Encoding Stored Video for Streaming Applications

I.-Ming Pao, Member, IEEE, and Ming-Ting Sun, Fellow, IEEE

Abstract—In streaming video applications, video sequences are encoded off-line and stored in a server. Users may access the server over a constant bit-rate channel. Examples of the streaming video applications are video-on-demand, archived video news, and noninteractive distance learning. Before the playback, part of the video bitstream is pre-loaded in the decoder buffer to ensure that every frame can be decoded at the scheduled time. For these streaming video applications, since the video is encoded off-line and the future video frames are available to the encoder, a more sophisticated bit-allocation scheme can be used to achieve better video quality. During the encoding process for streaming video, two requirements need to be considered: the pre-loading time that the video viewers have to wait and the physical buffer-size at the receiver (decoder) side. In this paper, we propose a sliding-window rate-control scheme that uses statistical information of the future video frames as a guidance to generate better video quality for video streaming involving constant bit-rate channels. A quantized discrete cosine transform coefficient selection scheme based on the rate-distortion measurement is also used to improve the video quality. Simulation results show video quality improvements over the regular H.263 TMN8 encoder.

Index Terms—Bit-allocation, buffer, rate control, streaming video.

I. INTRODUCTION

DIGITAL video applications have become increasingly popular in our everyday life. Currently, there are several video standards established for different purposes, for example, MPEG-1 [1] and MPEG-2 [2] for multimedia applications, and H.263 [3], [4] for video-conferencing applications. All these standards use discrete cosine transform (DCT), motion compensation (MC) (which involves motion estimation and motion-compensated prediction), quantization, and variable length coding (VLC) as the building blocks. A rate-control scheme, which decides the quantization step-size and monitors the buffer fullness, is another important part of the video encoder and can greatly affect the video quality; it is not specified in the standards and is left open for application-dependent implementation.

With the knowledge of the channel model, the rate-control scheme in the video encoder produces video bitstreams which can be supported by the channels. Rate-control schemes in current standard test models (e.g., MPEG2 TM5 [5] and H.263 TMN8 [6]) are usually used for real-time visual communications. Video encoders receive the video (image frames) from the video capture device and generate the compressed bitstream.

The bitstream is then send to the decoder over the channel. Delay is an important issue in real-time communication. For example, a delay of a few seconds is not acceptable for video conferencing applications. The whole process of capturing video, encoding, transmission, and decoding needs to be done within the delay constraint in real-time communication applications.

In this paper, our focus is on the non-real-time visual communications, such as video-on-demand, digital library, and non-interactive distance learning. For these applications, video sequences are encoded in advance and stored in the server. Users may access the server over a constant bit-rate channel, such as the Public Switched Telephone Network (PSTN) or Integrated Services Digital Network (ISDN) (Fig. 1). Before the playback, part of the video bitstream is pre-loaded in the decoder buffer to ensure that every frame can be decoded at the scheduled time. The pre-loading time (or how many bits need to be pre-loaded) depends on several factors, such as how much network delay jitter needs to be smoothed out, the physical buffer-size at the decoder, and the waiting time a viewer is willing to accept. The distribution of the bit counts for each frame may also affect the pre-loading time. For example, if the encoder uses the same number of bits (channel-rate/frame-rate) to encode every frame, the pre-loading of only the first frame is needed. However, due to the different complexity of video frames, every frame usually needs different number of bits for encoding, in order to produce good quality video. Pre-loading is necessary for the video sequence with active frames using a lot of bits which exceeds the channel bandwidth. Without proper pre-loading, these active frames will not be decoded at the right time because their bits may not have arrived at the decoder buffer due to the channel bandwidth limitation (in this paper we assume that there is no feedback from the client to the server).
Current rate-control schemes in the standard test models are focused on low-delay real-time visual communications. They use the information, such as the scene complexity and video encoder buffer fullness, from previous and current frames to make the bit-allocation decision (the quantization step-size decision). These rate-control schemes may not give good video quality during the scene-change period because they do not have the information of the future frames. In nonreal-time visual communication, such as streaming video applications, the future video frames are available to the encoder. We propose a rate-control scheme which uses future frame statistics to allocate the bits for video frames in nonreal-time video applications over a constant bitrate channel. Scene changes and frames with large motion can be properly taken care of. Bits can be saved from the frames that are easy to encode and hence can be used on the frames that require more bits to encode, to generate higher video quality.

The rest of the paper is organized as follows. In Section II, we formulate the problem and review the related literature. In Section III, we propose a sliding-window encoding scheme. In Section IV, we discuss the physical decoder buffer-size and maximum pre-loading time requirements. In Section V, we present how to perform the sliding-window encoding scheme under the decoder buffer size and pre-loading time constraints. In Section VI, we describe how to quantize the DCT coefficients to achieve better quality in a rate-distortion sense. In Section VII, we show the simulation results. The last section is the conclusion.

II. Problem Definition

The bit allocation problem in video encoding can be stated as follows. Given a quota of bits, how do we encode the video sequence such that the encoded video can achieve the highest quality? Other constraints, such as channel condition and delay limitation, should also be included in the consideration [7]. The decision-making process of the video encoding includes the selection of picture-types (I-, P-, and B-pictures), encoding modes (intra and inter mode), motion vectors, and quantization step-sizes. To achieve the best video quality, rate-distortion optimization techniques were used to find the optimal coefficients and quantizers [8], [9], motion vector [10], [11], and mode selection [12], in various applications (e.g., video over wireless channels [13]). Lagrange multiplier and dynamic programming are two popular techniques to find the optimal bit-allocation that results in minimum distortion for the compressed video under the bit-budget constraint [14]. One difficulty is that they require a huge amount of computations, especially for video encoding which involves choices of selecting macroblock modes, quantizers, and motion vectors. Also, the quantizer and motion vector are differentially coded from the data of the previous macroblocks; this dependency makes the complexity grow exponentially with the number of macroblocks considered.

The other type of approach for rate control is through mathematical modeling [15]–[19]. Formulas are derived from optimization based on the rate and distortion models. With information such as the current buffer fullness and the variance of the residual prediction-errors as the inputs, mathematical formulae are then used to decide the quantization step-sizes for rate-control. Current test models of video standards mainly use this type of rate-control strategy due to its simplicity. The disadvantage is that if the model does not fit the input video well, the video quality will not be as good.

The rate-control schemes mentioned above are mainly used in real-time communication situations. For nonreal-time streaming video applications over a constant bitrate channel, the constraints are different. Since the video sequence is available and encoded off-line, the encoding delay is not a major concern. The pre-loading time that the video is pre-loaded in the decoder buffer before playback, needs to be considered. Although the encoder complexity may not be a major concern, it needs to be still reasonable for a long video sequence. To reduce the complexity, our proposed encoding scheme considers the statistics from the future frames within a sliding-window [20]. Our scheme can achieve better video quality because the encoder utilizes the future frame statistics. Within the sliding windows, we assign more bits to the complex frames and fewer bits to the less complex frames to achieve an overall better video quality.

After the video bitstream is generated from the sliding-window encoding scheme, we discuss how to calculate the required decoder buffer-size and the pre-loading time, based on the number of bits of each frame. On the other hand, if the maximum pre-loading time and the physical decoder buffer size is known before encoding, a rate-control algorithm considering the constraints is proposed [21]. The encoder allocates bits to video frames while satisfying the decoder buffer-size and the pre-loading time constraints.

III. A Sliding-Window Encoding Scheme

To encode video sequences without the real-time constraint, we propose a sliding-window encoding scheme which uses future frame statistics to achieve significant video quality improvement. Let \( W \) be a sliding window size. For encoding the current frame (frame \( i \)), the statistics of \( W \) frames, which include frame \( i, i + 1, \ldots, i + W - 1 \), are collected. The information is then used to generate a better quality video bitstream. Compared with the real-time encoder, the proposed encoder can achieve a better video quality for the same bitrate, because the information of the future frames is also considered for properly allocating the bits for the current frame.

A. Information Collection

First, video sequences are encoded using a fixed quantization step-size. If the encoder is encoding frame \( i \), information, such as variances of macroblocks (motion compensated difference blocks for inter mode and pixel blocks for intra mode), and the bit-counts for header and motion vector of frame \( i, i + 1, \ldots, i + W - 1 \) are collected. These information will be used as the guidance to re-allocate bits and to achieve a better video quality.

B. Bit Re-Allocation

Mathematical modeling is a common approach used in the rate control. The rate and distortion models we used in this paper...
are based on [18]. The number of bits $B_i$ produced by frame $i$ can be approximated by

$$B_i(\sigma_i, Q_i) = A \left( K \frac{\sigma_i^2}{Q_i} + C \right) \quad (1)$$

where
- $A$: number of pixels in a frame;
- $K$: if the DCT coefficients can be modeled as Laplacian distribution;
- $\sigma_i^2$: variance of frame $i$;
- $Q_i$: quantization step-size used in coding the frame $i$;
- $C$: models the average rate (in bits per pixel) to encode the motion vectors and the header information for the frame.

The distortion in the macroblock is introduced by uniformly quantizing its DCT coefficients with a quantizer of step-size $Q$. We consider the following typical approximation to the mean squared error (MSE) between the original and encoded pixels $Distortion_i = Q_i^2/12$.

In our rate-control algorithm, we would like to derive formulas for the target bit numbers for each frame that minimize distortion. Let the bit budget to encode these $W$ frames be $B$. The optimal quantizer found by using Lagrange multiplier (refer [18] for more details) for frame $i$ will be

$$Q_i^* = \sqrt{ \frac{AK}{(B - AWC)} \sigma_i \sum_{k=i}^{i+W-1} \sigma_k } \quad (2)$$

We can derive the ideal number of bits $B_i^*$ required for encoding the quantized coefficients in frame $i$ using the ideal quantizer $Q_i^*$ from the model as $B_i^* = A(K \times \sigma_i^2/Q_i^2 + C)$. With $Q_i^*$ from (2), we have

$$B_i^* = (B - AW) \sigma_i \sum_{k=i}^{i+W-1} \sigma_k + AC \quad (3)$$

$B_i^*$ will be used as the target number of bits for encoding frame $i$. The encoder will re-allocate the number of bits for each frame proportional to the weight of the standard deviation $\sigma_i$ of frame $i$ in the $W$ frames. The following section describes the sliding-window encoding scheme.

**Step 0: Initialization:** Initialize the bit-count regulator $\delta_{i-1} = 0$, where $\delta_i$ is used to monitor the difference (excess or deficit) between the ideal target bit number and the actual number of bits generated from the encoder up to frame $i$.

**Step 1: Compute the Proposed Target Bits for Frame $i$:** The ideal target number of bits $\hat{B}_i$ for frame $i$ can be computed as follows ($i = 0, 1, 2, \ldots$):

$$\hat{B}_i = \left( \frac{R}{F} \times W - \sum_{k=i}^{i+W-1} H_k - \delta_{i-1} \right) \times \frac{\sigma_i}{\sum_{k=i}^{i+W-1} \sigma_k} + H_i \quad (4)$$

where $R$: channel rate; $F$: frame rate; $W$: window size; $H_i$: number of bits for the header and motion vector for frame $i$.

$\hat{B}_i$ is then used as the target number of bits for encoding frame $i$.

**Step 2: Macroblock-Layer Rate-Control:** TMN8 macroblock-layer rate-control is used to distribute $\hat{B}_i$ to the macroblocks in the $i$th frame. The quantization parameter $QP$ and quantized DCT coefficients of each macroblock are encoded according to the rate-distortion scheme described in Section VI.

The actual bit number $B_i$ used to encode frame $i$ will be used to update the bit-count regulator.

**Step 3: Update Bit-Count Regulator:** After encoding each frame, the regulator is updated by $\delta_i = \delta_{i-1} + B_i - R/F$. If there are more frames to be encoded, go to Step 1, or else stop.

### IV. Decoder Buffer-Size and Pre-Loading Time Requirement

In streaming-video applications, videos stored in the server are sent to the decoder through communication channels. A buffer at the decoder is needed to store the excess bits received from the channel, waiting to be decoded at the right time. Bits of the future frames can be sent to the decoder buffer in advance as long as the channel bandwidth is available and the decoder buffer does not overflow. Pre-loading is allowed as long as the users can tolerate the pre-loading delay before playback. In this section, we discuss the pre-loading time and the decoder buffer-size constraints on the rate-control scheme.

The number of bits in the decoder buffer is related to the pre-loading time and the bitstream generated by the encoder, and can be derived by the following formula. Assume that the frame-rate is $F$ [e.g., 10 frames per second (fps)], the number of bits generated from frame $i$ is $B_i$ (we define the first frame to be frame 0 with number of bits $B_0$), the channel rate is $R(t)$, and the pre-loading time is $L$ s. At the frame level, frame $i$ will be decoded at time instant $i/F$ and the decoder will remove $B_i$. 

Fig. 2. Bit counts in the decoder buffer.
number of bits from the decoder buffer. Assuming the initial buffer is empty before pre-loading \( p(-L) = 0 \) and \( p(t) \) is the number of bits in the decoder buffer at time \( t \), we have

\[
\begin{align*}
  p(t) &= \int_{-L}^{t} R(t) \, dt, \quad -L \leq t < 0 \\
  p(t) &= \int_{-L}^{t} R(t) \, dt - \sum_{k=0}^{s} B_k, \quad t \geq 0
\end{align*}
\]  

where \( s \) is the maximum integer \( \leq t \). In the following, we assume a constant transmission channel (i.e., \( R(t) = R \)). Fig. 2 shows an example of the decoder buffer variation. The \( x \)-axis in the figure is the time-axis represented by frame numbers with the first frame (frame 0) being decoded at \( t = 0 \) (e.g., for a 10 fps video, the frame 10 is decoded at time instant 1.0 s). The pre-loading period is from \( t = -L \) to \( t = 0 \), during which the video bitstream is received from the channel and accumulated in the buffer at a fixed rate (assuming a constant channel). The video playback starts at \( t = 0 \). At the time instant \( t = i/F \) that frame \( i \) is scheduled to be decoded, the decoder takes the bits for decoding frame \( B_i \) out of the buffer. Equation (5) can be expressed in a recursive form for the time instant at \( t = i/F \) (i.e., the time instant for frame \( i \) to be decoded) as

\[
\begin{align*}
  p_0 &= P - B_0 \\
  p_i &= p(i/F) = p_{i-1} + R/F - B_i, \quad \text{for } i = 1, 2, 3, \ldots
\end{align*}
\]  

where

- \( B_i \) number of bits for frame \( i \);
- \( p_i \) number of bits in the decoder buffer right after \( B_i \) is taken out of the buffer;
- \( P \) number of bits pre-loaded in the buffer before playback.

Right after decoding a frame, if \( p_i > 0 \), it means there are \( p_i \) bits of future frames that have already arrived and were stored in the buffer. In order to prevent decoder buffer overflow, the buffer size \( G \) should be larger than \( \max(p(t)) \).

If \( p_i < 0 \) occurs after decoding a frame, underflow occurs; this means the buffer is depleted and the decoder does not have the needed bits of the current frame to decode. In this case, the decoder cannot decode the frame at the scheduled time instant and needs to wait for the arrival of the belated bits. To prevent this situation from happening, we need to increase the pre-loading time \( L \) and store more bits in advance. The encoder should not generate bitstreams which will cause \( p(t) < 0 \).

### A. Finding Decoder Buffer Size and Pre-Loading Time Given a Video Bitstream

The required decoder buffer size and pre-loading time can be calculated for a given video bitstream, as shown in Fig. 3. Given a bitstream, the number of bits \( B_i \) for each frame \( i \) are known to the encoder. Assume that we have a virtual decoder buffer and the first frame (frame 0) is decoded at time \( t = 0 \). Fig. 3(a) shows an example of the decoder buffer variation. The \( x \)-axis in the figure is the time-axis represented by frame numbers with the first frame (frame 0) being decoded at \( t = 0 \) (e.g., for a 10 fps video, the frame 10 is decoded at time instant 1.0 s). The pre-loading period is from \( t = -L \) to \( t = 0 \), during which the video bitstream is received from the channel and accumulated in the buffer at a fixed rate (assuming a constant channel). The video playback starts at \( t = 0 \). At the time instant \( t = i/F \) that frame \( i \) is scheduled to be decoded, the decoder takes the bits for decoding frame \( B_i \), out of the buffer. Equation (5) can be expressed in a recursive form for the time instant at \( t = i/F \) (i.e., the time instant for frame \( i \) to be decoded) as

\[
\begin{align*}
  p_0 &= P - B_0 \\
  p_i &= p(i/F) = p_{i-1} + R/F - B_i, \quad \text{for } i = 1, 2, 3, \ldots
\end{align*}
\]  

where

- \( B_i \) number of bits for frame \( i \);
- \( p_i \) number of bits in the decoder buffer right after \( B_i \) is taken out of the buffer;
- \( P \) number of bits pre-loaded in the buffer before playback.

Right after decoding a frame, if \( p_i > 0 \), it means there are \( p_i \) bits of future frames that have already arrived and were stored in the buffer. In order to prevent decoder buffer overflow, the buffer size \( G \) should be larger than \( \max(p(t)) \).

If \( p_i < 0 \) occurs after decoding a frame, underflow occurs; this means the buffer is depleted and the decoder does not have the needed bits of the current frame to decode. In this case, the decoder cannot decode the frame at the scheduled time instant and needs to wait for the arrival of the belated bits. To prevent this situation from happening, we need to increase the pre-loading time \( L \) and store more bits in advance. The encoder should not generate bitstreams which will cause \( p(t) < 0 \).

### B. Generating a Video Bitstream Given Decoder Buffer Size and Pre-Loading Time

If a maximum pre-loading time \( L \) and the decoder buffer-size \( G \) are given, the encoder should produce a bitstream which makes the decoder buffer-fullness curve \( p(t) \) in the range of \([0, G]\). We now discuss what the encoder can do to obey these requirements and yet create a better video quality.

To prevent the decoder buffer-underflow (i.e., the situation where the decoder is scheduled to decode frame \( i \), but the bits associated with frame \( i \) is not in the buffer yet), we need to keep \( p(t) \geq 0 \). From (5), we have

\[
P + \frac{R}{F} \sum_{i=1}^{s} B_i \geq 0
\]  

where \( P \), the bits pre-loaded in the decoder buffer, is given by \( P = L \times R \).

To prevent the decoder buffer-overflow (i.e., the bits stored in the decoder buffer exceeds the decoder buffer-size), we need
to keep \( p(t) \leq G \), where \( G \) is the physical decoder buffer-size. From (5), we have

\[
P + \frac{R}{F} \cdot (t + 1) - \sum_{k=0}^{i} B_k \leq G \tag{8}
\]

Equations (7) and (8) are the requirements that the encoder needs to follow to avoid decoder buffer overflow and underflow. Let the bit-count regulator \( \delta_{i-1} = 0 \) and \( \delta_i = \delta_{i-1} + B_i - R/F \); then (7) and (8) become

\[
P \geq \delta_{i-1} + B_i \tag{9}
\]

\[
\delta_{i-1} + B_i - R/F \geq P - G. \tag{10}
\]

If the decoder has a buffer size of \( G \) bits and the pre-loading time is set to \( L \) seconds (i.e., \( P = L \times R \) bits are pre-loaded), and the encoder can generate video bitstream with number of bits \( B_i \) for each frame \( i \) (first frame is frame 0) satisfying (9) and (10), then the decoder buffer will not underflow nor overflow.

V. BIT REALLOCATION WITH DECODER BUFFER SIZE AND PRE-LOADING TIME CONSTRAINTS

If the decoder buffer size and the pre-loading time are limited, the following encoding scheme will generate bitstreams which meet the constraints.

Let us say that the encoder wants to generate a bitstream for a decoder with a decoder buffer-size of \( G \) bits and the user can tolerate \( L \) seconds of pre-loading delay. For a constant channel with a rate \( R \), the maximum number of bits that can be pre-loaded into the decoder buffer is \( P = R \times L \) bits. The following section describes the sliding-window encoding scheme considering the constraints.

**Step 0: Initialization:** Initialize the bit-count regulator \( \delta_{i-1} = 0 \), where \( \delta_i \) is used to monitor the difference (excess or deficit) number of bits between the ideal target bit number and the actual number of bits generated from the encoder up to frame \( i \).

**Step 1: Compute the Proposed Target Bits for Frame \( i \):** The ideal target number of bits \( \hat{B}_i \) for frame \( i \) can be computed as shown in (4). To avoid violating the underflow and overflow constraints, (9) and (10) need to be satisfied. Based on the bit-count regulator \( \delta_{i-1} \) and the ideal target number of bits \( \hat{B}_i \), we check if the decoder buffer is close to overflow or underflow ((11) and (12) are used to check whether the buffer is close to underflow or overflow, respectively), where \( \alpha_1 \) and \( \alpha_2 \) are threshold constants (<1). In the simulation, we set \( \alpha_1 = \alpha_2 = 0.9 \).

\[
\text{if } \alpha_1 \cdot P < \delta_{i-1} + \hat{B}_i
\]

\[
\hat{B}_i = \alpha_1 \cdot P - \delta_{i-1} \tag{11}
\]

\[
\text{else if } \delta_{i-1} + \hat{B}_i - R/F < \alpha_2 \cdot (P - G)
\]

\[
\hat{B}_i = \alpha_2 \cdot (P - G) + R/F - \delta_{i-1} \tag{12}
\]

\( \hat{B}_i \) is then used as the target number of bits for encoding frame \( i \).

Step 2: Macroblock-Layer Rate-Control: TMN8 macroblock-layer rate-control is used to distribute \( \hat{B}_i \) to the macroblocks in the \( i \)th frame. The quantization parameter \( QP \) and quantized DCT coefficients of each macroblock are encoded according to the rate-distortion scheme described in Section VI.

The actual bit number \( B_i \) used to encoding frame \( i \) will be used to update the bit-count regulator.

**Step 3: Update Bit-Count Regulator:** After encoding each frame, the regulator is updated by \( \delta_i = \delta_{i-1} + B_i - R/F \). If there are more frames to be encoded, go to Step 1, or else stop.

VI. DCT COEFFICIENT SELECTION

In this section, we propose a scheme to perform quantization on DCT coefficients in a rate-distortion sense at the macroblock level, which can help improve the video quality [1]. Quantizer step-sizes largely determine the rate-distortion tradeoff in the compressed video. However, the quantizer step-size choices in the video standards are limited. For example, in H.263, the quantization parameter (which is half of the quantizer step-size) ranges from 1 to 31 and is applied to all coefficients (except intra/dc). After quantization, codewords for nonzero coefficients are found by table lookup. The run-length coding tables for quantized coefficients are derived from statistics of typical training video sequences and are not customized for arbitrary inputs. The limited quantizer selections and the predetermined run-length codeword tables will not be optimal for all video sequences. Better performance can be achieved by adjusting the level of the quantized coefficients, which minimizes the distortion subject to the rate constraint. Instead of encoding the quantized coefficients faithfully (i.e., encoding every quantized coefficient with its original quantized value), the encoder can adjust the quantized coefficient’s level which may result in a marginal distortion increase but with a significant bit-rate reduction. We have used the Lagrange multiplier method for rate-distortion optimization in selecting the quantization parameter \( QP \) and adjusting the levels of the DCT coefficients. The best combination of QP and levels of DCT coefficients, which results in the lowest cost in the rate-distortion sense, is selected and encoded. In the process, the bits used for header information (which includes the macroblock mode and quantizer) are also considered as part of the bitrate for optimization, since in the low bitrate case, the header can contribute to a significant fraction of the overall bit rate.

In H.261 and MPEG, the nonzero quantized coefficients are run-length coded, based on a combination of (RUN, LEVEL), and the last nonzero coefficient is followed by an EOB symbol. In H.263, 3-D VLC is used to encode (LAST, RUN, LEVEL) where LAST is the last nonzero coefficient indication ("0": there are more nonzero coefficients in the current block; "1": this is the last nonzero coefficient in the current block), RUN is the number of successive zeros preceding the coded coefficient, and LEVEL shows the nonzero value of the coded coefficient. The less-probable combinations of (LAST, RUN, LEVEL) are coded with a 22-bit fixed-length codeword. Let \( C = \{ \hat{c}_1, \hat{c}_2, \ldots, \hat{c}_{22} \} \) be the \( 8 \times 8 \) DCT coefficients in the zigzag scan order and \( \hat{C} = \{ \hat{c}_1, \hat{c}_2, \ldots, \hat{c}_{32} \} \) be the quantized coefficients. We wish.
to find the optimal set of quantized DCT coefficients for every 8 × 8 block of a video frame such that the MSE distortion between the original block and the reconstructed block is minimized subject to a maximum target coding bit rate constraint.

The inverse quantization process is defined in the video standard documents and should be followed by every standard decoder. For example, in H.263+, the inverse quantization rules to find the quantized coefficient \( \hat{c}_k \) are as follows:

\[
\begin{align*}
|\hat{c}_k| &= QP \times (2 \times |LEVEL_k| + 1), & \text{if } QP \text{ is odd} \\
|\hat{c}_k| &= QP \times (2 \times |LEVEL_k| + 1) - 1, & \text{if } QP \text{ is even}
\end{align*}
\]

where \( QP \) is the quantizer parameter used in the macroblock.

Determining the LEVELs of DCT coefficients in the encoding process is not defined in the standards and is left open to the encoder’s choice. A straightforward implementation to find the LEVEL such as the one used in TMN8 (for intra mode, except dc) is:

\[
|LEVEL_k| = \text{Truncate} \left( \frac{|c_k|}{2 \times QP} \right)
\]

where \( \text{Truncate}(x) \) truncates \( x \) to the nearest integer toward zero.

Each DCT frequency component has a different distribution and the real input distribution varies with time. Using the same quantizer step-size for every frequency component in one macroblock and using (14) will not give optimal LEVEL in the \( R-D \) sense. Thus, our goal is to find the minimum distortion under the rate constraint by selecting the optimal QP for the macroblock and LEVEL for each coefficient

\[
\min D(\hat{C}), \quad \text{subject to } R(\hat{C}) < R_{\text{budget}}
\]

where \( R_{\text{budget}} \) is the rate constraint. The MSE is used as the distortion metric: \( D(\hat{C}) = \sum_{i=0}^{63} (c_i - \hat{c}_i)^2 \). \( R(\hat{C}) \) is the number of bits used for encoding \( \hat{C} \)

\[
R(\hat{C}) = \sum_{i=0}^{63} r(\hat{c}_i) + MB\text{header\_bits}
\]
where $r(\hat{c}_i)$ is the bits used for encoding $\hat{c}_i$ and MB-header bits is the number of bits used to encode the macroblock header information. Usually in H.263, an extra five bits are required for macroblocks using a QP different from that for the previous macroblock.

We should note that $r(\hat{c}_i) = 0$ if $\hat{c}_i = 0$ and $r(\hat{c}_i)$ is not only determined by the LEVEL of $\hat{c}_i$ but also by the number of successive zeros (RUN) preceding the coefficient. Adjusting a coefficient’s LEVEL to zero will affect the RUN of the following nonzero coefficient. $\hat{C}$ can be thought of as a function of QP and LEVEL, as shown in (13). The constrained problem of (15) can be solved by converting it to an unconstrained problem through the Lagrange multiplier $\lambda$. The problem becomes the determination of the LEVELs of the coefficients, which results in the minimum Lagrangian cost function $J$, defined as

$$
\min \left[ J(\lambda, \hat{C}) \right] = \min \left[ D(\hat{C}) + \lambda R(\hat{C}) \right]. \quad (17)
$$

Based on (17), the encoder searches for the optimal QP and coefficients’ LEVELs which minimize the distortion under the bit budget constraint and are compliant to the standard syntax. Based on the inverse quantization rules given in (13), we quantize the DCT coefficients $\hat{C}$ using the quantizer based on the nearest neighbor condition [22], i.e., for a given coefficient $c_i$, the quantized coefficient $\hat{c}_i$ gives the minimum distortion from all possible LEVEL selections. Let $C' = \{c'_1, c'_2, \ldots, c'_3\}$ be the quantized coefficients using QP with corresponding LEVELs $L' = \{L'_0, L'_1, \ldots, L'_{63}\}$. We then find the optimal set of quantized DCT coefficients $\hat{C}$ (or the corresponding optimal LEVELs $L = \{L_1, L_2, \ldots, L_{63}\}$) satisfying (17). For each coefficient, the encoder searches the possible candidates for $\hat{c}_i$ between 0 and $c'_i$ (corresponding to $L_i$ between 0 and $L'_i$) since, for coefficients with same RUN, bigger LEVEL will always have longer or equal codeword length in the H.263 VLC tables.
For example, let QP = 5 (quantizer step-size is $2 \times QP = 10$) and $(c_1, c_2, c_3, c_4, c_5) = (10, 0, 0, 4, 45, 10)$. After quantization, the LEVELs for coefficients $c_2$ to $c_5$ are $(l_2, l_3, l_4, l_5) = (0, 0, 4, 45)$. These four coefficients will be coded as (LAST, RUN, LEVEL) = (0, 3, 4) with a 22-bit codeword using the H.263 VLC table. If we artificially set the LEVEL of the fourth coefficient to be 1, i.e., $(l_2, l_3, l_4, l_5) = (0, 0, 1, 4)$, the coefficients $(l_2, l_3, l_4)$ will be coded as (LAST, RUN, LEVEL) = (0, 2, 1) with a 5-bit codeword and the coefficient $l_5$ will be coded as (LAST, RUN, LEVEL) = (0, 0, 4) with an 8-bit codeword. Extra distortion is introduced in the fourth coefficient, but the bit count is reduced from 22 to 13 bits. Another way to reduce the bit count is by setting $(l_2, l_3, l_4, l_5) = (0, 0, 0, 4)$ with an 8-bit codeword. To decide the best LEVEL combinations in the rate-distortion sense given the VLC table and the bit-budget constraint, (17) is applied to find the optimal solution.

### VII. Simulation Results

Simulations have been performed at different bitrates (32, 64, and 128 kbits/s). Different types of video sequences are tested. Foreman sequence is a video sequence of a foreman with large facial movements and camera panning at the end. Miss_am + Suzie sequence is a video sequence that concatenates two head-and-shoulder sequences (Miss_am and Suzie) to test the scene-change situation. Camera sequence is a typical sequence of video conferencing scene which is captured by a desktop digital video camera.
TABLE II
PSNR COMPARISON FOR H.263 TMN8 AND PROPOSED METHOD WITH WINDOW SIZE = 100

<table>
<thead>
<tr>
<th>Video sequence</th>
<th>Channel rate</th>
<th>Average PSNR (dB)</th>
<th>PSNR standard deviation</th>
<th>Average PSNR Improvement (dB)</th>
<th>Maximum PSNR improvement (dB)</th>
<th>Maximum PSNR degradation (dB)</th>
<th>Required buffer size (kbits)</th>
<th>Required pre-loading time (second)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foreman</td>
<td>32 kb/s</td>
<td>TMN8: 28.95</td>
<td>1.18</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>14.00</td>
<td>0.44</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Proposed: 29.33</td>
<td>0.97</td>
<td>0.38</td>
<td>1.87</td>
<td>0.53</td>
<td>26.10</td>
<td>0.75</td>
</tr>
<tr>
<td></td>
<td>64 kb/s</td>
<td>TMN8: 31.53</td>
<td>1.24</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>23.12</td>
<td>0.36</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Proposed: 31.98</td>
<td>0.90</td>
<td>0.45</td>
<td>3.35</td>
<td>0.56</td>
<td>53.75</td>
<td>0.46</td>
</tr>
<tr>
<td></td>
<td>128 kb/s</td>
<td>TMN8: 34.28</td>
<td>1.25</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>42.57</td>
<td>0.33</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Proposed: 34.91</td>
<td>1.04</td>
<td>0.63</td>
<td>4.51</td>
<td>0.79</td>
<td>135.04</td>
<td>0.44</td>
</tr>
<tr>
<td>Miss_am</td>
<td>32 kb/s</td>
<td>TMN8: 36.93</td>
<td>3.35</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>12.42</td>
<td>0.39</td>
</tr>
<tr>
<td>+ Suzie</td>
<td></td>
<td>Proposed: 37.42</td>
<td>2.29</td>
<td>0.49</td>
<td>6.88</td>
<td>0.84</td>
<td>42.48</td>
<td>0.39</td>
</tr>
<tr>
<td></td>
<td>64 kb/s</td>
<td>TMN8: 39.42</td>
<td>3.17</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>17.84</td>
<td>0.28</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Proposed: 39.98</td>
<td>2.09</td>
<td>0.56</td>
<td>9.07</td>
<td>0.75</td>
<td>82.25</td>
<td>0.32</td>
</tr>
<tr>
<td></td>
<td>128 kb/s</td>
<td>TMN8: 41.52</td>
<td>2.47</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>27.64</td>
<td>0.22</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Proposed: 42.53</td>
<td>1.61</td>
<td>1.01</td>
<td>9.04</td>
<td>0.00</td>
<td>177.79</td>
<td>0.35</td>
</tr>
<tr>
<td>Camera</td>
<td>32 kb/s</td>
<td>TMN8: 34.59</td>
<td>2.58</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>24.90</td>
<td>0.78</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Proposed: 35.04</td>
<td>1.83</td>
<td>0.45</td>
<td>3.96</td>
<td>0.53</td>
<td>71.14</td>
<td>1.48</td>
</tr>
<tr>
<td></td>
<td>64 kb/s</td>
<td>TMN8: 37.23</td>
<td>2.32</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>37.75</td>
<td>0.59</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Proposed: 37.81</td>
<td>1.51</td>
<td>0.58</td>
<td>6.03</td>
<td>0.32</td>
<td>137.79</td>
<td>1.29</td>
</tr>
<tr>
<td></td>
<td>128 kb/s</td>
<td>TMN8: 39.73</td>
<td>2.19</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>59.11</td>
<td>0.46</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Proposed: 40.22</td>
<td>0.97</td>
<td>0.49</td>
<td>7.38</td>
<td>1.66</td>
<td>303.99</td>
<td>1.37</td>
</tr>
</tbody>
</table>

Fig. 10. PSNR for Camera sequence coded at: (a) 64 and (b) 128 kbits/s (window size = 100) with pre-loading time constraint of 1 s and physical decoder buffer size of 50, 100, and 200 kbits for 32, 64, and 128 kbits/s channels, respectively.

Figs. 4–6 give the subjective evaluation of the video quality where the largest peak signal-to-noise ratio (PSNR) improvement and degradation frames in video sequences are shown. There are significant improvements in many frames using our proposed scheme, yet the degradations to the other frames are hardly noticeable. Tables I and II show the video quality results represented in PSNR (in decibels) with different sliding-window sizes ($W = 50$ and 100). Figs. 7–9 show the frame-by-frame PSNR performance. As shown in the tables, our proposed scheme achieves higher PSNR performance compared to TMN8. Larger window size tends to smooth the video quality more in a wider frame sequence range compared to a smaller window size. The maximum PSNR improvement of window-size of 100 is usually greater than that of window-size of 50; this is because more future frames are considered while performing the encoding. Also the standard deviation is smaller, which indicates more uniform video quality in the larger window size situation because more number of bits are averaged in a wider range. In both window sizes ($W = 50$ and 100), the standard deviation of PSNR is smaller than that in TMN8. Also shown in the tables, the average required buffer size is only several hundred kilobits and the required pre-loading time is less than 1.5 s, which is reasonable for real video applications.

Bit re-allocation saves bits from low-activity frames and uses these saved bits on high activity frames to produce a smoother video quality. Our proposed method also performs well at scene changes. Because the encoder has the information of
future frames, a scene change resulting in a significant content complexity gap can be detected in advance. Many bits are usually needed to encode the scene-change frames. By reducing the bits for the frames before and after the scene change, more bits can be used to encode the scene-change frame. The result shows that reducing the bits for the frames before and after the scene change does not cause significant video quality degradation, and that these saved bits can significantly improve the video quality on the scene-change frames.

In our previous simulation (where there are no constraints on decoder buffer size and pre-loading time), the Camera video sequence coded using our proposed method (with $W = 100$) will require (pre-loading time in seconds, decoder buffer size in kilobits) of (1.48, 71), (1.30, 138), and (1.37, 304) for 32, 64, and 128 kbit/s channels, respectively.

In the following simulation, we set the constraints of physical decoder buffer size to 50, 100, and 200 kbits for 32, 64, and 128 kbit/s channels, respectively.

In the following simulation, we set the constraints of physical decoder buffer size to 50, 100, and 200 kbits for 32, 64, and 128 kbit/s channels, respectively, and the pre-loading time constraint to 1 s. We use the algorithm proposed in Section V to generate the video bitstreams which satisfies these constraints (i.e., the generated bitstream requires a buffer size less than 50 kbits and less than 1-s pre-loading time). As shown in Fig. 10 and Table III, the proposed method achieves slightly less PSNR improvement; yet it satisfies the decoder buffer size and pre-loading time constraints and the results are significantly better compared to TMN8.

The sliding-window scheme gives a better PSNR during the high motion activity period. As mentioned in Kwok’s paper [24], the video degradation in a high quality (>40-dB) video frame is less perceptible. It makes sense to improve the video quality by re-allocating the bits from the frames that has very high PSNR to the low quality frames.

**VIII. CONCLUSION**

We propose a video-encoding scheme that can achieve better video quality for streaming video applications. Video qualities in high-motion-activity frames and scene-change frames are significantly better than those encoded by TMN8 encoder. Subjective evaluation also shows the improvement which can be achieved using our proposed method. Both decoder buffer-size and pre-loading time are taken into consideration in the discussion.

**TABLE III**

<table>
<thead>
<tr>
<th>Video sequence</th>
<th>Channel rate</th>
<th>Average PSNR (dB)</th>
<th>PSNR standard deviation</th>
<th>Average PSNR Improvement (dB)</th>
<th>Maximum PSNR improvement (dB)</th>
<th>Maximum PSNR degradation (dB)</th>
<th>Maximum buffer fullness (kbits)</th>
<th>Pre-loading time (second)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camera</td>
<td>32 kbit/s</td>
<td>TMN8 34.59</td>
<td>2.58</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>24.90</td>
</tr>
<tr>
<td></td>
<td>Proposed</td>
<td>35.03</td>
<td>1.95</td>
<td>0.43</td>
<td>3.84</td>
<td>0.53</td>
<td>46.68</td>
<td>0.95</td>
</tr>
<tr>
<td></td>
<td>TMN8 37.23</td>
<td>2.32</td>
<td>—</td>
<td>—</td>
<td>24.73</td>
<td>0.59</td>
<td>—</td>
<td>0.91</td>
</tr>
<tr>
<td></td>
<td>Proposed</td>
<td>37.76</td>
<td>1.65</td>
<td>0.53</td>
<td>5.58</td>
<td>0.49</td>
<td>90.48</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>TMN8 39.73</td>
<td>2.19</td>
<td>—</td>
<td>—</td>
<td>59.11</td>
<td>0.46</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>Proposed</td>
<td>40.18</td>
<td>1.18</td>
<td>0.45</td>
<td>6.45</td>
<td>1.55</td>
<td>180.30</td>
<td>—</td>
</tr>
</tbody>
</table>

**REFERENCES**


I.-Ming Pao (S’97–M’00) received the B.S. degree from National Taiwan University, Taiwan, R.O.C., in 1991, and the M.S. and Ph.D. degrees from the University of Washington, Seattle, in 1994 and 1999, respectively, all in electrical engineering.

During the summer of 1998, he was an Intern with the Digital Video Department, Sharp Laboratories of America, Camas, WA. He is currently with Protocom Corporation, Cupertino, CA. His research interests are in the areas of video compression and communication.

Ming-Ting Sun (S’79–M’81–SM’89–F’96) received the B.S. degree from National Taiwan University in 1976, the M.S. degree from the University of Texas at Arlington in 1981, and the Ph.D. degree from the University of California at Los Angeles in 1985, all in electrical engineering.

He joined the faculty of the University of Washington at Seattle in September 1996, where he is a Professor. Previously, he was the Director of the Video Signal Processing Group, Bellcore, Red Bank, NJ. He has been awarded seven patents and has published more than 100 technical papers, including several book chapters in the area of video technology. His research interests include video coding, multimedia networking, and VLSI architecture and implementation for real-time video signal processing.

Dr. Sun is currently the Editor-in-Chief of IEEE TRANSACTIONS ON MULTIMEDIA and a Distinguished Lecturer of the IEEE Circuits and Systems (CAS) Society. From 1988 to 1991, he served as the Chairman of the IEEE CAS Standards Committee and established an IEEE Inverse Discrete Cosine Transform Standard. He was the Editor-in-Chief of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY (T-CSVT) from 1995 to 1997, Conference Co-Chair of VCIP 2000, Technical Committee Co-Chair of the Workshop and Exhibition on MPEG-4 at ISCAS 2000, and Express Letter Editor of T-CSVT from 1993 to 1994. He received an Award of Excellence from Bellcore in 1987 for his work on the Digital Subscriber Line, was a co-recipient of the T-CSVT Best Paper Award in 1993, and a recipient of the IEEE CAS Golden Jubilee Medal in 2000.