Transactions Papers

A Frequency-Domain Video Transcoder for Dynamic Bit-Rate Reduction of MPEG-2 Bit Streams

Pedro A. A. Assunção and Mohammed Ghanbari, *Senior Member, IEEE*

***Abstract*—Many of the forthcoming video services and multimedia applications are expected to use preencoded video for storage and transmission. Video transcoding is intended to provide transmission flexibility to preencoded bit streams by dynamically adjusting the bit rate of these bit streams according to new bandwidth constraints that were unknown at the time of encoding. In this paper, we propose a drift-free MPEG-2 video transcoder, working entirely in the frequency domain. The various modes of motion compensation (MC) defined in MPEG-2 are implemented in the discrete cosine transform (DCT) domain at reduced computational complexity. By using approximate matrices to compute the MC–DCT blocks, we show that computational complexity can be reduced by 81% compared with the pixel domain approach. Moreover, by using a Lagrangian rate-distortion optimization for bit reallocation, we show that optimal transcoding of high-quality bit streams can produce better picture quality than that obtained by directly encoding the uncompressed video at the same bit rates using a nonoptimized Test Model 5 (TM5) encoder.**

***Index Terms*—Data compression, digital TV, frequency domain analysis, image converters, motion compensation, video signal processing.**

I. Introduction

IT is expected that many video services and multimedia applications will use preencoded video bit streams for storage and transmission. The flexibility of the present video coding algorithms, such as MPEG-2, enables the use of this standard in a great variety of applications [1], including video on demand (VoD), digital TV, and distance learning, for instance. Even though one can expect that a common communication network will be available in the near future, these services are already being deployed using the existing networks [2]–[6].

In coding of video for storage or transmission, the channel characteristics have to be assumed and given as coding parameters to the encoder. Therefore, a great lack of flexibility arises in the transmission of these bit streams regardless of the type of bit rate produced by the encoder, either constant bit rate (CBR) or variable bit rate (VBR). A CBR bit stream needs to be reduced in its bit rate when the transmission channel or the user demands a lower bit rate than that originally used to encode the video sequence. On the other hand, VBR transmission of preencoded video through asynchronous transfer mode (ATM) networks requires the bit stream to be dynamically controlled in order to minimize cell loss during congestion periods. In both cases, a video transcoder can perform the necessary traffic shaping of the preencoded bit streams according to the new bandwidth constraints or network demand.

Layered video was originally meant to solve these kind of problems [7]. For instance, the scalable modes defined in the MPEG-2 standard [8] are intended to provide support for various transmission scenarios. However, if preencoded video is used, the lack of flexibility remains since the number of different predefined layers is limited and no dynamic changes can be done on the compressed video during transmission. Moreover, for some scalable modes, such as SNR and data partitioning, only one scaled version of the bit stream can be decoded without drift, unless high-complexity encoders with several loops are used. The accumulation of error leading to drift can be a major drawback when long groups of pictures (GOP) are encoded.

In this work, transcoding is essentially regarded as a process of converting a compressed bit stream into lower rates without modifying its original structure. Simple drift-free transcoding of compressed video into lower bit rates is to decode the bit stream into reconstructed pixels and reencode them again. This method, apart from being very expensive (encoders are 10–50-fold more expensive than decoders), make the transcoding process very slow, and in the case of networking applications, the demand cannot be met on time. This is particularly significant for MPEG coded bit streams due to picture reordering and an uneven bit stream.

In recent years several techniques for bit-rate reduction of compressed video have been devised [9]–[11]. Although these techniques can be used in some applications, they cannot be regarded as generic video transcoding methods due to either their high complexity or drift in transcoded pictures. For

Manuscript received April 20, 1997; revised February 10, 1998 and July 10, 1998. The work of P. A. A. Assunção was supported by the Instituto Politécnico de Leiria-ESTG and JNICT, Praxis XXI: Sub-Programa C&T, 2° QCA, Portugal. This paper was presented in part at the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'97, Munich, Germany, April 1997. This paper was recommended by Associate Editor K. Ngan.

The authors are with the Department of Electronic Systems Engineering, University of Essex, Colchester CO4 3SQ, U.K.

Publisher Item Identifier S 1051-8215(98)09288-X.

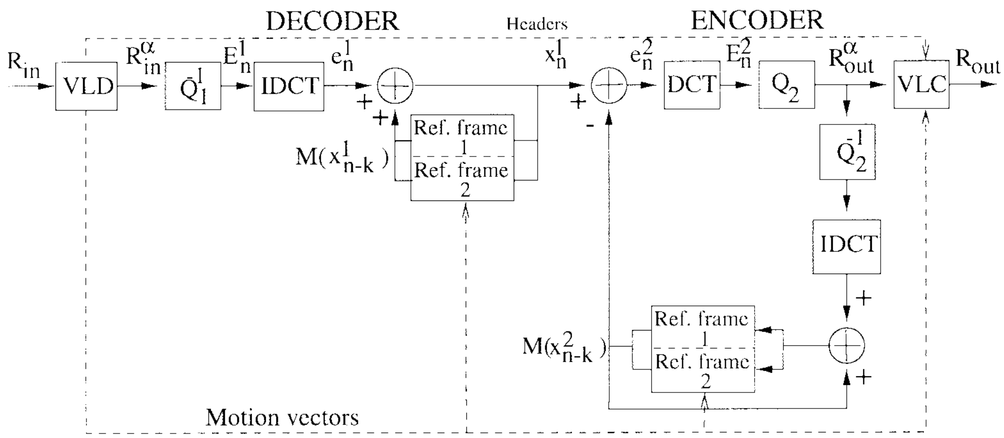


Fig. 1. Cascade of decoder–encoder.

instance, the architectures proposed in [9] are still complex since two reconstruction loops are needed, whereas the open-loop schemes of [10] and [11] introduce noticeable drift over long GOP's. In [12], we have extended the previous work of [13], and derived a simple drift-free transcoder for MPEG-2. More recently, a similar transcoder, also based on [13], was presented in [14].

In this paper, we propose a simplified and optimized transcoder based on that of [12] which is capable of outperforming those of both [12] and [14]. Its lower complexity is achieved by fully operating in the frequency domain [15], thus avoiding the implementation of both the forward discrete cosine transform (DCT) and its inverse, IDCT. A fast method for implementing motion compensation (MC) in the DCT domain (MC–DCT) is also proposed, achieving a reduction of 81% in computational complexity when compared with that of the pixel domain transcoders of [12] and [14]. Additionally, we optimize the transcoding process in a rate–distortion context by minimizing the transcoding distortion using the Lagrange multiplier method to choose the optimal set of quantizer scales for transcoding that can meet the given bandwidth constraint. We show that transcoding a high-quality bit stream into lower rates can produce better picture quality than encoding the original video at the same bit rates, using a nonoptimized Test Model 5 (TM5) encoder [16].

The paper is organized as follows. In Section II, the basic scheme for video transcoding is derived from a cascade of decoder–encoder. The MC–DCT for various MC modes defined in MPEG-2 and its fast implementation are described in Section III. Section IV describes the optimal bit reallocation procedure used for transcoding. In Section V, the simulation results of the proposed frequency-domain video transcoder are shown and discussed. Finally, Section VI concludes the paper.

II. MPEG-2 Video Transcoder

We derive a drift-free transcoder from the cascade of decoder–encoder depicted in Fig. 1. The motion estimation function of the encoder was removed, and the motion vectors (MV) of the incoming bit stream are used instead of calculating new ones. This is a significant step toward a simple architecture since motion estimation is undoubtedly the most complex function of the MPEG-2 coding algorithm. The GOP structure of the input bit stream is also kept unchanged for the sake of simplicity. Otherwise, it would be necessary to reorder the picture sequence, producing a delay of several pictures which would be unacceptable for a low-delay transcoder aimed at networking applications such as reactive congestion control in ATM networks carrying preencoded video traffic.

In the following, we will analyze transcoding of the three types of pictures, intra (I), predicted (P), and bidirectionally interpolated (B) separately. Fig. 1 is a block diagram of the cascade of decoder and encoder that can be used for three picture type bit streams. B pictures employ all parts of the codec, P pictures use only one of the two frame buffers in the feedback loop, and I pictures do not use the feedback loops at all. In the figure, $R_{\rm in}, R_{\rm out}$ are the input and output bit rates, respectively. For simplicity, we use $R_{\rm in}^{\alpha}, R_{\rm out}^{\alpha}$ with $\alpha = I, P, B$ as the corresponding nonvariable length coded of input and output bit streams for each type of picture.

A. I Pictures

The I pictures are transcoded by coarsely encoding the decoded picture x_n^1 with $Q_2 > Q_1$ (Fig. 1). Thus, the output rate for this type of picture $R_{\rm out}^I$ is given by (1)

$$R_{\rm out}^I = Q_2\left[\mathrm{DCT}\left(x_n^1\right)\right] \tag{1}$$

where x_n^1 is given by

$$x_n^1 = \mathrm{IDCT}\left[Q_1^{-1}\left(R_{\rm in}^I\right)\right] \tag{2}$$

and substituting (2) into (1), considering the orthonormality of DCT, gives the transcoding equation (3) for I frames. Note that the dc coefficients of intracoded macroblocks (MB's) always use a fixed quantization step size of 8 regardless of the actual values of Q_1 and Q_2; thus, (3) only applies to ac coefficients

$$R_{\rm out}^I = Q_2\left[Q_1^{-1}\left(R_{\rm in}^I\right)\right]. \tag{3}$$

This corresponds to coarse requantization of the ac coefficients of the input bit stream, and leads to an overall

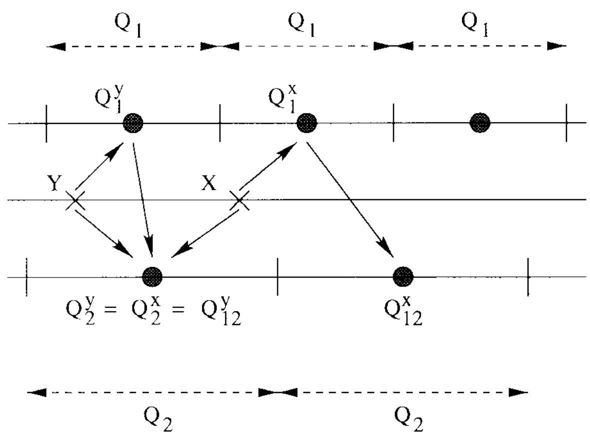


Fig. 2. Requantization of DCT coefficients.

transcoding distortion for I frames consisting of three components: 1) some nonzero coefficients of the input frame that become zero after coarse requantization; 2) the quantization error; and 3) the requantization error. While the former two are well-known causes of distortions, the latter one is not so obvious. In certain cases, requantization can lead to an additional error, which would not be introduced had the original DCT coefficients been quantized with the same coarser quantization step size. This is illustrated in Fig. 2 which shows how requantization can lead to a higher distortion than that produced by quantizing the original DCT coefficients using the same quantizer step size.

According to the figure, the reconstructed levels of the DCT coefficients X, Y, quantized with a quantizer step size of Q_1, are Q_1^X and Q_1^Y, respectively. If the coarser quantizer step size Q_2 were used, then both would be reconstructed to the same level $Q_2^X = Q_2^Y$. However, if X, Y are first quantized with Q_1 and then requantized with Q_2, the reconstruction level of Y will be the same as that of direct coarser quantization with Q_2, i.e., $Q_2^Y = Q_{12}^Y$, whereas in the case of X, the reconstruction value after requantization will be different from that of direct coarser quantization, i.e., $Q_2^X \neq Q_{12}^Y$. The requantization error of Y is zero, while that of X is not.

Hence, whenever the coarse quantization interval contains entirely the corresponding finer one, the direct coarse quantization and requantization distortions are equal. On the other hand, if the finer interval overlaps between two different coarser intervals, then the requantization distortion is larger in the case where the reconstruction value of the first quantization and the original coefficient fall into different coarser quantization intervals. Since the first quantization is performed independently of subsequent requantization, then the requantization error cannot be avoided.

The effect of the requantization error is depicted in Fig. 3 where the FLOWER sequence was encoded using $M = 3$ and $N = 15$ and fixed triplet quantizer scales of $Q_1^I = 3, Q_1^P = 4, Q_1^B = 5$. In the figure, the peak signal-to-noise ratio (PSNR) of I pictures, transcoded with $Q_2^I = 6$, is compared with that of the same pictures directly encoded using $Q_1^I = 6$ (encoded only). As the figure shows, the requantization error leads to a drop in picture quality of about 1.5 dB in all I pictures, which is significant.

B. P Pictures

In Fig. 1, the P pictures are accumulated in both MC loops as they are used for decoding and encoding subsequent pictures. If the nth incoming picture is of P type, then it needs the $(n - M)$th picture to be reconstructed, where M is the distance between anchor pictures. The input and output prediction errors e_n^1 and e_n^2, respectively, are related to the input and output rates R_{in}^P and R_{out}^P by expressions (4) and (5), respectively

$$e_n^1 = \text{IDCT}\left[Q_1^{-1}\left(R_{\text{in}}^P\right)\right] \tag{4}$$
$$R_{\text{out}}^P = Q_2\left[\text{DCT}\left(e_n^2\right)\right]. \tag{5}$$

The decoded picture x_n^1 and the corresponding prediction error e_n^2 are given by

$$x_n^1 = e_n^1 + \mathcal{M}_{\mathcal{P}}\left(x_{n-M}^1\right) \tag{6}$$
$$e_n^2 = x_n^1 - \mathcal{M}_{\mathcal{P}}\left(x_{n-M}^2\right) \tag{7}$$

where x_{n-M}^2 is the previous anchor picture decoded after coarser quantization and $\mathcal{M}_{\mathcal{P}}(\cdot)$ is the MC function for P pictures. Substituting (6) into (7), (8) is obtained

$$e_n^2 = e_n^1 + \mathcal{M}_{\mathcal{P}}\left(x_{n-M}^1\right) - \mathcal{M}_{\mathcal{P}}\left(x_{n-M}^2\right). \tag{8}$$

From (8), the prediction error e_n^2 can be obtained by adding the input prediction error e_n^1 to the difference between the input and output MC anchor pictures. This difference is actually the transcoding error introduced in each anchor picture which can be calculated prior to accumulation. Since the MV's used by both loops are the same, (8) can be simplified to (9)

$$e_n^2 = e_n^1 + \mathcal{M}_{\mathcal{P}}\left(x_{n-M}^1 - x_{n-M}^2\right). \tag{9}$$

Note, however, that, in general, MC is not a linear operation because of the integer truncation used in the loop, i.e., $\mathcal{M}_{\mathcal{P}}(z_n) - \mathcal{M}_{\mathcal{P}}(w_n) \neq \mathcal{M}_{\mathcal{P}}(z_n - w_n)$; therefore, some arithmetic inaccuracy is introduced. However, our previous experiments have shown that no significant drift in P pictures is introduced when only one buffer is used to accumulate the transcoding error [12]. Substituting (9) into (5) and taking into account the linearity of DCT, we obtain the final expression for transcoding P frames, where $E_n^1 = Q_1^{-1}(R_{\text{in}}^P)$

$$R_{\text{out}}^P = Q_2\left[E_n^1 + \text{DCT}\left(\mathcal{M}_{\mathcal{P}}\left(x_{n-M}^1 - x_{n-M}^2\right)\right)\right]. \tag{10}$$

This equation implies that, for transcoding a P picture, the accumulated transcoding error has to be added to the incoming DCT coefficients and then coarsely quantized. Since I pictures are the anchors for subsequent P pictures, their transcoding errors should also be stored, but without being added to the previous accumulated error.

Fig. 4 shows the PSNR of P pictures only, obtained by transcoding the same bit stream used in the above simulation for I pictures, using $Q_2^I = 6, Q_2^P = 8$. The PSNR of P pictures is shown for the following bit streams: 1) originally encoded using $Q_1^P = 8$ (encoded only), 2) transcoded from $Q_1^P = 4$ to $Q_2^P = 8$ using (10), and 3) transcoded $Q_1^P = 4$ to $Q_P = 8$ without taking into account the accumulated error of previous reference pictures (open loop), i.e., applying (3) to P pictures.

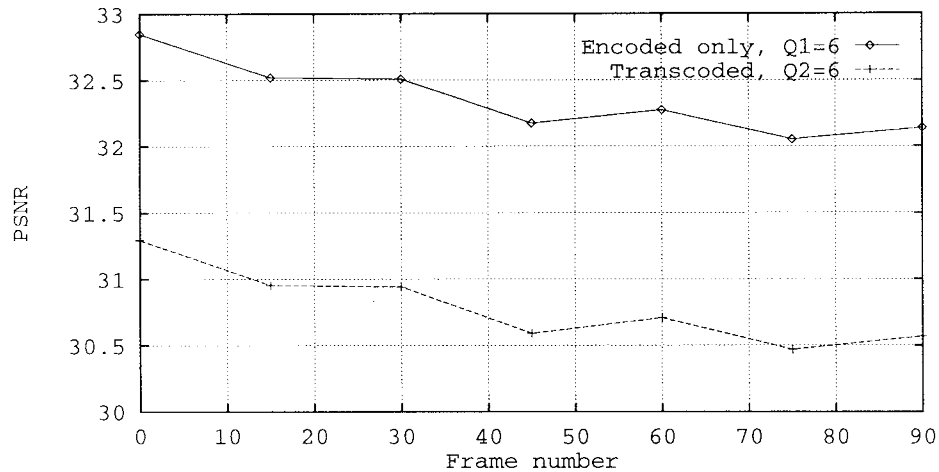


Fig. 3. Transcoding of I pictures from $Q_1^I = 3$ to $Q_2^I = 6$.

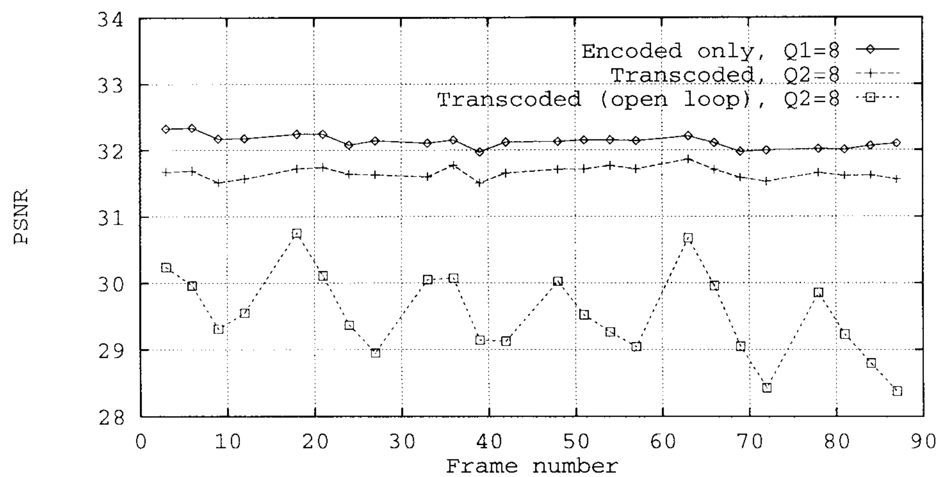


Fig. 4. Transcoding of P pictures from $Q_1^P = 4$ to $Q_2^P = 8$ $(Q_1^I = 3, Q_2^I = 6)$.

As the figure shows, not adding the MC transcoding error of previous anchor pictures produces a significant distortion, whereas the difference between encoded-only and transcoded P pictures using (10) is about 0.5 dB or less. Note that pictures 0, 15, 30, 45, 60, 75, 90 are I type (not shown in the figure) of exactly the same quality for both cases of transcoding. Note also the effect of drift due to error accumulation in the case of open-loop transcoding; the quality of the last P picture of each GOP drops about 2.5–3 dB compared with those that were transcoded through (10).

C. B Pictures

Transcoding of B pictures is just an extension of the method used for P pictures. The only difference is that MC for B pictures is related to two reference pictures. Therefore, (10) should be modified as given by (11). In this equation, $\mathcal{M}_{\mathcal{B}}(\cdot)$ is the MC operation for B pictures, and the indexes p and f refer to the past and future anchor pictures, respectively:

$$R_{\text{out}}^B = Q_2\big[E_n^1 + \text{DCT}\big(\mathcal{M}_{\mathcal{B}}\big(x_p^1 - x_p^2, x_f^1 - x_f^2\big)\big)\big]. \quad (11)$$

In order to determine R_{out}^B, two buffers are needed, one for each anchor picture since the accumulated transcoding errors of both have to be present to keep a track of drift in B pictures. However, since B pictures are not used as references for further prediction, the system could be simplified by eliminating one buffer. In this case, B pictures are treated similarly to the I pictures, and the transcoding follows (3).

Fig. 5 shows the PSNR of B pictures only, obtained by transcoding the same bit stream from $Q_1^I = 3, Q_1^P = 4, Q_1^B = 5$ to $Q_2^I = 6, Q_2^P = 8, Q_2^B = 10$. Equations (3) and (10) were used to transcode I pictures and P pictures, respectively. The PSNR is shown for the following bit streams: 1) originally encoded with $Q_1^I = 6, Q_1^P = 8, Q_1^B = 10$ (encoded only); 2) transcoded from $Q_1^B = 5$ to $Q_2^B = 10$ using (11); and 3) transcoded from $Q_1^B = 5$ to $Q_2^B = 10$ without taking into account the accumulated error of the anchor pictures (open loop), i.e., applying (3) to B pictures.

As Fig. 5 shows, for these types of pictures, the difference between transcoded using (11) and encoded only is about 0.25 dB or less. Note that the effect of not adding the accumulated transcoding error of the anchor pictures to B pictures, i.e., the case of open-loop transcoding, produces a drop in

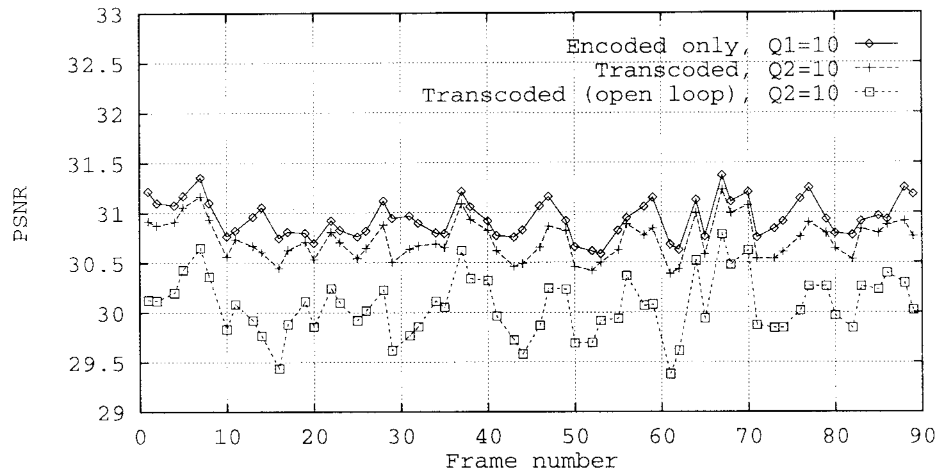


Fig. 5. Transcoding of B pictures from $Q_1^B = 5$ to $Q_2^B = 10$ ($Q_1^I = 3, Q_1^P = 4, Q_2^I = 6, Q_2^P = 8$).

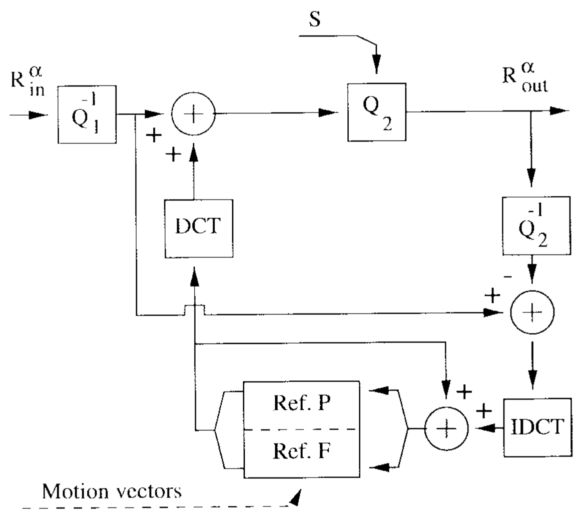


Fig. 6. Pixel-domain transcoder.

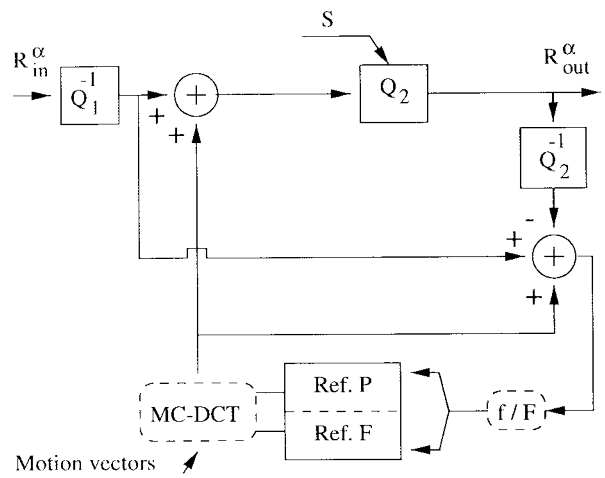


Fig. 7. DCT-domain transcoder.

quality of about 1 dB. Note also that P and I pictures (not shown in the figure) have exactly the same quality in both cases of transcoding.

D. Pixel-Domain Transcoder

Excluding the VLD and VLC functions, (3), (10), and (11) indicate that Fig. 1 can be simplified to Fig. 6. This makes use of the linearity of DCT where the transcoding error (difference between input and output) is evaluated in the DCT domain, and thus only one DCT/IDCT is needed. By using the MV's of the incoming bit stream and just one reconstruction loop, this transcoder is much simpler than the cascade of decoder–encoder in Fig. 1. In our previous work presented in [12], with a different approach of using step-by-step simplification, we showed how a cascade of decoder–encoder can be simplified into the one shown in Fig. 6.

In the transcoder of Fig. 6, the DCT and IDCT functions are only necessary because MC is primarily defined as a pixel-domain operation; otherwise, this transcoder could work entirely in the DCT domain. We refer to the transcoder of Fig. 6 as a pixel-domain transcoder. In order to improve the efficiency and at the same time reduce the computational complexity of the pixel-domain transcoder, we have derived a fully frequency-domain transcoder with an optimized bit reallocation process, as described in the next sections.

III. DCT-Domain Transcoder

If MC can be performed in the frequency domain, there is no need for the DCT and the IDCT of Fig. 6, and thus the transcoder can be further simplified. In the new scheme, shown in Fig. 7, the transcoding error, given by the difference between the inverse quantized input and output DCT coefficients, is accumulated in the DCT domain and added to the DCT of the current picture after motion compensation in the DCT domain (MC–DCT). The dashed blocks of Fig. 7 represent two functions that support the MPEG-2 syntax: 1) field/frame DCT conversion (f/F), and 2) the various modes of MC defined for MPEG-2 [8]. Since the transcoding error of frame pictures is accumulated in frame format, whenever an MB of a frame picture is *field DCT* coded, the f/F function converts it into *frame DCT* format.

In the next subsection, we describe a fast method to implement MC–DCT with reduced computational complexity. In order to compare our results with previous work, we use the same concept of computational complexity used in [17] which is the number of basic integer operations such as division by a power of 2 (*shift*) and addition (*add*).

A. Motion Compensation in the DCT Domain

The basic MC operation consists of extracting pixel blocks from a reference picture by shifting the horizontal and vertical positions of the current blocks a number of pixels dictated by the corresponding MV's. In general, neither the vertical nor the horizontal MV is an integer multiple of the block size; thus, the displaced block intersects four neighboring blocks $\boldsymbol{b_i}, i = 1, \cdots, 4$ of the reference picture. Therefore, the MC block $\hat{\boldsymbol{b}}$ comprises four pixel subblocks, one from each intersected block $\boldsymbol{b_i}$. These subblocks can be extracted from the respective blocks by multiplying the latter with the appropriate matrices $\boldsymbol{h_{hi}}, \boldsymbol{h_{wi}}$ that perform window and shift operations as described in previous work [18], [19]. The number of rows (h) and columns (w) that each block $\boldsymbol{b_i}$ is intersected by the MC block $\hat{\boldsymbol{b}}$, i.e., the size of each subblock, defines which matrices should be applied for each $\boldsymbol{b_i}$. This operation, in the pixel domain, is given by

$$\hat{\boldsymbol{b}} = \sum_{i=1}^{4} \boldsymbol{h_{hi}} \cdot \boldsymbol{b_i} \cdot \boldsymbol{h_{wi}}, \qquad \boldsymbol{1 \leq h, w \leq 7}. \tag{12}$$

The matrices $\boldsymbol{h_{hi}}, \boldsymbol{h_{wi}}$ have the structure of $\boldsymbol{u_h}, \boldsymbol{l_w}$, where I_h, I_w are identity submatrices of size $h \times h$ and $w \times w$, respectively

$$\boldsymbol{h_{h1}} = \boldsymbol{h_{h2}} = \boldsymbol{u_h} \equiv \begin{bmatrix} 0 & I_h \\ 0 & 0 \end{bmatrix} \tag{13}$$

$$\boldsymbol{h_{w1}} = \boldsymbol{h_{w3}} = \boldsymbol{l_w} \equiv \begin{bmatrix} 0 & 0 \\ I_w & 0 \end{bmatrix} \tag{14}$$

and

$$\boldsymbol{h_{h3}} = \boldsymbol{h_{h4}} = \boldsymbol{u}_{8-\boldsymbol{h}} \tag{15}$$

$$\boldsymbol{h_{w2}} = \boldsymbol{h_{w4}} = \boldsymbol{l}_{8-\boldsymbol{w}}. \tag{16}$$

By applying the distributive property of matrix multiplication with respect to DCT, one can use the DCT matrices $H_{hi} \equiv \mathrm{DCT}(\boldsymbol{h_{hi}}), \boldsymbol{H_{wi}} \equiv \mathbf{DCT}(\boldsymbol{h_{wi}})$ to extract the MC–DCT block $\hat{B} \equiv \mathrm{DCT}(\hat{\boldsymbol{b}})$ directly from the DCT blocks $B_i \equiv \mathrm{DCT}(\boldsymbol{b_i})$ of the reference picture. This leads us to the general equation (17) of MC–DCT, where the matrices H_{hi}, H_{wi} are constant so they can be precomputed and stored in a memory. Taking into account that $\boldsymbol{u_n}$ is the transpose of $\boldsymbol{l_m}$ for $m = n$, and thus their transforms $U_n \equiv \mathrm{DCT}(\boldsymbol{u_n}), L_m \equiv \mathrm{DCT}(\boldsymbol{l_m})$ are also transposed, only seven (instead of 14) different matrices need to be stored.

$$\hat{B} = \sum_{i=1}^{4} H_{hi} \cdot B_i \cdot H_{wi}, \qquad 1 \leq h, w \leq 7. \tag{17}$$

TABLE I
MOTION COMPENSATION IN MPEG-2

Current picture	Pred. mode	Ref. picture (transcoding error)
frame	frame	frame
		field
	field	frame
		field
	dual prime	frame
		field
field	field	frame
		field
	dual prime	frame
		field
	16x8	frame
		field

B. Specific Functions of MPEG-2

We aim for a compatible MPEG-2 video transcoder; therefore, all MC modes defined in the standard should be supported by the frequency-domain transcoder. These can be implemented by using the same property of DCT as above and the appropriate matrices. In the following, we show how to support *field/frame DCT* coding, *half-pixel accuracy,* and *field MC* [8]. Furthermore, several combinations of different types of current/reference pictures and MC modes are possible to occur in the transcoding of generic MPEG-2 bit streams. The current and reference pictures can be either *frame* or *field* using the MC modes of *frame, field,* 16 $\times$ 8, *dual prime*. These combinations are shown in Table I.

In the scheme of Fig. 7, the transcoding error is accumulated either as a frame or field picture according to the type of picture being currently transcoded. In order to keep the same DCT format for all MB's of the same picture, *field/frame DCT* conversion is necessary to convert the MB's *field DCT* coded into *frame DCT* format before storing the transcoding error of frame pictures. This is implemented by the function f/F as described in the following subsection.

1) Field DCT Coding: If an MB of a frame picture is *field DCT* coded, then each DCT block of that MB comprises frequency information from one field only. In this case, the *field DCT* MB is converted into *frame DCT* format by generating DCT blocks of coefficients from both fields.

Any two vertically aligned *frame DCT* blocks $B_k^*, k = 1, 2,$ can be obtained directly in the DCT domain from the corresponding *field DCT* blocks $B_i, i = 1, 2$ by applying (18). The matrices F_{ki} are the DCT transforms of those that perform the equivalent operation in the pixel domain. A description of these operations is given in Section A of the Appendix

$$B_k^* = \sum_{i=1}^{2} F_{ki} \cdot B_i, \qquad k = 1, 2. \tag{18}$$

Note that such an operation is not necessary in field pictures, and in frame pictures, it is not likely to be needed for all

TABLE II
MATRIX OPERATIONS TO CALCULATE FOUR MC–DCT BLOCKS WITH HALF-PIXEL MV

Method	Multiplications	Additions
brute force	96	60
filtering	36	16

blocks since *field/frame DCT* coding is a decision taken at the MB level. Also, if all MB's of a frame picture are *field DCT* coded, then no conversion is necessary since this is equivalent to having two separate field pictures.

2) Half-Pixel Accuracy: When an MV with half-pixel precision is used, either two or four pixels are needed to calculate the actual prediction of one single pixel. In terms of blocks, this is equivalent to computing the average, for each pixel, of either two or four blocks. In the DCT domain, this needs the extraction of two or four blocks from the reference picture; hence, (17) is applied either twice or four times to obtain the final predicted block. Since the blocks involved in half-pixel prediction are displaced from each other only by one pixel, the extraction of four different DCT blocks can be avoided by applying a linear filter to the MC–DCT MB reconstructed with the integer component of the motion vector. Vertical filtering should be used when half-pixel accuracy exists in the vertical direction, while horizontal filtering is used when half-pixel accuracy exists in the horizontal direction.

Let us consider the DCT blocks $B_i, i = 1, \cdots, 4$ of an MC–DCT luminance MB ordered according to their location within the MB (top left: B_1, top right: B_2, bottom left: B_3, bottom right: B_4). Suppose also that all of these blocks are spatially adjacent in the pixel domain. The horizontally filtered blocks B_i^h, in the DCT domain, are obtained from (19), while the vertically filtered blocks B_i^h are obtained from (20). The filter coefficient matrices F_i^h, F_i^v of these equations are defined in Section B of the Appendix. In the case of both horizontal and vertical half-pixel prediction, vertical filtering should follow the horizontal filtering or vice versa

$$B_i^h = \begin{cases} B_i F_1^h + B_{i+1} F_2^h, & i = 1,3 \\ B_i F_3^h, & i = 2,4 \end{cases} \tag{19}$$

$$B_i^v = \begin{cases} F_1^v B_i + F_2^v B_{i+2}, & i = 1,2 \\ F_3^v B_i, & i = 3,4. \end{cases} \tag{20}$$

The use of this method introduces some distortion in those blocks located on the right and bottom boundaries of the MB's since the predictions of these blocks do not take into account the blocks belonging to the adjacent MB's. Nevertheless, as the simulation results show, this is a minor error, and a significant reduction in computational complexity can be achieved if those blocks are not extracted. For example, in order to extract one MC–DCT luminance MB (four blocks) using half-pixel accuracy in both directions, the number of matrix multiplications and additions of the brute-force method (17) is quite different from that of filtering, as shown in Table II.

Note that for calculating one MC–DCT block with an MV of half-pixel accuracy in both directions using the brute-force method, four blocks need to be extracted through (17); thus, a total of 16 blocks needs to be extracted to generate only one luminance MB.

3) Field Prediction: In the case of field and frame pictures being encoded into the same bit stream, e.g., interlaced video, the MB's of frame pictures can be predicted from two reference fields, which in turn could have been encoded either as frame or field pictures. Also, MB's of field pictures may be predicted from frame pictures. Since the transcoding error is always stored in the format of the current input picture (either frame or field), conversion from field to frame and vice versa is necessary for interlaced sequences. Therefore, the frequency domain transcoder should be capable of generating, in the DCT domain, 1) field blocks from frame pictures, and 2) frame blocks from field pictures. The former is obtained by extracting the MC–DCT frame blocks B_i^* that contain the field blocks B_k, and then applying (21) to obtain B_k from B_i^*. M_i are constant matrices as described in Section C of the Appendix. The latter is obtained by applying (18) since the operation is the same as for converting a *field DCT* coded MB into a *frame DCT* coded one

$$B_k = \sum_{i=1}^{2} M_{ki} \cdot B_i^*, \qquad k = 1, 2. \tag{21}$$

Another possibility is the occurrence of field prediction when both the current and reference pictures are frame pictures. In this case, the field blocks have to be first extracted using (21), and the final prediction in frame format is obtained by applying (18) to the extracted blocks. These two equations can still be merged together, thus generating a new set of constant matrices that allows the implementation of the two operations in a single step.

4) 16 × 8 and Dual Prime: In the 16 × 8 MC mode, the only difference is that two, instead of one, MV's are used for each MB, one for the 16 × 8 upper region and another for the lower 16 × 8 region. In dual-prime mode, either two or four predictions are used for each block of a field or frame picture, respectively. Therefore, these cases are supported by the equations derived above, and no further development is necessary.

C. Computational Complexity

Most of the computational complexity of the MC–DCT method comes from (17). In fact, the brute-force computation of (17) in the case where the MC block is not aligned in any direction with the block structure requires six matrix multiplications and three matrix additions using floating-point arithmetic. Therefore, we aim to reduce the number and complexity of the operations involved in solving (17) such that the frequency-domain transcoder of Fig. 7 becomes less complex than its pixel domain counterpart of Fig. 6. The computational complexity is measured using the definition of [17], i.e., the number of integer additions and shifts-right required to calculate one MC block.

In order to achieve reduced computational complexity, we approximate the elements of H_{wi}, H_{hi} to binary numbers with a maximum distortion of 1/32. By approximating all of the constant matrices in a similar way, only basic integer operations, such as *shift-right* and additions (*add*), are needed to solve (17). As an example, we show a number of elements of matrix L_5

$$L_5 = \begin{bmatrix} 0.62500 & 0.41858 & -0.16332 & \cdots & 0.08326 \\ -0.41858 & -0.06260 & 0.49812 & \cdots & -0.11549 \\ -0.16332 & -0.49812 & -0.56694 & \cdots & 0.13641 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -0.08326 & -0.11549 & -0.13641 & \cdots & -0.54095 \end{bmatrix}. \quad (22)$$

Each element of the rounded matrix, in terms of powers of 2

$$\hat{L}_5 = \begin{bmatrix} \frac{1}{2}+\frac{1}{8} & \frac{1}{2}-\frac{1}{16} & -\frac{1}{8}-\frac{1}{16} & \cdots & \frac{1}{16} \\ \frac{1}{16}-\frac{1}{2} & -\frac{1}{16} & \frac{1}{2} & \cdots & -\frac{1}{8} \\ -\frac{1}{8}-\frac{1}{16} & -\frac{1}{2} & -\frac{1}{2}-\frac{1}{16} & \cdots & \frac{1}{8} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\frac{1}{16} & -\frac{1}{8} & -\frac{1}{8} & \cdots & -\frac{1}{2}-\frac{1}{16} \end{bmatrix}. \quad (23)$$

Since the elements of DCT blocks lie in the range [−2048, 2047], shifting the actual values would result in zero for most elements. In order to maintain precision in the intermediate operations, the transcoding error of each DCT coefficient is multiplied by 2^8 and stored in this format. This is, in fact, a scaling factor which can be included in the quantization and inverse quantization functions without introducing additional complexity.

Furthermore, by employing simple data manipulation, the multiplication of the constant matrices by the DCT blocks can be implemented with a minimum number of operations. If all DCT coefficients of the same column that are *right-shifted* by k (multiplied by 2^{-k}) are added together before shifting, then the number of *shift* operations will be reduced. This is explained in the following example using the third line of the above matrix, where both (24) and (25) need five additions, but the number of *shifts* is quite different; (24) requires six *shifts*, while (25) is implemented with three *shifts* only

$$y = -\left(\tfrac{1}{8}+\tfrac{1}{16}\right)x_1 - \tfrac{1}{2}x_2 - \left(\tfrac{1}{2}+\tfrac{1}{16}\right)x_3 + \tfrac{1}{8}x_8 \quad (24)$$
$$= \tfrac{1}{2}(-x_2 - x_3) + \tfrac{1}{8}|;(x_8 - x_1) + \tfrac{1}{16}(-x_1 - x_3). \quad (25)$$

In order to calculate the number of operations required to implement (17) using the approximate matrices, the following have to be taken into account.

- The number of matrix multiplications of (17) can be reduced to six, instead of eight, taking into account that $H_{h1} = H_{h2}$ for all cases.
- The number of operations required for premultiplication with any of the H_{wi}, H_{hi} matrices is the same as for postmultiplication.
- There are 14 matrices, but since seven of them are the transposes of the other 7, only seven matrices need to be taken into account.

The number of operations (*shift, add*) required for each matrix is depicted in Table III, and the total number of operations for extracting one MC–DCT basic block is given in Table IV, where the various cases correspond to the possible combinations of intersections between the MC block and the block structure of the reference picture. The table entries correspond to the width and height (in number of pixels) of the top-left subblock.

TABLE III
NUMBER OF OPERATIONS FOR EACH MATRIX

Matrix (n)	No of shift	No of add
1	17	65
2	24	67
3	26	78
4	28	66
5	29	73
6	30	67
7	29	87

TABLE IV
TOTAL NUMBER OF OPERATIONS TO EXTRACT ONE BLOCK NOT ALIGNED WITH THE BLOCK STRUCTURE FOR ALL COMBINATIONS OF w_1, h_1

h_1	w_1							
	0	1	2	3	4	5	6	7
0	0	263	252	270	252	270	252	263
1	263	786	766	802	766	802	766	786
2	252	776	756	792	756	792	756	776
3	270	794	774	**810**	774	**810**	774	794
4	252	776	756	792	756	792	756	776
5	270	794	774	**810**	774	**810**	774	794
6	252	776	756	792	756	792	756	776
7	263	786	766	802	766	802	766	786

In order to measure the computational complexity of the proposed fast implementation, we consider the worst case in terms of the number of operations. This is for $H_{hi}, H_{wi} \in \{U_3, L_5, U_5, L_3\}$ corresponding to the cases of intersection specified in Table IV. Considering that all block coefficients are nonzero, the total number of operations for extracting one block is 810. The fast MC–DCT algorithm recently proposed by Merhav and Bhaskaran [17] is now used for comparison with the results achieved. For its worst case, their algorithm requires 2928 operations of the same type (shift-right and addition) to perform the extraction of one MC–DCT block. Therefore, the method proposed here has only 28% of the computational complexity of their method. Assuming a uniform probability distribution on the pairs (w, h) shown in Table IV, then the average number of operations to extract one block using approximate matrices is 654, while that of the algorithm cited above is 2300, which corresponds to the same reduction ratio of computational complexity as for the worst case.

On the other hand, if the equivalent operation in the pixel domain had been used, then it would have been necessary to perform the IDCT of the four intersected blocks, cut the appropriate MC block in the pixel domain, and then transform it back with DCT. Using the fast algorithm for the eight-point DCT proposed in [20] as the reference for accounting the number of operations, the pixel domain approach gives a total of 4320 operations (see the most recent published work by the same authors for a complete description of the method used to determine the number of operations of DCT and IDCT [21]). Therefore, comparing the pixel domain approach with the frequency-domain one, the fast MC–DCT method using approximate matrices provides a reduction of 81% in computational complexity, whereas that of the algorithm proposed in [17] gives a reduction of 32%. Note that this is a worst case comparison since the reference for accounting complexity of DCT/IDCT for both algorithms is the fastest existing algorithm for eight-point DCT (see also [22]). Had a less efficient DCT algorithm been used, the result would have been even more impressive. Moreover, if the block to be predicted is aligned either horizontally or vertically, then only two rather than four blocks from the reference frame are needed, and for perfect alignment (e.g., both horizontally and vertically), (17) is simplified to $\hat{B} = B_1$.

Note, however, that the number of matrix operations in the DCT domain for a whole picture is very dependent on the motion activity of the video sequence, and also on some coding parameters such as field/frame motion, for instance. The MC type, the percentage of nonzero MV's, and the number of MV's which are multiples of the block size have a major influence on the amount of computations required. This leads to a variable number of matrix operations required for MC–DCT, and thus the average computational complexity can be much less than what is shown above for the worst case. The influence of most of these aspects in the computational complexity of MC–DCT was addressed in [19].

IV. Bit Reallocation

Since the transcoded bit stream should comply with the constraint $0 < S \leq 1$ (Fig. 7), a bit-reallocation process should take place in the transcoder. This is implemented by choosing new quantizer scales for transcoding each MB or group of MB's such that the output rate does not exceed the given constraint. The optimum set of quantizer scales for a group of MB's would produce a minimum average distortion and comply with rate constraint. Such quantizers can be found by using the Lagrange multiplier method [23].

A. Lagrangian Optimization

The problem of optimal transcoding is to find a set of quantizer step sizes for a group of N MB's $\{q_1, q_2, \cdots, q_N\}$ such that the average distortion $\mathcal{D}$ is minimized and the total rate $\mathcal{R}$ complies with a given target posed by the constraint $S, \mathcal{R} \leq R_T = S \cdot R_{\text{in}}^{\alpha}$ with $\alpha = I, P, B$. This can be formulated as

$$\min \mathcal{D}, \qquad \text{subject to} \quad \mathcal{R} \leq R_T \tag{26}$$

with $\mathcal{D}$ and $\mathcal{R}$ given as

$$\mathcal{D} = \sum_{k=1}^{N} d_k(q_k) \quad \mathcal{R} = \sum_{k=1}^{N} r_k(q_k)$$

where $d_k(q_k)$ and $r_k(q_k)$ are the distortion and rate of the kth MB after transcoding with a quantizer scale q_k.

The constrained problem of (26) can be solved by converting it into the unconstrained problem of (27) where rate and distortion are merged through a Lagrange multiplier $\lambda \geq 0$ [23]. The main advantage of solving (27) instead of (26) is that the Lagrangian costs $J_k(\lambda)$ for each MB k are independently calculated

$$J_k(\lambda) = \min_{q_k} \{d_k(q_k) + \lambda r_k(q_k)\}, \qquad \text{for} \quad k = 1, 2, \cdots, N. \tag{27}$$

Let $(r_k^*(\lambda), d_k^*(\lambda))$ be the solution to the minimum Lagrangian cost $J_k(\lambda)$ for MB k, and let $q_k^*(\lambda)$ be the corresponding quantizer step size. For any $\lambda \geq 0$, the optimal solution $(\mathcal{R}^*(\lambda), \mathcal{D}^*(\lambda))$ is given as the sum of the solutions $(r_k^*(\lambda), d_k^*(\lambda))$ for $k = 1, 2, \cdots, N$. If, for a particular value of $\lambda = \lambda_s$, the total rate happens to be equal to the given constraint, i.e., $\mathcal{R}^*(\lambda_s) = R_T$, then the set $\{q_1^*(\lambda_s), q_2^*(\lambda_s), \cdots, q_N^*(\lambda_s)\}$ is the optimal set of quantizer scales to be used for transcoding. Therefore, the optimal value λ_s has to be found for each group of N MB's. We have implemented a fast search method based on a bisection algorithm. This algorithm was adapted from the method used in [24].

Since predicted MB's are added to the MC transcoding error of their respective reference pictures prior to transcoding (refer to Fig. 7), the distortion $d_k(q_k)$ should take into account the type of MB being transcoded. We have measured the distortions according to (28) for the three types of MB's: 1) intra, 2) predicted either forward or backward, and 3) interpolated, where L is the number of blocks in the MB and $C_k^{ni}(q_k')$ is the ith inverse quantized DCT coefficient of block n in MB k. The quantizer scales q_k' and q_k are those of the input and transcoded MB's, respectively, while X_k^{ni}, Y_k^{ni} correspond to the accumulated MC–DCT transcoding error of the anchor pictures x, y, respectively. Note that this measure of distortion does not add any additional complexity to the architecture of Fig. 7 since the transcoding error has already been accumulated for drift compensation

$$d_k(q_k) = \frac{1}{L^2} \sum_{n=0}^{L-1} \sum_{i=0}^{63} \cdot \begin{cases} [C_k^{ni}(q_k) - C_k^{ni}(q_k')]^2 \text{ (intra)} \\ [C_k^{ni}(q_k) - C_k^{ni}(q_k') + X_k^{ni}]^2 \text{ (forw, back)} \\ \left[C_k^{ni}(q_k) - C_k^{ni}(q_k') + \dfrac{X_k^{ni} + Y_k^{ni}}{2}\right]^2 \text{ (interp).} \end{cases} \tag{28}$$

V. Simulation Results

In the experiments carried out to evaluate the performance of the proposed frequency-domain transcoder, we have used two standard sequences (SIF format) with different motion characteristics: MOBILE and MUSIC. The former is relatively slow in motion, whereas the latter has high motion, and both were encoded using frame motion estimation with MV's of half-pixel precision. All operations in the DCT domain were performed using fast computation with approximate matrices.

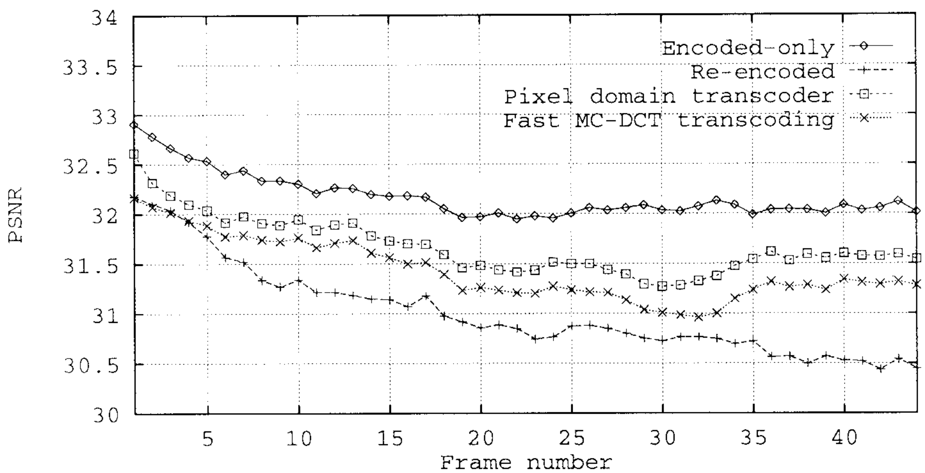


Fig. 8. Effect of matrix approximation.

The following characteristics of the optimized frequency-domain transcoder were evaluated.

- The drift introduced in transcoded pictures by using approximate matrices for fast computation. The results of the pixel-domain transcoder are compared with those of the frequency-domain transcoder.
- The overall efficiency of transcoding is compared with that of a standard encoder and a cascade of decoder–encoder.
- The additional distortion introduced in B pictures when the accumulated MC–DCT transcoding error of the anchor pictures is not added to the transcoded B pictures.
- The effect of one frame and one slice transcoding delay on the picture quality.

A. Picture Drift of Fast MC–DCT

In order to evaluate the picture drift of the fast MC–DCT, we have encoded a high-quality bit stream using the MOBILE sequence with a fixed quantizer step size of $Q_1 = 8$. Since drift accumulates in P pictures only, no B pictures were encoded $(M = 1)$. This bit stream was then transcoded using three different architectures at fixed $Q_2 = 14$: 1) reencoding; 2) transcoding using the pixel-domain scheme; and 3) transcoding using the fast MC–DCT method. For comparison, we also show the picture quality of the same sequence directly from the uncompressed video encoded using $Q_1 = 14$ (encoded only).

As Fig. 8 shows, the difference in PSNR between transcoded pictures using the pixel-domain transcoder and those using the proposed fast MC–DCT is about 0.2 dB, while the computational complexity of the fast implementation is 81% less than that of the pixel-domain approach. Note that transcoding always gives better performance than reencoding. This is because transcoding uses the MV's of the incoming bit stream, whereas in reencoding, new MV's are calculated based on the poorer quality of decoded pictures. We have also found that the frequency-domain transcoder using the brute-force method with floating-point arithmetic in all operations produces the same picture quality as the pixel-domain transcoder. Hence, the above difference of 0.2 dB between fast MC–DCT and pixel-domain transcoders is due to matrix approximation. Note that transcoding gives worse performance than encoded only, which is due to the requantization error as explained earlier.

B. Transcoding Performance

In this section, we compare the overall efficiency of the frequency-domain transcoder using the optimal bit reallocation algorithm described in Section IV with reencoding (cascade of decoder–encoder) and encoding only using the bit allocation method of TM5 [16]. The reason for not optimizing the standard encoder is that the MPEG-2 encoding algorithm is already a complex system which requires heavy processing, and the optimization algorithm also requires significant processing; hence, a standard encoder may not have sufficient processing power to support such an implementation. For this reason, various authors have recently proposed suboptimal schemes for MPEG encoders to reduce the complexity of optimization at the expense of a lower quality [25], [26]. On the other hand, we have shown that the frequency-domain transcoder is far less complex than an encoder; hence, extra processing power is available to make the transcoder more efficient by using the Lagrangian rate–distortion optimization algorithm. Note that since we are comparing an optimized transcoder with a nonoptimized TM5 encoder, for high-quality bit streams (i.e., high bit rate–low distortion), transcoded pictures can even be better than those of encoded only.

In order to show the overall efficiency of the optimized frequency-domain transcoder, the two sequences were encoded at 4 Mbit/s with a standard TM5 MPEG-2 encoder using a GOP structure of $N = 15, M = 3$. The bit stream was then transcoded into lower rates, and the resultant picture quality was compared with that of a fully reencoded (cascade of decoder–encoder) bit streams at the same lower rates. Also, the original image sequence was directly encoded (encoded only) at the lower rates. The bit-rate constraints used for transcoding were set to the bit rates of encoded-only bit streams.

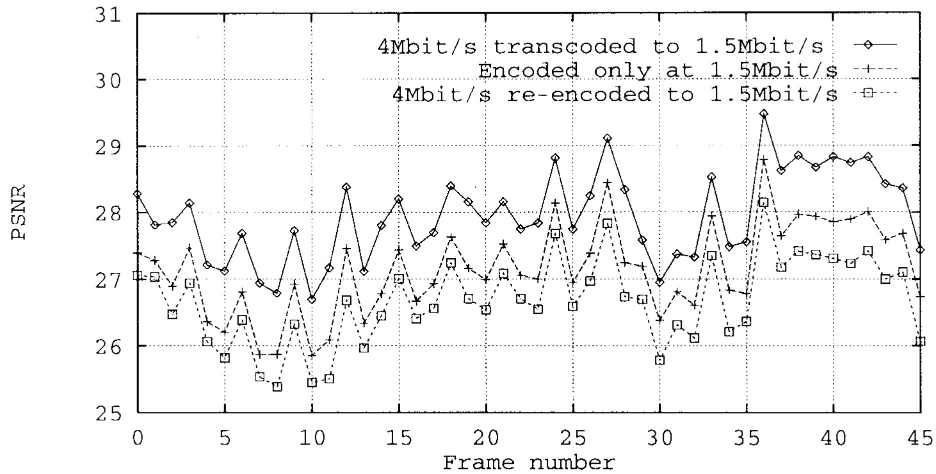


Fig. 9. Transcoding from 4 to 1.5 Mbit/s.

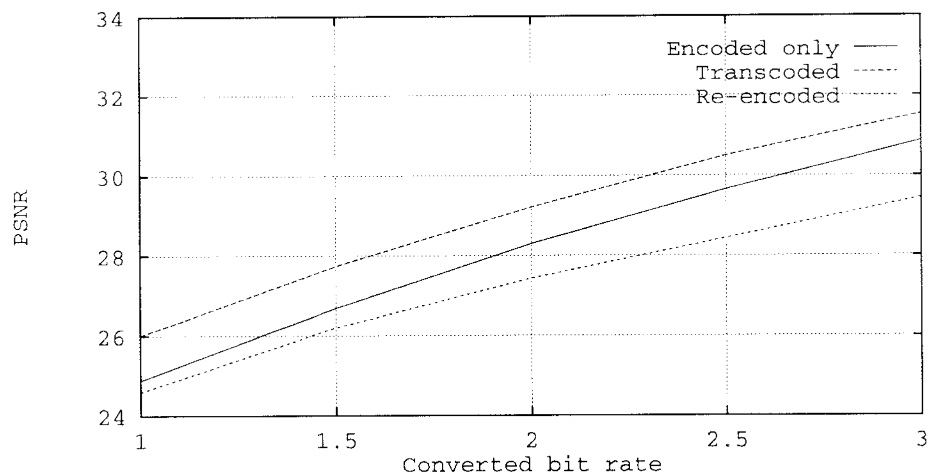


Fig. 10. Transcoding of MOBILE sequence (average PSNR).

Fig. 9 shows the PSNR of the first 45 pictures of the MOBILE sequence for the case where the 4 Mbit/s bit stream was converted into 1.5 Mbit/s using the Lagrangian optimization over an entire frame. As shown in the figure, the optimized frequency-domain transcoder outperforms both the cascaded decoder–encoder, using the same input bit streams, and the TM5 encoder, using the original uncompressed pictures, at the same bit rate. Note that while in Fig. 3 transcoded I pictures suffer about 1.5 dB loss compared to encoded only, the optimized transcoder does not have such a deficiency, and is consistently better than both encoded-only and reencoded ones. Figs. 10 and 11 show the average PSNR of 120 pictures using the 4-Mbit/s bit stream for transcoding at several reduced bit rates.

Again, the optimized frequency-domain transcoder outperforms the TM5 MPEG-2 encoder and the cascade of a TM5 decoder–encoder for all of the bit rate conversion ratios, in particular at higher compression ratios. This is because the bit stream used for transcoding is of high quality, and the rate–distortion optimization is still capable of allocating the same number of output bits more efficiently than the TM5 model used by the standard encoder, despite the use of uncompressed video in TM5.

Note, however, that encoded-only and reencoded bit streams use a nonoptimized algorithm as defined by TM5, whereas optimally transcoded bit streams are obtained from a cascade of a nonoptimized encoder introducing low distortion (4 Mbit/s is a high bit rate for SIF pictures) and an optimal transcoder. Overall, the distortion introduced by this cascade is shown to be less than the distortion introduced by a single nonoptimized encoder. Had the encoder used the same optimization algorithm, the encoding only would have performed better than transcoding. In this case, the relative improvement of encoding only over transcoding would be similar to that of Fig. 8 where the same quantization method was used for both.

These results show that video can be stored in compressed format, and further compressed at the time of transmission through channels with lower bandwidth. The complexity of transcoding is less than that of encoding the original pictures, and far less than that of reencoding; thus, many services and applications can benefit from such a transcoder.

C. Effect of No Drift Compensation on B Pictures

Since MC for B pictures is the most complex (two anchor pictures are required), avoiding use of MC–DCT for B pic-

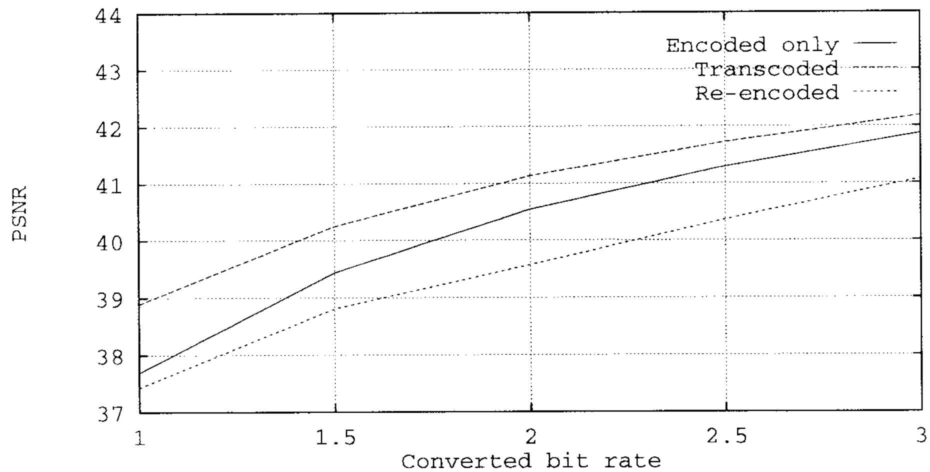


Fig. 11. Transcoding of MUSIC sequence (average PSNR).

TABLE V
PSNR OF B PICTURES TRANSCODED WITH (B-ON) AND WITHOUT (B-OFF) DRIFT COMPENSATION (MOBILE)

Mbit/s	B-on [dB]	B-off [dB]
1.0	25.896	25.809
1.5	27.543	27.463
2.0	28.917	28.864
2.5	30.125	30.092
3.0	31.076	31.052

TABLE VI
PSNR OF B PICTURES TRANSCODED WITH (B-ON) AND WITHOUT (B-OFF) DRIFT COMPENSATION (MUSIC)

Mbit/s	B-on [dB]	B-off [dB]
1.0	38.512	38.343
1.5	39.793	39.716
2.0	40.638	40.601
2.5	41.229	41.205
3.0	41.703	41.686

TABLE VII
PSNR FOR TRANSCODING MOBILE USING DELAYS OF ONE FRAME AND ONE SLICE

Mbit/s	frame [dB]	slice [dB]	diff. [dB]
1.0	26.00	25.90	0.32
1.5	27.74	27.62	0.34
2.0	29.21	29.03	0.53
2.5	30.50	30.28	0.53
3.0	31.53	31.34	0.45

TABLE VIII
PSNR FOR TRANSCODING MUSIC USING DELAYS OF ONE FRAME AND ONE SLICE

Mbit/s	frame [dB]	slice [dB]	diff. [dB]
1.0	38.89	38.74	0.65
1.5	40.25	40.07	0.49
2.0	41.13	40.95	0.60
2.5	41.72	41.57	0.60
3.0	42.18	42.07	0.53

tures can reduce the complexity of the transcoder. We have evaluated the performance of the optimized transcoder in the case of not using the MC transcoding error to transcode these pictures. This also allows the use of only one buffer in the feedback loop, and thus reduces the memory size of the whole system. Tables V and VI show the average PSNR of B pictures only, transcoded from the 4-Mbit/s bit stream into 1.0, 1.5, 2.0, 2.5, 3.0 Mbit/s for two cases: 1) using drift compensation, and 2) not compensating for drift. Note that in the case of no drift compensation, these pictures are transcoded using exactly the same procedure as for I pictures.

As shown in these tables, the difference between the two cases is very small for all conversion ratios. This is mainly due to the fact that errors of B pictures do not accumulate. Taking into account that the computational complexity of transcoding without drift correction on B pictures is far less than transcoding with drift correction, these results show that a much lower complex transcoder can be implemented, and still provides the required performance.

D. Low-Delay Operation

Since control of video traffic usually requires fast response to network demand, we have simulated the rate–distortion optimization of transcoding using only one slice delay. In order to compare the results with the case of one frame delay used in the experiments described above, we have used the same 4-Mbit/s bit streams for transcoding. The rate constraints were the same as the previous experiment. The number of bits originally allocated to the encoded-only bit streams set the target output rate of the transcoder.

Tables VII and VIII show the difference in picture quality between transcoding using one frame delay and one slice delay. The maximum difference between any two corresponding fames is also shown in these tables. In the case of one slice delay, there is a 0.3–0.6 dB drop in picture quality due to the reduced number of MB's used at each optimization step. However, this is the price to pay to keep the transcoder delay at a minimum for networking applications where fast response in bit-rate regulation will prevent picture degradations.

VI. Conclusion

We have proposed a low-complexity video transcoder, derived from a cascade of decoder–encoder, for converting MPEG-2 bit streams into lower rates. Its low complexity was shown to be achieved by: 1) using only one MC feedback loop with the MV's taken from the incoming bit stream unmodified, and 2) implementing the MC operation in the frequency domain using approximate matrices for fast computation of MC–DCT blocks. By using the input MV's, the most complex function of the MPEG-2 coding algorithm, the motion estimation, is not needed in the transcoder. On the other hand, we have shown that fast computation of MC–DCT reduces the computational complexity of a pixel-domain transcoder by 81% while maintaining its drift-free picture performance.

The overall performance of the proposed frequency-domain transcoder shows that transcoding high-quality bit streams into lower rates can produce better picture quality than both reencoding and encoding the original pictures at the same reduced bit rates with TM5. This is due to the optimal bit reallocation which takes into account the different MB types for minimizing the transcoding distortion. When operating at delays of about one slice, the drop in picture quality was found to be less than 0.65 dB compared with one frame delay. A variety of services and applications such as VoD, traffic control of preencoded video over ATM networks, and video multicasting over heterogeneous networks can benefit from the video transcoder proposed in this paper.

Appendix

A. Field DCT Coding

Example: Let us assume that $\boldsymbol{b_1}$ and $\boldsymbol{b_2}$ are the pixel blocks of different fields located in vertically aligned positions of the same MB. By interleaving one line of each block $\boldsymbol{b_i}, \boldsymbol{i = 1, 2}$, one can generate two frame blocks $\boldsymbol{b_i^*}, \boldsymbol{i = 1, 2}$. These can be obtained by multiplying the field blocks with the appropriate matrices $\boldsymbol{f_{ki}}, k, i = 1, 2$, as shown below.

$$\boldsymbol{b_1} = \begin{bmatrix} x_{00} & x_{01} & \cdots & x_{07} \\ x_{10} & x_{11} & \cdots & x_{17} \\ \cdots & \cdots & \cdots & \cdots \\ x_{70} & x_{71} & \cdots & x_{77} \end{bmatrix}$$

$$\boldsymbol{b_2} = \begin{bmatrix} y_{00} & y_{01} & \cdots & y_{07} \\ y_{10} & y_{11} & \cdots & y_{17} \\ \cdots & \cdots & \cdots & \cdots \\ y_{70} & y_{71} & \cdots & y_{77} \end{bmatrix}$$

$$\boldsymbol{f_{11}} = \begin{bmatrix} 1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \end{bmatrix}$$

$$\boldsymbol{f_{12}} = \begin{bmatrix} 0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \end{bmatrix}$$

$$\boldsymbol{f_{21}} = \begin{bmatrix} 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0 \end{bmatrix}$$

$$\boldsymbol{f_{22}} = \begin{bmatrix} 0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1 \end{bmatrix}$$

The frame blocks $\boldsymbol{b_1^*}, \boldsymbol{b_2^*}$, are obtained as follows:

$$\boldsymbol{b_1^*} = \boldsymbol{f_{11}} \cdot \boldsymbol{b_1} + \boldsymbol{f_{12}} \cdot \boldsymbol{b_2} = \begin{bmatrix} x_{00} & x_{01} & \cdots & x_{07} \\ y_{00} & x_{01} & \cdots & x_{07} \\ x_{10} & x_{11} & \cdots & x_{17} \\ \cdots & \cdots & \cdots & \cdots \\ y_{30} & y_{31} & \cdots & y_{37} \end{bmatrix}$$

$$\boldsymbol{b_2^*} = \boldsymbol{f_{21}} \cdot \boldsymbol{b_1} + \boldsymbol{f_{22}} \cdot \boldsymbol{b_2} = \begin{bmatrix} x_{40} & x_{41} & \cdots & x_{47} \\ y_{40} & x_{41} & \cdots & x_{47} \\ x_{50} & x_{51} & \cdots & x_{57} \\ \cdots & \cdots & \cdots & \cdots \\ y_{70} & y_{71} & \cdots & y_{77} \end{bmatrix}.$$

In order to obtain the DCT frame blocks $B_i^* \equiv \mathrm{DCT}(\boldsymbol{b_i^*})$ directly from the DCT field blocks $B_i \equiv \mathrm{DCT}(\boldsymbol{b_i})$, the matrices $F_{ki} \equiv \mathrm{DCT}(\boldsymbol{f_{ki}})$ are precomputed and used as constants to multiply by B_i. This is given by (29) in the DCT domain:

$$B_k^* = \sum_{i=1}^{2} F_{ki} \cdot B_i, \qquad k = 1, 2. \tag{29}$$

Note that this operation also uses the distributive property of DCT to matrix multiplication, i.e., $\mathrm{DCT}(\boldsymbol{p} \cdot \boldsymbol{q}) = \mathbf{DCT}(\boldsymbol{p}) \cdot \mathbf{DCT}(\boldsymbol{q})$.

B. Half-Pixel Accuracy

The DCT's of the following matrices are used for filtering blocks in the horizontal and vertical directions when MV's with half-pixel accuracy are used:

$$F_i^h \equiv \mathrm{DCT}\left(\boldsymbol{f_i^h}\right), \qquad \boldsymbol{i = 1, 2, 3} \tag{30}$$

$$F_i^v \equiv \mathrm{DCT}(\boldsymbol{f_i^v}), \qquad \boldsymbol{i = 1, 2, 3} \tag{31}$$

$$\boldsymbol{f_1^h} = \begin{bmatrix} 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 \end{bmatrix}$$

$$\boldsymbol{f_1^v} = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5 \end{bmatrix}$$

$$\boldsymbol{f_2^h} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{f_2^v} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{f_3^h} = \begin{bmatrix} 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 1 \end{bmatrix}$$

$$\boldsymbol{f_3^v} = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

C. Field Prediction

The M_{ki} matrices are the DCT's of the $\boldsymbol{m_{ki}}$ ones as follows:

$$M_{ki} \equiv \boldsymbol{DCT}(\boldsymbol{m_{ki}}), \qquad \boldsymbol{k, i = 1, 2} \tag{32}$$

$$\boldsymbol{m_{11}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{m_{12}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\boldsymbol{m_{21}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{m_{22}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

REFERENCES

[1] L. Chiariglione, "The development of an integrated audiovisual coding standard: MPEG," *Proc. IEEE*, vol. 83, pp. 151–157, Feb. 1995.

[2] G. Kerr, "BT's video on demand technology," in *5th IEE Conf. Telecommun.*, Brighton, U.K., Mar. 1995, pp. 318–322.

[3] H. J. Stuttgen, "Network evolution and multimedia communication," *IEEE Multimedia*, vol. 2, pp. 42–59, Fall 1995.

[4] C. Kaplan, "Interactive video services," in *5th IEE Conf. Telecommun.*, Brighton, U.K., Mar. 1995, pp. 323–326.

[5] J. Sandvoss and T. Schuttt, "Ecole: An ATM based environment for distributed learning," in *IEEE Int. Conf. Commun.*, Seattle, WA, June 1995, vol. 1, pp. 585–590.

[6] M. R. Civanlar and G. L. Cash, "An experimental system for MPEG-2 video transmission over ATM networks," in *6th Int. Workshop Packet Video*, Portland, OR, Sept. 1994, pp. C2.1–C2.4.

[7] M. Ghanbari, "Two-layer coding of video signals for VBR networks," *IEEE J. Select. Areas Commun.*, vol. 7, pp. 771–781, June 1989.

[8] ISO/IEC 13818-2, "Information technology—Generic coding of moving pictures and associated audio information—Part 2: Video," 1995.

[9] H. Sun, W. Kwok, and J. W. Zdepski, "Architectures for MPEG compressed bitstream scaling," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 191–199, Apr. 1996.

[10] A. Eleftheriadis and D. Anastassiou, "Meeting arbitrary QoS constraints using dynamic rate shaping of coded digital video," in *5th Int. Workshop Network and Operating Syst. for Digital Audio and Video*, Durham, NC, Apr. 1995, pp. 95–106.

[11] Y. Nakajima, H. Hori, and T. Kanoh, "Rate conversion of MPEG coded video by re-quantization process," in *IEEE Int. Conf. Image Processing*, Washington, DC, Oct. 1995, vol. 3, pp. 408–411.

[12] P. Assuncao and M. Ghanbari, "Post-processing of MPEG2 coded video for transmission at lower bit rates," in *IEEE Int. Conf. Acoust., Speech, Signal Processing, ICASSP'96*, Atlanta, GA, May 1996, vol. 4, pp. 1998–2001.

[13] D. G. Morrison, M. E. Nilsson, and M. Ghanbari, "Reduction of bit-rate of compressed video while in its coded form," in *6th Int. Workshop Packet Video*, Portland, OR, Sept. 1994, pp. 392–406.

[14] G. Keesman, R. Hellinghuizen, F. Hoeksema, and G. Heideman, "Transcoding of MPEG bitstreams," *Signal Processing: Image Commun.*, vol. 8, pp. 481–500, Sept. 1996.

[15] P. Assuncao and M. Ghanbari, "Transcoding of MPEG-2 video in the frequency domain," in *IEEE Int. Conf. Acoust., Speech, Signal Processing, ICASSP'97*, Munich, Germany, Apr. 1997, vol. 4, pp. 2633–2636.

[16] ISO/IEC, "Test Model 5, ISO/IEC JTC1/ SC29/ WG11/ N0400, MPEG93/457," Apr. 1993.

[17] N. Merhav and V. Bhaskaran, "A fast algorithm for DCT-domain inverse motion compensation," in *IEEE Int. Conf. Acoust., Speech, Signal Processing*, Atlanta, GA, May 1996, vol. 4, pp. 2307–2310.

[18] R. Plompen, B. F. Schuurink, and J. Biemond, "A new motion-compensated transform coding scheme," in *IEEE Int. Conf. Acoust., Speech, Signal Processing, ICASSP'85*, Tampa, FL, Mar. 1985, vol. 1, pp. 371–374.

[19] S.-F. Chang and D. G. Messerschmitt, "Manipulation and compositing of MC-DCT compressed video," *IEEE J. Select. Areas Commun.*, vol. 13, pp. 1–11, Jan. 1995.

[20] Y. Arai, T. Agui, and M. Nakajima, "A fast DCT-SQ scheme for images," *Trans. IEICE*, vol. E71, pp. 1095–1097, Nov. 1988.

[21] N. Merhav and V. Bhaskaran, "Fast algorithms for DCT-domain image down-sampling and for inverse motion compensation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, pp. 468–476, June 1997.

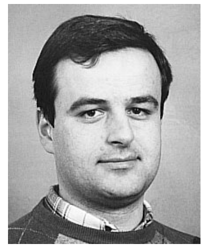
[22] W. B. Pennebaker and J. L. Mitchell, *JPEG: Still Image Data Compression Standard*. New York: Van Nostrand Reinhold, 1993.

[23] Y. Shoham and A. Gersho, "Efficient bit allocation for an arbitrary set of quantizers," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 36, pp. 1445–1453, Sept. 1988.

[24] K. Ramchandran and M. Vetterli, "Best wavelet packet bases in a rate-distrotion sense," *IEEE Trans. Image Processing*, vol. 2, pp. 160–175, Apr. 1993.

[25] W. Y. Lee and J. B. Ra, "A fast algorithm for optimal bit allocation," in *SPIE, Visual Commun. Image Processing, VCIP'97*, San Jose, CA, Feb. 1997, vol. 3024, pp. 167–175.

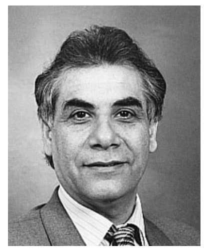
[26] W. Coene and G. Keesman, "A fast route for application of rate-distortion optimal quantization in an MPEG video encoder," in *IEEE Int. Conf. Image Processing, ICIP'96*, Lausanne, Switzerland, Sept. 1996, vol. 2, pp. 825–828.



Pedro A. A. Assunção was born in Coimbra, Portugal, in 1965. He graduated in electrical engineering in 1988 and received the M.Sc. degree in 1993, both from the University of Coimbra, Portugal. In 1998, he received the Ph.D. degree from the University of Essex, U.K.

He is currently a Lecturer at the Instituto Politécnico de Leiria and a Senior Research Engineer at Instituto de Telecomunicações in Portugal. His research interests include video coding and networking, compressed-domain algorithms, robust coding techniques for wireless communications, and multimedia networks.

Dr. Assunção is the recipient of the British Telecom Postgraduate Publication Prize for the best paper published in the Department of Electronic Systems Engineering, University of Essex, during the academic year of 1997–1998.



Mohammed Ghanbari (M'78–SM'97) received the B.Sc. degree in electrical engineering from Aryamehr University of Technology, Tehran, Iran, in 1970, the M.Sc. degree in telecommunications, and the Ph.D. degree in electronics engineering, both from the University of Essex, U.K., in 1976 and 1979, respectively.

After working almost ten years in industry, he started his academic career as a Lecturer in the Department of Electronic Systems Engineering, University of Essex, U.K., in 1988 and was promoted to Senior Lecturer, Reader, and then Professor in 1993, 1995, and 1996, respectively. His research interests are video compression and video networking. He is best known for his pioneering work on two-layer video coding for ATM networks.

Dr. Ghanbari is the corecipient of the 1995 A.H. Reeves Premium Prize for the year's best paper published in the *IEE Proceedings* on the theme of digital coding. He has been a member of the organizing committee of several international workshops and conferences. He was the Chairman of the Steering Committee of the 1997 International Workshop on Audio Visual Services over Packet Networks, AVSPN'97.