the three possible pruned subtrees, namely $T_1 = \{1\}$, $T_2 = \{1, 2, 3\}$, and $T_3 = \{1, 2, 3, 4, 5\}$. The root node pruned subtree T_1 corresponds to a linear filter

$$\tilde{y}_{T_1} = \tilde{y}_1.$$

The pruned subtree T_2 with terminal nodes {2, 3} corresponds to a piecewise linear filter comprised of two linear filters restricted to two separate polygonal domains:

$$\tilde{y}_{T_2} = \begin{cases} \tilde{y}_2 & \text{if } \mathbf{x} \in \chi_2 \\ \tilde{y}_3 & \text{if } \mathbf{x} \in \chi_3. \end{cases}$$

The pruned subtree T_3 with terminal nodes {2, 4, 5} corresponds to a piecewise linear filter comprised of three filters restricted to three separate polygonal domains:

$$\tilde{y}_{T_3} = \begin{cases} \tilde{y}_2 & \text{if } \mathbf{x} \in \chi_2 \\ \tilde{y}_4 & \text{if } \mathbf{x} \in \chi_4 \\ \tilde{y}_5 & \text{if } \mathbf{x} \in \chi_5. \end{cases}$$

The above piecewise linear filter can be made adaptive by updating the filter input domain and the filter coefficients when new samples arrive. That is, the values of $\mathbf{c_t}$, d_t, and θ_t are adjusted by applying the least mean squares (LMS) algorithm to the input data sequentially. A thorough discussion of the tree-structured piecewise linear filter can be found in [8, 9]. The version designed particularly for bits estimation is described below.

3.2. *Adaptive Piecewise Linear Bits Estimation Model*

In the proposed adaptive piecewise linear bits estimation model, each node in the tree is associated with a first-order linear bits model restricted to a certain range of ACT/Q values. In other words, in our bits estimation model the filter output $\tilde{y}$ represents the estimated bits $\widetilde{BIT}$. The input vector $\mathbf{x}$ becomes ACT/Q (a scalar) and the coefficient vector $\mathbf{c}$ and the offset d are now model parameters m and n, respectively. Threshold θ is used to restrict the active range of the corresponding linear bits model. Our adaptive bits estimation algorithm is similar to the original adaptive filter algorithm in [8, 9] except for the initialization. To ensure a reasonable initial performance, each node in the tree begins with the same parameters that are derived offline using the constant coefficient first-order bits estimation model.

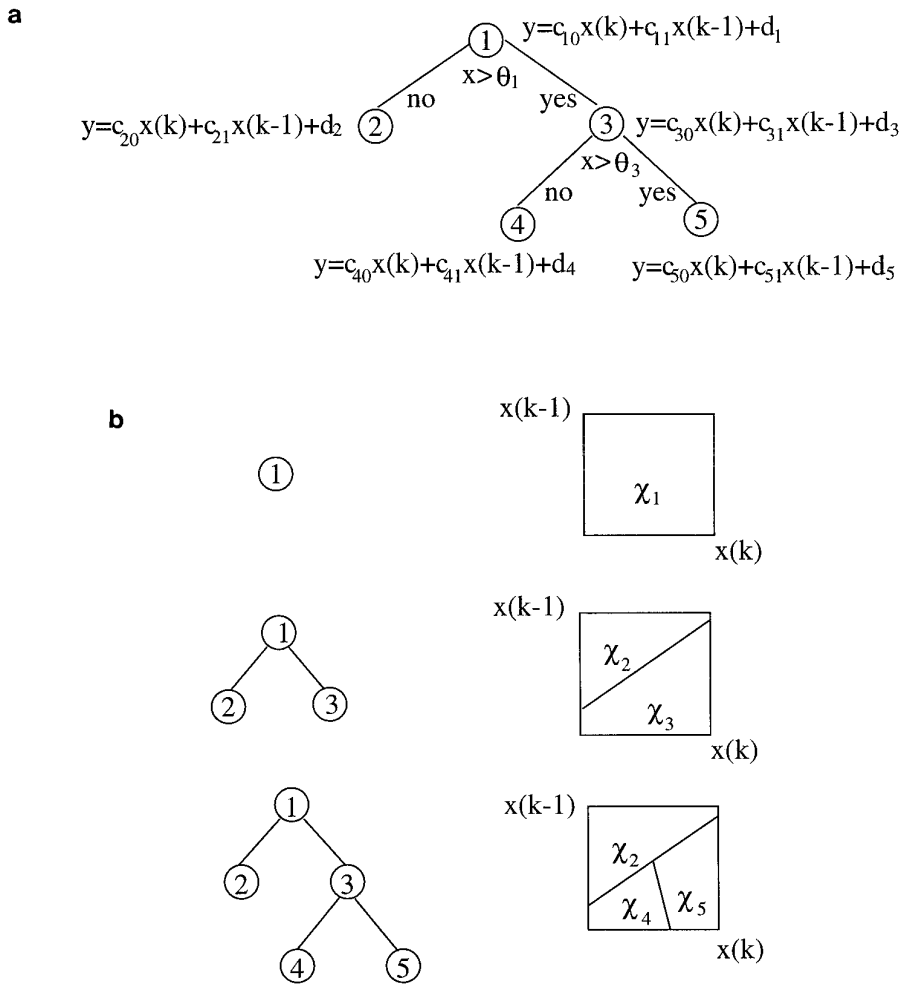

FIG. 6. (a) A tree-structured piecewise linear filter. (b) Pruned subtrees and their associated input domain partition.

The adaptive algorithm for piecewise linear bits estimation model is as follows.

⟨*Initialization*⟩

Let m_0 be the slope term and n_0 be the bias term of the initial linear bits model.

We initialize each node t in the tree T by

$$p_t(0) = \frac{1}{2^{\text{depth}(t)}},$$

$$m_t(0) = m_0, \quad n_t(0) = n_0, \quad \theta_t(0) = 0,$$

where p_t is the probability of the input domain associated with node t.

⟨Updating⟩

Let $(BIT(k), ACT/Q(k))$ be the $(k+1)$th arriving coded data pair. Assume $\widetilde{BIT}_t(k) = m_t(k)\ (ACT/Q)(k) + n_t(k)$.

Propagate the data sample from the root node to a terminal node of T according to the rule:

$$\widetilde{BIT}_t > \theta_t, \quad \text{go to } r(t)$$
$$\widetilde{BIT}_t \le \theta_t, \quad \text{go to } l(t).$$

If the data sample passes through node t, then its associated parameters are updated:

$$p_t(k+1) = p_t(k) + \mu(1 - p_t(k))$$

$$m_t(k+1) = m_t(k) + \frac{\mu}{p_t(k+1)}(BIT(k) - \widetilde{BIT}_t(k))\frac{ACT}{Q}(k)$$

$$n_t(k+1) = n_t(k) + \frac{\mu}{p_t(k+1)}(BIT(k) - \widetilde{BIT}_t(k))$$

$$\theta_t(k+1) = \theta_t(k) + \frac{\mu}{p_t(k+1)}(BIT(k) - \theta_t(k)),$$

where the parameter μ controls the convergence speed. If the data sample does not pass through node t, then the above parameters remain the same except that

$$p_t(k+1) = p_t(k) - \mu p_t(k).$$

4. ADAPTIVE BITS ALLOCATION

We now describe how the adaptive bits estimation model works together with bits allocation strategy. The task of bits allocation is to distribute bits to each macroblock properly so that the following two goals can be achieved: (1) the total coded bits should meet the bit budget; (2) the perceptual quality of every coded image block should be approximately equal. The above two requirements lead to the following two additional problems: (1) how to come up with an adequate bit budget for each picture frame; (2) how to decide the proper coded image perceptual quality for every block. We do not attempt to solve these two additional problems thoroughly here. But rather, for demonstration purposes, a simple yet quite effective bit budget planning scheme is devised. Our focus is that for a given bit budget and a given picture quality measure, we want to allocate bits to each block so that the total coded bits would match the preassigned bit budget.

In a MPEG coding scheme, there are three types of coded frames: *I-frame* (intra-coded frame), *P-frame* (predictive frame), and *B-frame* (bidirectional frame). The I-frames are independently coded and generally require the largest number of bits. The temporal redundancy in B-frames are estimated and removed using the already coded P-frame(s) and/or I-frame. Therefore, the B-frames generally require the fewest bits. An MPEG-coded sequence is partitioned into segments of pictures, called a *group of pictures* (GOP). Typically, a GOP contains an I-frame and several P- and B-frames.

Our bit budget planning strategy is as follows. Assume our GOP structure is fixed; that is, the numbers and positions of the P- and B-frames in one GOP are known and unchanged. For a given average bit rate per second, we thus know the bit budget of a GOP, which is called BIT_{GOP}. Then, the distribution of bits among the I-, P-, and B-frames are

$$BIT_{\mathrm{I}}: BIT_{\mathrm{P}}: BIT_{\mathrm{B}} = ACT_{\mathrm{I}}: ACT_{\mathrm{P}}: ACT_{\mathrm{B}}, \qquad (6)$$

where ACT_{I} is the total activity measure of the I-frame, and ACT_{P} and ACT_{B} are the frame activity measures of the P- and B-frames averaged over all the P- and B-frames in one GOP. In theory, we could precompute the activity measures of the entire GOP and then perform coding operations. However, to reduce implementation complexity, we simply use the previous GOP activity attributes for planning the bit budget of the next GOP. Even at scene changes, the variation of frame bit budgets is not drastic. Therefore, this *predictive* type frame budget planning scheme seems to work well in our simulations.

Once the frame bit budget is decided, we next decide the bit budget for every image macroblock inside a picture frame. In principle, we assign more bits to a macroblock of higher activity. The exact bits allocation strategy is described by

$$BIT(MB_0) + BIT(MB_1) + \cdots = BIT_{\mathrm{frame}} \qquad (7)$$

and

$$\begin{aligned} &BIT(MB_0): BIT(MB_1): \cdots \\ &\quad = \log(ACT(MB_0)): \log(ACT(MB_1)): \cdots, \end{aligned} \qquad (8)$$

where MB_i is the ith macroblock of a frame. Our experiments indicate that the log operation in (8) would lead to a more uniform perceptual image quality. Intuitively, $\log(ACT(MB_i))$ can be viewed as an information measure (in bits) of image content based on our activity function. Since the rate distortion function of a Gaussian source under mean square criterion has a similar log form [10], the choice of log operation seems to be reasonable. There are several studies on how to relate the human subjective criterion to the quantization stepsizes in DCT coding (see [11] and its references). However, since the focus of this paper is primarily on the bits estimation model, we adopt the simple bits assignment rule defined by (8).

Our bits (or quantizer) control algorithm consists of two steps: a *bits estimation step* and a *model updating step*. In the bits estimation step, we look for the most appropriate stepsize that generates coded bits closest to the given bit budget. For a given macroblock, we first calculate its activity measure (ACT). Then, we guess an initial stepsize value (Q). The ratio ACT/Q is the input to the piecewise linear bits model. This input, ACT/Q, propagates from the root node to a particular terminal node according to (5) and the estimated bit number is the output of the terminal node. If the estimated bit number does not match the preselected bit budget, the stepsize is increased or decreased (according to the bit difference) and the previous process is repeated. A flowchart describing this procedure is shown in Fig. 7. The best stepsize is the one that produces the estimated bits closest to the bit budget. Then we use the estimated stepsize to quantize the current macroblock. In the model updating step, the bit estimation error, which is the difference between the coded bits and the estimated bits, is used to update the piecewise linear bits estimation model. The model parameters are modified by the LMS algorithm described in the previous section. This completes the encoding of one macroblock as shown in the flowchart of Fig. 7. In the meantime, the activity measure of every block is collected and at the end of this GOP the frame total activity will be used to update the bit budget distribution for the next GOP.

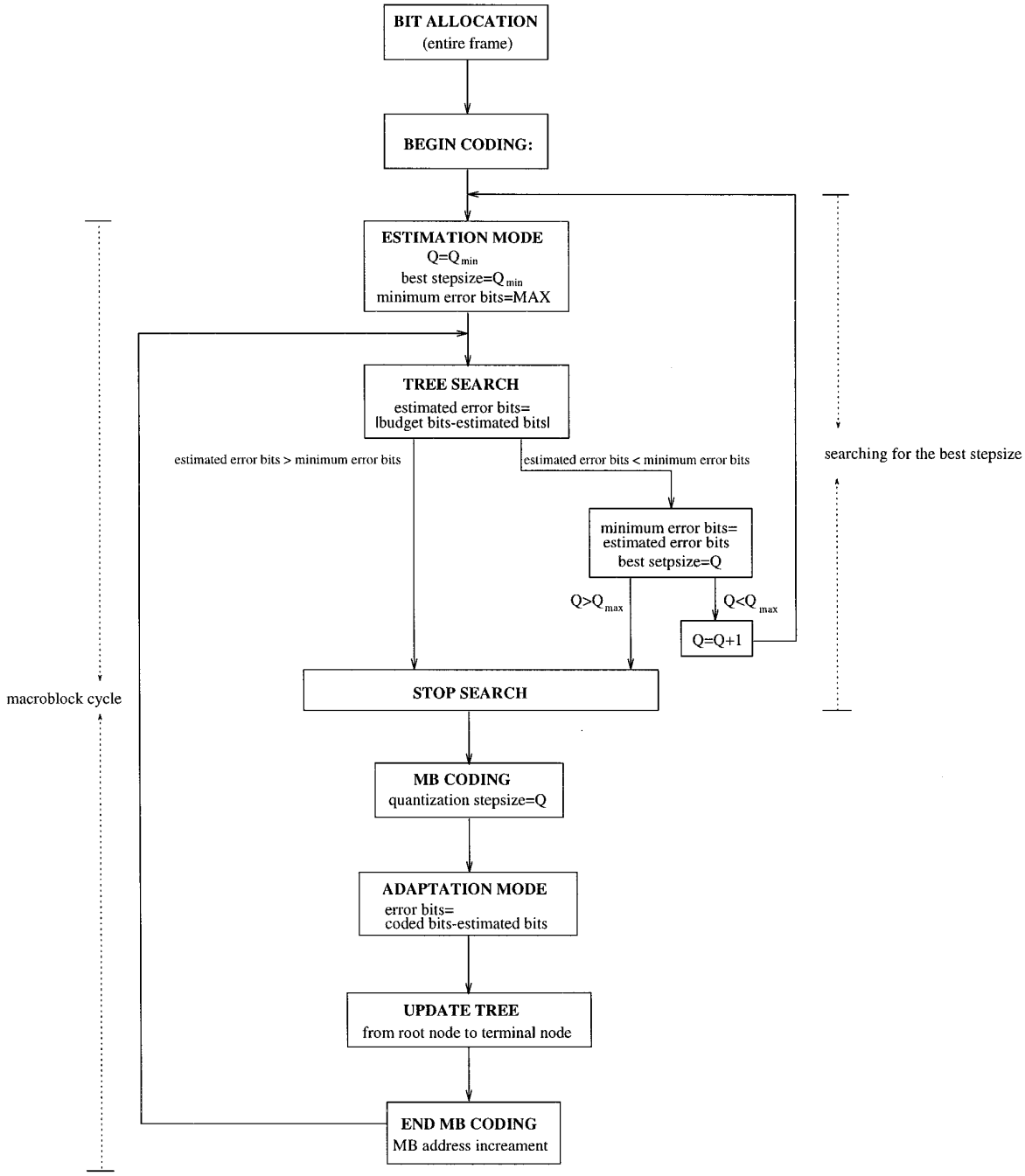

FIG. 7. Flowchart of bit estimation and model updating.

There are three main advantages of using the tree-structured filter structure. First, the piecewise linear filter is clearly superior to the single linear filter in solving nonlinear problems. Second, since the tree structure employs standard linear adaptive filtering techniques at each node, it is simpler than many other nonlinear adaptive filters, such as polynomial filters. Third, the most important feature of the tree-structured adaptive algorithm is that it provides "piecewise" adaptation. It means that when each training sample arrives, we update the parameters of the nodes that are located on a data-dependent path from the root node to a terminal node. As a result, only a small portion of nodes in the tree should be modified and the other nodes remain unchanged. Meanwhile, the updating process modifies slightly the domain of the corresponding filter and those of its neighbors. Hence, the whole process

is simple yet effective. Since the bits model is gradually dominated by the newly arrived coding macroblocks, the influence of the far-away blocks becomes less important. This also matches the locality characteristics of images.

5. SIMULATIONS

The target application of our computer simulations is digital video cassette recording (DVCR) [12, 13]. Because of the mechanical limitations (magnetic heads and tape movement) there are special requirements posed by DVCR system [12]. In order to implement variable speed playback and in the meanwhile retain a high compression efficiency, we have proposed a modified MPEG compression algorithm dedicated to DVCR application [13]. To enable fast playback, a short GOP containing only one I-, one B-, and one P-frames is chosen. According to the ''Consumer-use DVCR Specifications'' [14], the bite rate for the standard TV (CCIR601) DVCR is 25 Mbps. Assuming that the audio and the other digital data would need 2 Mbps, the video data bit rate is thus around 23 Mbps or 2.3 Mbits for each GOP. In this particular application, the coded bits have to be carefully controlled to match the exact bits assignment because data have to be placed at the exact location to facilitate the variable playback requirement (scanning heads read in data only on the regularly spaced specific locations). This application is an example used to demonstrate a potential use of our bits model and rate control algorithm. It can certainly be used for the other applications that require accurate bit rate control.

The bit budget of each macroblock is determined at the beginning of encoding a picture frame. Figure 8 shows a typical example of bit budget distribution. For the convenience of comparison, the original picture is split into macroblocks in accordance with the bit budget distribution. One may notice that the more complicated (high activity) macroblock is assigned more bits. However, since the log operation is used in (8), the bit budget increases slowly in the high activity regions. This may be justified from a perceptual viewpoint that coding errors are often less visible in high variance regions. From our experiments, the picture quality using log scale is better than that without it.

5.1. Constant versus Adaptive Models

Because the coding behavior is rather different for intracoded macroblocks (I-macroblocks), P-frame predictive-coded macroblocks (P-macroblocks), and B-frame predictive-coded macroblocks (B-macroblocks) in MPEG coding, three tree-structured bits estimation models are separately constructed. For comparison and initialization purposes, the constant coefficient first-order linear model is also derived from the data. A typical example of the constant coefficient model for the *Flowergarden* sequence is:

$$\widetilde{BIT} = 0.58\,\frac{ACT}{Q} + 164 \quad \text{for I-macroblock}$$

$$\widetilde{BIT} = 1.00\,\frac{ACT}{Q} - 38 \quad \text{for P-macroblock}$$

$$\widetilde{BIT} = 1.13\,\frac{ACT}{Q} - 92 \quad \text{for B-macroblock}$$

(A scaling factor may have to be inserted in applying the above formulas to different pictures.) In the adaptive model, the tree depth is three so that the corresponding piecewise linear bits estimation models have adequate complexity. Larger tree depths have been test without significant improvement. Before an image sequence is coded, the tree-structured bits estimation models are initialized using the constant linear bits models, which are obtained off-line from encoding the first three image frames. However, because of the adaptivity of the tree-structured bits model, any reasonable constant model can be used as the initial values. They all shortly converge to the model of the processed data.

In the following discussions, the *error bits* are defined as the difference between the coded bits and the estimated bits for each macroblock. Because P-frames are typically dominated by P-macroblocks and B-frames are dominated by B-macroblocks, for convenience, our statistics are performed on the entire picture rather than on different types of blocks. In Fig. 9, we first show the absolute error bits with and without adaptation for the I-frames in encoding the video sequence *Flowergarden*. The horizontal axis is the accumulated macroblock (MB) number. Since one picture contains 1350 MBs, the portion in Fig. 9 is the end of the second I-frame and the beginning of the third I-frame. They are the fourth and the seventh pictures in the original sequence. In the cases of ''without adaptation'' we simply use the constant estimation models described previously. It is clear from this figure that the adaptive piecewise linear bits estimation model decreases the error bits significantly. Similar performance improvement is obtained also for P-frames and B-frames (Figs. 10 and 11). The stepsize adjustment parameter μ for I-macroblocks, P-macroblocks, and B-macroblocks is 0.01, 0.001, and 0.001, respectively. These values are decided empirically to maintain a reasonable speed of convergence, typically around a few hundred macroblocks.

In Fig. 12, we demonstrate the adaptation ability of the proposed piecewise linear bits estimators when scene change occurs. In the middle of the test sequence (the 150th frame), the scene changes from *Flowergarden* to *Football*. The adaptive bits model is able to adjust its parameters rapidly to cope with picture variation, whereas the constant model has a significant higher average error for the new sequence. Therefore, suboptimal model param-

566 561 550 555 550 545 546 543 553 575 561 566 568 569 563 574 581 570 570 556 562 564 581 597 581 554 566 581 583 555
564 558 567 559 567 574 566 557 564 560 551 564 578 572 566 552 552 575 578 558 558 552 567 581 568 566 570 573 549 553
568 545 552 567 568 565 555 555 567 563 551 557 584 587 555 548 563 570 570 566 560 562 564 565 565 580 586 573 576 578
560 569 568 543 539 569 582 575 563 598 603 613 609 587 573 572 576 566 568 563 570 588 609 587 565 560 582 565 555 562
562 562 581 575 563 575 574 567 572 562 578 580 594 598 675 636 588 577 563 554 561 571 566 556 559 575 577 564 567 556
553 555 576 594 575 549 572 587 607 581 576 548 551 696 655 694 688 560 545 542 549 563 565 550 562 574 568 558 565 560
561 578 600 590 570 569 638 693 686 711 624 549 552 726 702 698 718 661 622 625 579 557 574 560 558 551 551 562 580 555
565 574 569 549 562 565 649 703 656 601 692 716 720 736 652 659 705 695 629 646 681 683 700 600 581 561 544 551 561 574
565 583 577 568 571 611 681 692 656 644 586 641 725 732 727 699 730 561 634 672 622 562 658 687 590 559 557 571 565 550
585 567 561 611 620 632 691 679 683 667 603 551 622 724 740 683 597 540 665 711 554 645 682 688 680 611 586 580 568 560
564 566 624 620 550 575 681 673 629 701 701 630 596 646 709 721 690 697 721 700 600 680 674 685 600 643 633 561 563 589
561 551 626 680 646 665 688 628 583 584 610 595 592 571 581 649 688 651 664 650 680 665 627 653 580 654 662 598 565 570
554 584 606 688 692 653 627 626 653 623 610 588 608 624 609 588 716 664 637 665 654 616 655 593 576 588 648 672 596 562
563 582 592 636 656 667 699 651 657 632 633 647 623 630 638 608 621 725 627 593 600 616 645 644 558 578 607 651 632 582
559 579 602 659 608 594 626 650 652 651 653 659 661 599 623 664 657 662 687 604 586 630 654 655 560 621 656 645 594 624
573 583 613 665 602 584 621 611 630 661 558 608 660 675 642 648 675 547 642 660 624 638 663 583 589 614 604 638 596 586
575 624 627 656 578 526 520 523 647 640 590 604 633 668 651 638 689 547 572 666 540 550 717 635 624 624 724 668 535 577
582 605 577 624 654 612 570 652 645 591 583 612 613 649 624 690 677 533 628 664 645 676 686 621 612 745 715 544 562 587
588 617 596 560 576 623 625 568 561 597 581 615 592 658 692 732 583 569 624 678 630 638 617 604 720 730 561 554 603 575
613 602 586 568 572 620 606 563 538 577 601 610 634 634 750 688 547 611 602 640 657 603 559 631 752 646 592 621 589 563
596 601 587 604 634 591 619 567 533 582 617 640 648 712 723 536 574 602 599 608 650 582 587 657 722 637 598 604 674 635
621 580 582 615 583 561 575 603 578 546 602 652 630 748 662 573 619 592 588 605 626 583 601 636 641 635 576 573 687 694
624 587 615 576 555 562 551 572 586 592 612 665 636 749 636 641 633 597 594 579 585 648 630 595 613 628 655 712 695 610
594 633 604 564 588 588 575 574 604 605 613 631 629 714 649 677 708 665 648 595 592 613 659 588 593 648 631 657 671 706
641 621 582 576 589 588 572 561 604 595 646 628 594 632 652 656 605 700 656 589 602 609 589 651 688 668 640 621 644 656
625 578 571 579 578 594 566 579 615 627 677 657 667 583 634 636 624 589 697 679 626 639 592 596 613 602 663 605 579 630
568 560 577 579 572 556 573 596 613 634 650 650 615 573 610 629 642 624 610 711 695 633 599 574 617 642 658 666 603 597
565 567 570 563 557 555 583 592 625 674 625 651 621 593 593 592 618 668 656 697 733 646 658 607 603 651 690 675 603 575
577 569 566 562 570 570 578 600 630 673 648 623 621 594 585 589 626 666 709 721 695 654 643 638 627 667 636 620 574 575
570 589 580 561 569 580 595 594 592 600 602 574 576 578 571 558 628 700 703 659 649 614 595 594 592 594 600 581 586 588

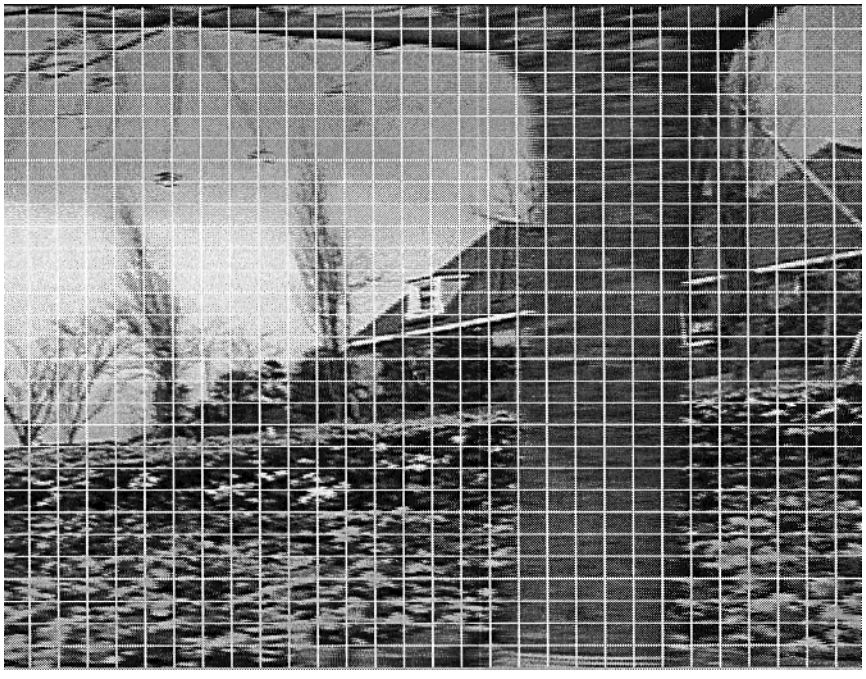

FIG. 8. An example of bit budget distribution.

eters can be used as the initial values of this piecewise linear model without degrading its long-term performance. Table 1 is the average absolute error bits per macroblock with and without adaptation. For example, in encoding the third image sequence composed of *Flowergarden* and *Football*, the error bits are reduced by 60% for I-frame, 40% for P-frame, and 30% for B-frame.

5.2. Adaptive versus Table-Lookup Models

Although the previous simulation results demonstrate the advantages of the adaptive bits estimation model over the constant coefficient model, we like to make one more comparison against a complicated yet potentially better approach. This bits estimation scheme is developed based

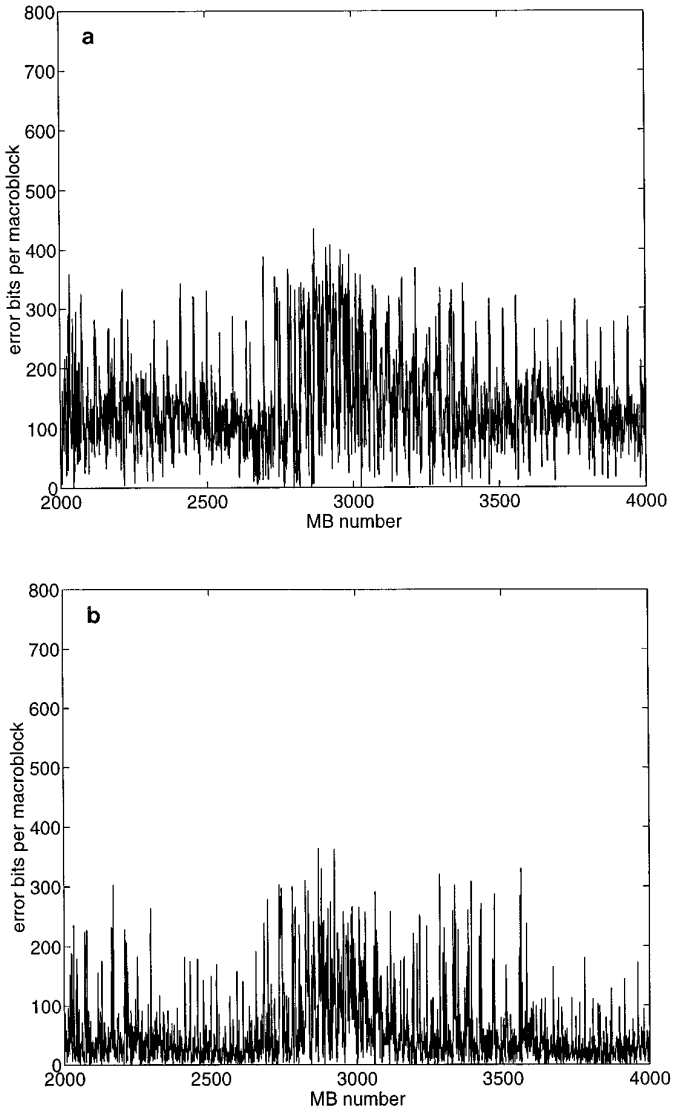

FIG. 9. The prediction error bits (absolute values) per macroblock of I-frame (a) using a constant-coefficient linear model and (b) using an adaptive piecewise linear model.

on clustering analysis. First, we collect the data pairs in the form of $(BIT, ACT/Q)$ and generate a table that contains the *representatives* of these data pairs using the K-means cluster algorithm [15, 16]. In other words, we chose a cluster number, say K; then, the iterative K-means procedure partitions the data samples into K cells, each centers around a *representative*. We continue modifying the cell representatives until the total square distance between the cell members and their representatives reaches a (local) minimum point. In our case, the representatives are denoted by $(Tbl1_{BIT}, Tbl1_{ACT/Q})$. They are entries in TABLE1. Then, we expand the dimension of each entry in the TABLE1 into the form of $(Tbl2_{BIT}, Tbl2_{ACT}, Tbl2_{Q})$. The coding data triplets, (BIT, ACT, Q), belonging to the same TABLE1 entry are further divided into M subcells based on their Q values. Thus we obtain a new table with entry $(Tbl2_{BIT}, Tbl2_{ACT}, Tbl2_{Q})$, where $Tbl2_{BIT}$ and $Tbl2_{ACT}$ are the average bits and activity values of all the coding data triplets classified to the same TABLE1 entry $(Tbl1_{BIT}, Tbl1_{ACT/Q})$ but with different quantization stepsize Q. This new table is called TABLE2.

Using clustering analysis, we divide the coding data space into many small regions (cells) in hoping that with appropriate choice of features (ACT and Q) the bits number in each small region (cell) is close to each other. Consequently, we could estimate the bit number based on the feature space partition specified by TABLE2. In other words, given an input MB, we first compute its activity

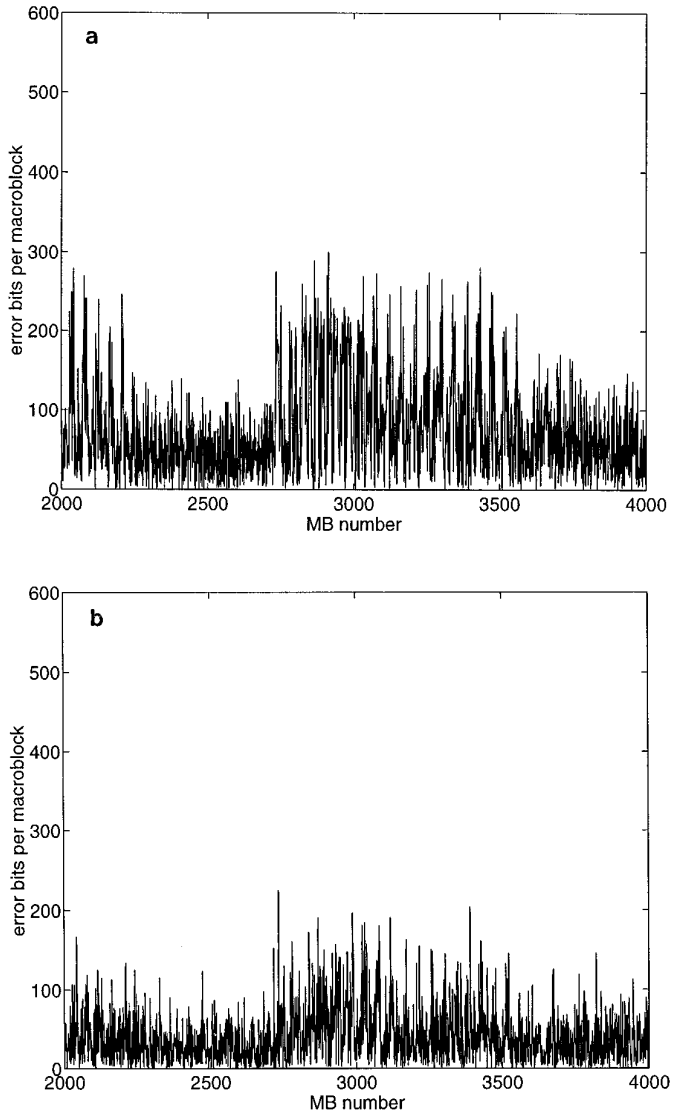

FIG. 10. The prediction error bits (absolute values) per macroblock of P-frame (a) using a constant-coefficient linear model, and (b) using an adaptive piecewise linear model.

value, *act*, and then for each possible quantization stepsize, q, we search this table to find the entry ($Tbl2_{BIT}$, $Tbl2_{ACT}$, $Tbl2_{Q}$) in which the $Tbl2_{Q}$ value matches q exactly and the $Tbl2_{ACT}$ value is closest to *act*; then, the $Tbl2_{BIT}$ value in that entry represents the estimated bits. Other than the fact that BIT, ACT, and Q are strongly related, this bits model does not assume a particular underlying relationship among BIT, ACT, and Q. In encoding an image sequence, (ACT, Q) acts as an index in searching the bits table. These features make this table lookup approach rather different from the previous adaptive bits estimation model. In the proposed adaptive piecewise linear bits model, we assume that the relationship between BIT and ACT/Q is nearly linear, and in the process of searching for the target linear

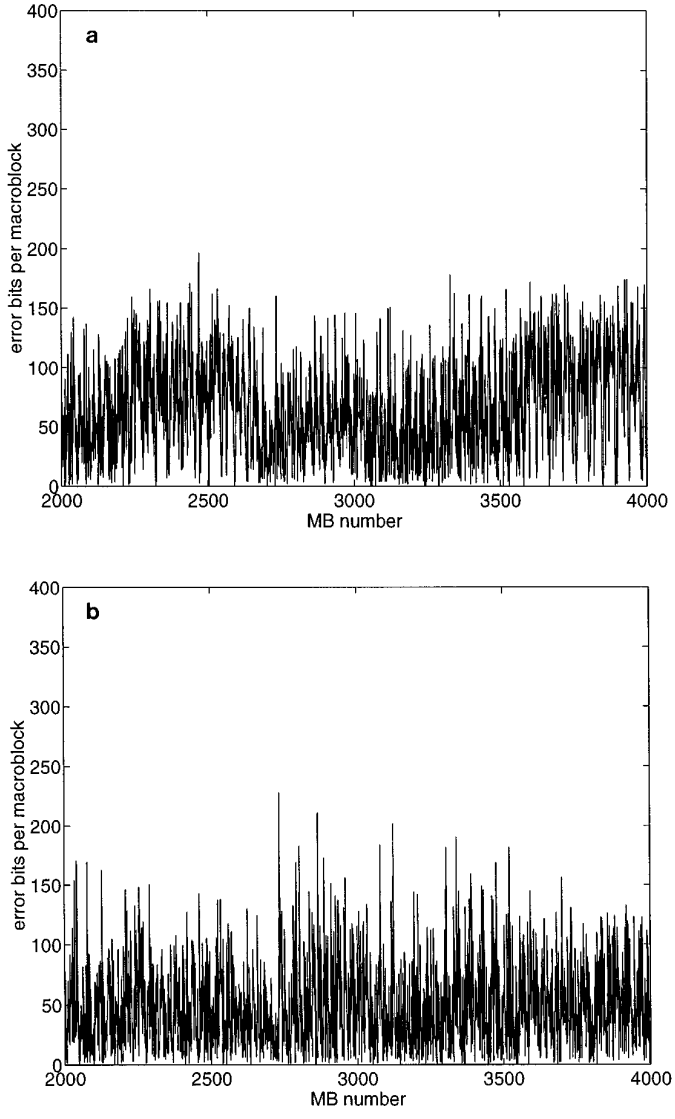

FIG. 11. The prediction error bits (absolute values) per macroblock of B-frame (a) using a constant-coefficient linear model and (b) using an adaptive piecewise linear model.

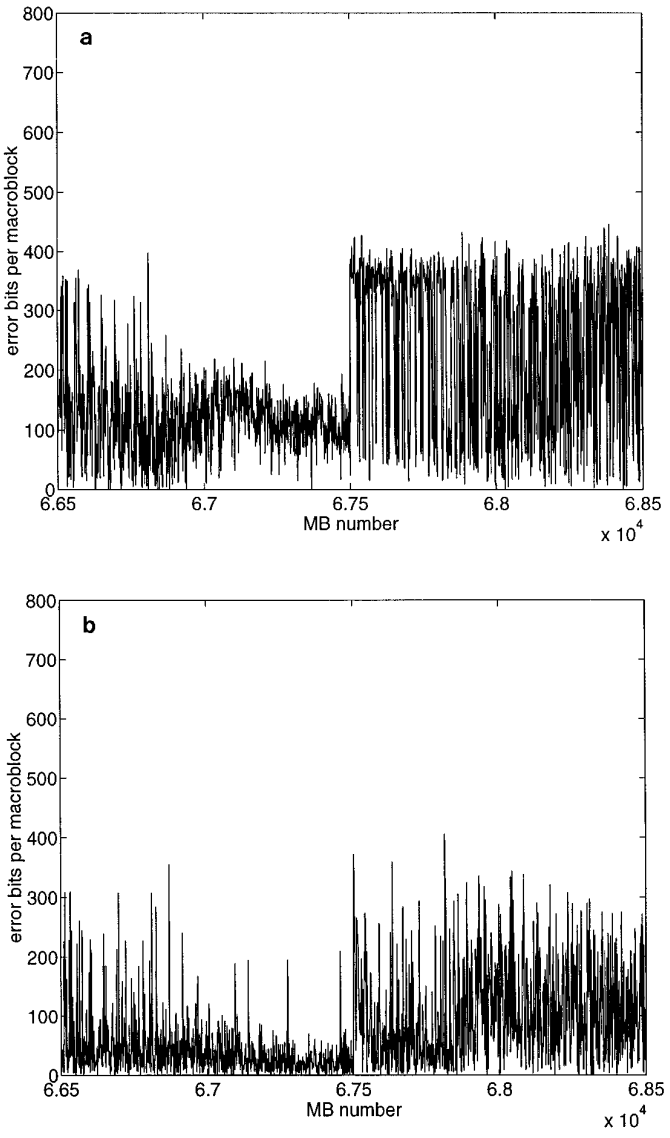

FIG. 12. The prediction error bits (absolute values) per macroblock at scene change (a) using a constant-coefficient linear model and (b) using an adaptive piecewise linear model.

bits estimation model we use ACT/Q not (ACT, Q). Other arrangements in table construction have been tested, but they produce less favorable results, such as constructing TABLE2 without constructing TABLE1 first. This may be due to the uneven distribution of data samples and, thus, performing the clustering operation directly on (BIT, ACT, Q) space leads to biased partitions.

Table 2 shows the average absolute error bits produced by the table lookup approach with different table sizes. The first two columns are the numbers of entries of TABLE1 and TABLE2. Because in our coding system the reasonable choice of Q could be any integer value from 1 to 30, the number of entries of TABLE2 is 30 times that

TABLE 1
The Average Absolute Error Bits per Macroblock of *Flowergarden*, *Football*, and *Flowergarden + Football* with/without Adaptation Using (piecewise) Linear Model

	I-frame		P-frame		B-frame	
Image sequence	Without	With	Without	With	Without	With
Flowergarden	149	60	83	39	54	43
Football	80	53	123	78	109	63
Flower + Football	146	57	107	58	74	52

of TABLE1. In the cases of which TABLE2 sizes are greater than 480, their average absolute error bits are comparable with those of the adaptive bits estimation model in Table 1. However, there are no benefits to increase the lookup table size beyond 960. The limitation comes from the uncertainty between the activity measure and the coding bits for a macroblock. That is, two macroblocks of the same activity value may produce different bit numbers. Unless we use a *better* activity measure (with less uncertainty), we could not further reduce the bit errors.

When Table 1 is compared to Table 2, the adaptive bits estimation model not only approaches the same performance limit, but it also uses a much smaller size memory than the table-lookup method. For example, the number of nodes in adaptive modeling is 15 for a tree of depth three, whereas the table size is 960 in the lookup-table approach. Also, the calculation required to produce a bit estimate is less for the adaptive bits estimation scheme. It takes three comparisons and three first-order linear equations in the adaptive bits model, but it takes 32 comparisons in the 960-entry lookup-table scheme. Most importantly, it is not easy to make the lookup table approach adaptive.

5.3. *Complete Coding Systems*

Finally we show the results of our complete MPEG coding system using the proposed bits estimation model for rate/buffer control. Figures 13 and 14 show the bits distributions per GOP, with and without adaptation for *Flowergarden* and *Football*, respectively. Also shown in Figs. 13 and 14 are the TM5 simulation results which will be explained in the next subsection. The fixed model designed for Q between 2 and 10 predicts fewer bits than the coder actually produces in the range of stepsizes used in simulation. Hence, the coded bits are constantly higher than the budget.

The motion compensation technique is quite effective in compressing the *Flowergarden* sequence because it is a panning image sequence. Since the image content is not changing very much between nearby frames, the fixed bits model is off the target, but it is still quite stable. In contrast, the *Football* sequence has fast, multiple object movement. Because its content changes rapidly, the motion estimation is not as effective. The fixed model fails in catching up with the changes. On the other hand, the adaptive bit estimation scheme controls the output bits rather precisely to match the design target. For reference, we also include the coded bits distribution of each frame of our proposed scheme (Figs. 15 and 16) and the average PSNR (Table 3). The effectiveness of motion compensation can also be observed from Figs. 15 and 16. To keep an equal PSNR for I-, P-, and B-frames, the bit rates of all these three types of frames in the *Football* sequence are almost equal, while the bit rates of the P- and B-frames are much smaller than that of the I-frame in the *Flowergarden* sequence. Because the coded pictures have rather high PSNR (35 to 40 dB), as one may expect, their subjective quality is very

TABLE 2
The Average Absolute Error Bits per Macroblock of *Flowergarden*, and *Football* Using Clustering/Table Method

TABLE1 size	TABLE2 size	Flowergarden			Football		
		I-frame	P-frame	B-frame	I-frame	P-frame	B-frame
16	480	83	65	50	72	78	73
32	960	68	43	35	57	63	58
64	1920	63	41	40	61	70	66
128	3840	67	40	43	67	70	69

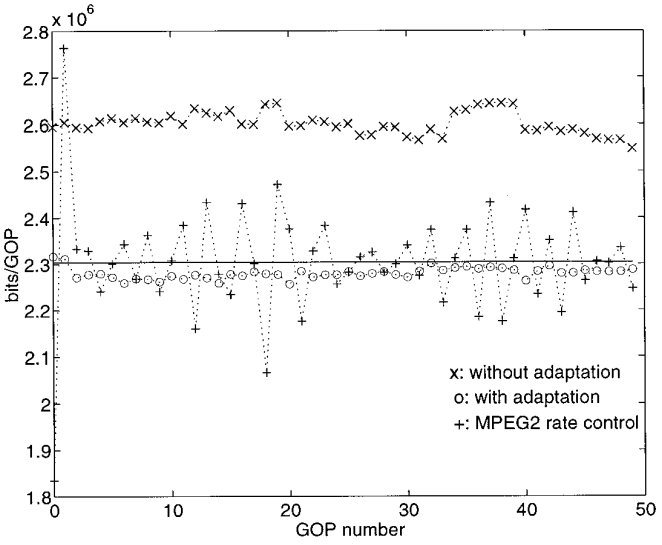

FIG. 13. Bits distribution of GOP for *Flowergarden*.

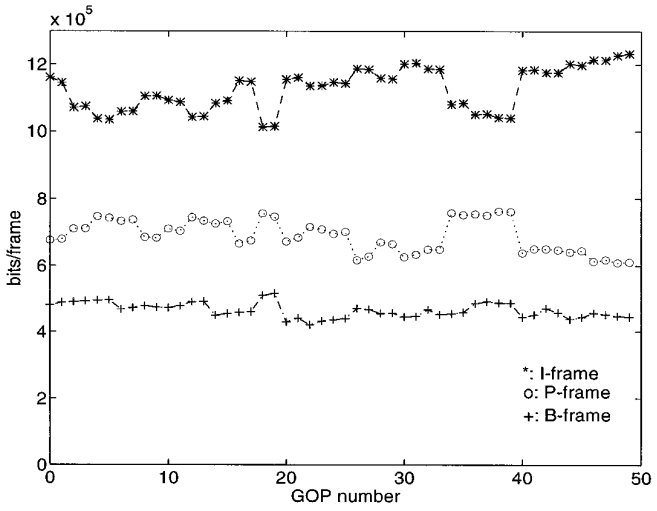

FIG. 15. Bits distribution of I-frames, P-frames, and B-frames for *Flowergarden* using an adaptive bit model.

good. Therefore, the reconstructed pictures are not included in the paper.

5.4. *MPEG2 Test Model 5*

At the end, we simulate the MPEG test model 5 (TM5, Rev. 2 in [3]) for comparison purposes. A typical rate/buffer control procedure can be split into two stages: bit allocation and quantization stepsize selection. Although the activity (image complexity) measures are different in TM5 and our scheme, the basic concepts behind their bit allocation algorithms are similar. The major difference between the TM5 and our rate control schemes is the selection of the quantization stepsize. TM5 decides the stepsize based on the buffer feedback information and the block activity measurement. It does not employ a bits model, whereas our scheme decides the stepsize based on the bits model (and the block activity measurement that determines the allocated bits of a block). There is no explicit use of buffer feedback information in our scheme, although it may also be included to further increase the preciseness of rate control and to prevent the buffer from overflow and underflow in the case of model breakdown in practical systems.

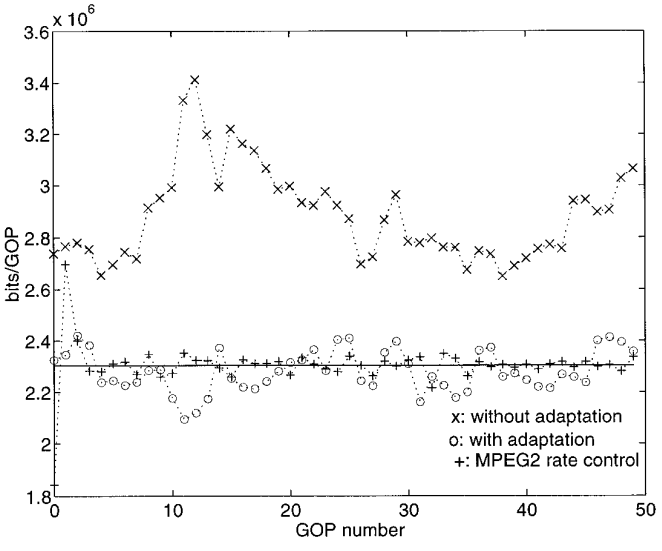

FIG. 14. Bits distribution of GOP for *Football*.

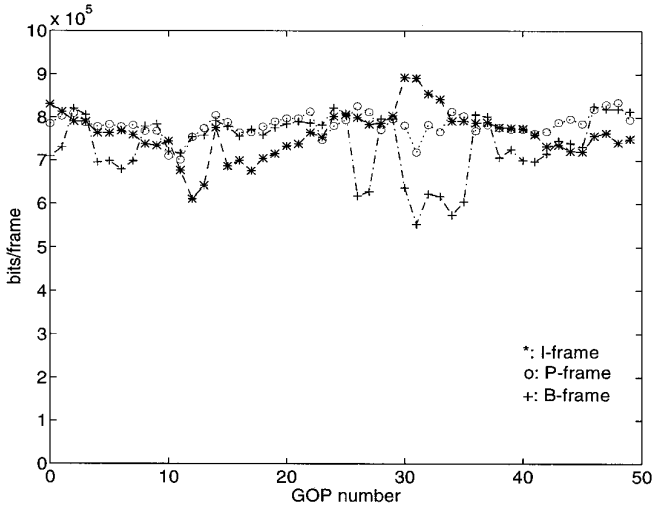

FIG. 16. Bits distributions of I-frames, P-frames, and B-frames for *Football* using an adaptive bit model.

TABLE 3
The Average PSNR of *Flowergarden* and *Football* (150 Frames/Sequence)

Image sequence	I-frame			P-frame			B-frame		
	Y	C_r	C_b	Y	C_r	C_b	Y	C_r	C_b
Flowergarden	34.8	32.2	29.6	34.7	31.9	29.5	35.6	32.2	29.6
Football	40.4	37.5	32.7	40.5	37.4	32.7	40.7	37.4	32.8

We review very briefly here the basic elements in the TM5 rate control scheme. Its complete description is tedious and is referred to [3]. The TM5 encoder also adheres to the single-pass coding philosophy. In other words, the current frame and GOP bit allocation is predicted by using the previously coded frame(s) and GOP information. The image complexity (or activity) measure (X) at the frame level is estimated based on the product of the average macroblock quantization stepsize (Q) and the number of coded bits (S) of the previous picture (of the same picture type); that is, $X = Q \times S$. Three image complexity measures are computed for the I-, P-, and B-frames separately. Although the exact formula in TM5 is more complicated in allocating the bits for each picture frame, its basic principle is similar to that of our frame bit allocation scheme (Eq. (6)).

The MB quantization stepsize in TM5 are decided by two factors: (1) the deviation from the planned buffer fullness, d; (2) the normalized MB activity, N_{act}. After the total bits of a frame are decided, TM5 further assumes the bits are evenly distributed for the entire picture. The virtual buffer fullness predicted by the preceding assumption is the *planned buffer fullness*. Again, three virtual buffers, one for each picture type, are used. The second factor, MB activity, is calculated based on the minimum variance of the four luminance 8×8 frame blocks and the corresponding four luminance 8×8 field blocks of the original picture. The variance measure is then normalized against the average variance of the most recently coded picture to produce N_{act}. Several constants are involved in the above calculations. Eventually, the MB quantization stepsize is a product of a scaled buffer fullness deviation d and the normalized MB activity N_{act}.

Comparing our quantization scheme with the one in TM5, we observe two major differences. The first one is the MB bits are assumed to be evenly distributed in TM5, but the MB bits in our scheme are decided by the MB activity (Eq. (8)). However, the MB activity still enters into the TM5 stepsize selection through a multiplicative factor (N_{act}). The second and more significant difference is that the TM5 stepsize is mainly decided by the buffer fullness deviation (the normalized MB activity does not change very much for typical blocks) to meet the planned bit budget. In our scheme, the stepsize is chosen solely based on our bits model to meet the bit budget. Therefore, it becomes very critical in our scheme to have an accurate bits model.

Another interesting point is that the complication in the TM5 rate control algorithm is due to the sophisticated formulas in calculating various parameters such as d and N_{act}. The complication in our rate control is due to bits model updating. Because N_{act} calculation in TM5 requires computing eight block variances, the TM5 rate control seems to be more demanding in computing power.

Because the purpose of this simulation is to compare the rate control algorithms, we did not activate the more advanced features in MPEG2 coding such as field estimation and field DCT. The TM5 simulation results using the same coding parameters as before (bit rate, IPB structure, etc.) are shown in Figs. 13 and 14. Figures 13 and 14 indicate that when the picture content is rather nonuniform such as in the case of *Flowergarden* sequence, the bit rate of individual GOP varies quite significantly. This is due to the fact that the even distribution of MB bits over the entire image is not valid in this case. The upper half of *Flowergarden* is the smooth sky and the lower half is the deep texture flower bed. The control parameter adjustment based on buffer fullness does not respond quick enough to match the image content changes. This phenomenon can be clearly observed from Fig. 17, which shows the (virtual I-frame) buffer fullness after every MB is coded. The goal of TM5 control is try to keep the buffer fullness curve nearly constant. In the case of *Flowergarden*, it generates too few bits in the upper half of the picture and it generates too many bits in the lower half. It turns out, in this particular frame, that the total bits is higher than expected at the end of the picture. In contrast, the moving objects and the background in the *Football* image are pretty much evenly distributed. Thus the buffer-fullness based TM5 rate control works quite well. Its buffer status is nearly flat in Fig. 17. The weakness of a buffer-feedback rate control scheme is that, although it may keep a long-term average rate close to the desired (one GOP, say), it usually cannot control bits precisely for every MB to meet the budget plan.

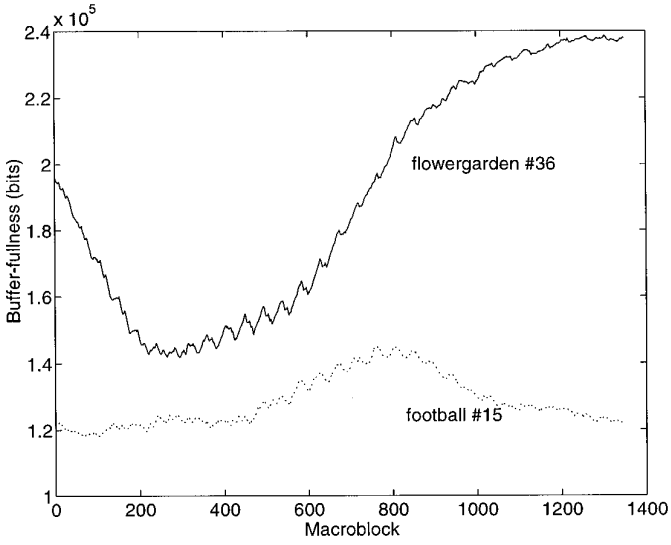

FIG. 17. Buffer status under TM5 rate control: 36th picture (I-frame) of *Flowergarden* and 12th picture (I-frame) of *Football*.

6. CONCLUSIONS

In this paper, we propose a bits estimation model with self-adaptation ability for real-time video coding. The tree structure of the adaptive bits estimation model provides a fast search for the best quantization stepsize and easy updating of the bits model. Experiments show that the number of error bits of the adaptive bits model decreases significantly as compared to the simple nonadaptive linear bits model. It is observed that this adaptation model is able to adjust itself quickly even at scene changes.

We also compare the adaptive bits estimation scheme with the bits model obtained using a clustering/table method. The latter bits model is a lookup table with entries storing the precomputed bits, activity, and quantization stepsize values. Simulation results indicate that the error bits of the adaptive bits model is comparable with that of the clustering/table model designed for a selected sequence. However, the adaptive bits model has a much lower complexity and is able to update its parameters to match the time-varying data.

At the end of the Simulation section, the experimental results using TM5 (of MPEG2) are included as a reference. The rate control scheme in TM5 is designed based on the buffer feedback concept. This type of schemes may work reasonably well, when properly designed, on maintaining long-term average bit rate and preventing the buffer from underflow or overflow. However, if we wish to have a precise control on the bits for every microblock, either a trial-and-error approach or a precise bits model has to be employed. Because the focus of our paper is on the bits model, the simple rate control scheme we propose does not include a buffer feedback mechanism. Its success relies solely on the accuracy of the bits model. We could further increase the stability and preciseness of our rate control scheme with a simple feedback mechanism.

REFERENCES

1. CCITT, Working Party XV/4, *Description of Ref. Model 8* (*RM8*), June 1989.
2. ISO/IEC JTCI/SC2/WG11, Doc. MPEG 90/41, *MPEG video simulation model three* (*SM3*), April 1991.
3. ISO/IEC JTCI/SC29/WG11, Doc. NO400 *Test Model 5*, April 1993.
4. L. Wang, Bit rate control for hybrid DPCM/DCT video codec, *IEEE Trans. Circuits Systems Video Technol.* **4,** No. 5, 1994, 509–517.
5. J.-J. Chen and H.-M. Hang, A transform video coder source model and its applications, *IEEE Int. Conf. Image Processing '94, Austin, Texas, Nov. 1994,* pp. 967–971.
6. A. Puri and R. Aravind, Motion-compensated video coding with adaptive perceptual quantization, *IEEE Trans. Circuits System Video Technol.,* **1,** No. 4, 1991, 351–361.
7. W.-Y. Sun, H.-M. Hang, and C.-B. Fong, Scene adaptive parameters selection for MPEG syntax based HDTV coding, in *Int'l Workshop on HDTV '93, Ottawa, Canada, Oct. 1993.*
8. S. B. Gelfand, C. S. Ravishankar, and E. J. Delp, Tree-structured piecewise linear adaptive equalization, *IEEE Trans. Commun.* **41,** 1993, 70–82.
9. S. B. Gelfand and C. S. Ravishankar, Tree-structured piecewise linear adaptive filter, *IEEE Trans. Inform. Theory* **39,** 1993, 1907–1922.
10. T. Berger, *Rate Distortion Theory, Prentice Hall,* Englewood Cliffs, NJ, 1971.
11. H.-Y. Gong and H.-M. Hang, Scene analysis for DCT image coding, in *Int'l Workshop on HDTV '93, Ottawa, Canada, Oct. 1993.*
12. S.-W. Wu and A. Gersho, Rate-constrained optimal block-adaptive coding for digital tape recording of HDTV, *IEEE Trans. Circuits Systems Video Technol.* **1,** No. 1, 1991, 100–112.
13. J.-B. Cheng, H.-M. Hang, and D. W. Lin, An image compression scheme for digital video cassette recording, in *IEEE International Conference on Consumer Electronics, Chicago, June 1994,* 26–27.
14. F. Azadegan *et al.,* "Data-placement procedure for multi-speed digital VCR, *IEEE Trans. Consumer Electron.* **40,** No. 3, 1994, 250-256.
15. J. A. Hartigan, *Clustering Algorithms*, Wiley, New York, 1975.
16. W. R. Dillon and M. Goldstein, *Multivariate Analysis: Methods and Applications,* Wiley, New York, 1984.

JIA-BAO CHENG received her B.S. and M.S. degrees in Electrical Engineering from National Chiao Tung University, Hsinchu, Taiwan in 1992 and 1994, respectively. From 1994 to 1995, she was with CCL/ITRI, Hsinchu, Taiwan, where she was a multimedium system engineer. Currently, she is an IC design engineer in SD Micro Co., Hsinchu, Taiwan. Her research interests include image/signal processing algorithms and architecture.

HSUEH-MING HANG received his B.S. degree and M.S. degree from National Chiao Tung University, Hsinchu, Taiwan in 1978 and 1980, respectively, and his Ph.D. in Electrical Engineering from Rensselaer Polytechnic Institute, Troy, NY in 1984. From 1984 to 1991, he was with AT&T Bell Laboratories, Holmdel, NJ, where he was engaged in digital image compression algorithm and architecture research. He joined the Electronics Engineering Department of National Chiao Tung University, Hsinchu, Taiwan in December 1991. His current research interests include digital video compression, image/signal processing algorithms and architecture, and digital communication theory. Dr. Hang was a conference co-chair of the *Symposium on Visual Communications and Image Processing* (*VCIP*)*, 1993,* and the Program Chair of the same conference in 1995. He guest coedited two *Optical Engineering* special issues on *Visual Communications and Image Processing* in July 1991 and July 1993. He was an associate editor of *IEEE Transactions on Image Processing* from 1992 to 1994 and a coeditor of the *Handbook of Visual Communications* (Academic Press, 1995). He is currently an editor of *Journal of Visual Communication and Image Representation,* Academic Press. He is a senior member of IEEE and a member of Sigma Xi.