Fast Extraction of Spatially Reduced Image Sequences from MPEG-2 Compressed Video

Junehwa Song, *Member, IEEE*, and Boon-Lock Yeo

***Abstract*— MPEG-2 video standards are targeted for high-quality video broadcast and distribution and are optimized for efficient storage and transmission. However, it is difficult to process MPEG-2 for video browsing and database applications without first decompressing the video. Yeo and Liu [1] have proposed fast algorithms for the direct extraction of spatially reduced images from MPEG-1 video. Reduced images have been demonstrated to be effective for shot detection, shot browsing and editing, and temporal processing of video for video presentation and content annotation. In this paper, we develop new tools to handle the extra complexity in MPEG-2 video for extracting spatially reduced images. In particular, we propose new classes of discrete cosine transform (DCT) domain and DCT inverse motion compensation operations for handling the interlaced modes in the different frame types of MPEG-2, and we design new and efficient algorithms for generating spatially reduced images of an MPEG-2 video. The algorithms proposed in this paper are fundamental for efficient and effective processing of MPEG-2 video.**

***Index Terms*— Compressed-domain processing, deinterlacing, discrete cosine transform (DCT), inverse motion compensation (MC), interlacing, MPEG-2 video, video browsing.**

I. Introduction and Motivation

MPEG has been established as an international standard for the compression of digital video for efficient storage and transmission. The popularity and availability of video in MPEG compressed formats have increased in recent years with advances in computing and communication technologies. MPEG-1 video is designed for CD-ROM storage applications, while MPEG-2 video [2] has more complex encoding modes and is tailored for high quality broadcast video.

MPEG achieves efficient compression through several modes of encoding. However, these encodings make the efficient processing of MPEG compressed video more difficult without first decompressing the video stream. To overcome this challenge, a novel framework for the compressed domain processing of MPEG-1 video has been proposed in [1] and [3]. In the paper, video processing is based on spatially reduced images extracted directly from MPEG-1 compressed video without the need for full decompression. This approach of using reduced image sequences in video processing is advantageous in many applications, and it has been shown to be effective for: i) efficient shot detection; ii) rapid shot viewing and editing; and iii) effective temporal processing of video to extract high level structure for video presentation and content annotation. Using reduced images leads to significant reduction of computation in the various image processing tasks without loss of effectiveness.

Efficient extraction of spatially reduced images from MPEG-2 compressed video imposes different and additional challenges from MPEG-1 video. First, we need to handle interlaced video. In an MPEG-2 video, the top and bottom fields of a frame could be encoded as separate pictures (field picture) or encoded together as a frame (frame picture). In a frame picture, the top and bottom fields of a macroblock could be encoded together [frame discrete cosine transform (DCT) coding] or encoded separately (field DCT coding).

Second, we need to handle the enhanced modes of motion compensation allowed in MPEG-2. MPEG-2 provides the frame prediction as in MPEG-1. In addition, it provides the field-based motion prediction, in which two separate motion vectors are used for the prediction of top and bottom fields. It further provides dual-prime motion prediction and 16×8 motion prediction.

Third, we need to handle the high complexity resulting from the different combination of options allowed at each step of encoding. In an MPEG-2 video stream, the mixture of frame and field-based encoding is allowed at intra- and interframe levels. For the motion prediction, different modes of motion prediction could be also mixed within an MPEG-2 video stream. For example, in the 450-frame "Table-Tennis" MPEG-2 test sequence, about 70% of the intracoded macroblocks are frame coded and about 30% are field coded. Also, about 78% of the intercoded macroblocks are generated by frame-based motion predictions while about 22% are predicted by field-based motion predictions. While the various options allowed at different levels increase the flexibility at the encoding step, the efficient processing of the video stream on compressed domain becomes more difficult.

Finally, we need to provide fast algorithms to cope with the higher data rates in MPEG-2 than in MPEG-1 video (Main Profile/Main Level can support data streams up to 15 Mbit/s [2].)

In this paper, we build upon the results in [1] and [3], and we propose new algorithms to handle the additional complexity in MPEG-2 and generate spatially reduced image sequences from

Manuscript received December 1, 1997; revised March 18, 1999. This work was supported in part by an IBM Cooperative Fellowship and the NIST/ATP under Contract 70NANB5H1174. This work was presented in part at the SPIE Conference on Storage and Retrieval for Image and Video Databases VI, San Jose, CA, January 1998. This paper was recommended by Associate Editor K. Aizawa.

J. Song is with IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (e-mail: junesong@us.ibm.com).

B.-L. Yeo was with IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA. He is now with Intel Research Labs, Santa Clara, CA 95052 USA (e-mail: boon-lock.yeo@intel.com).

Publisher Item Identifier S 1051-8215(99)08178-1.

MPEG-2 compressed video. We identify a new class of DCT domain operations, namely DCT domain deinterlacing and DCT domain interlacing. DCT domain deinterlacing converts frame DCT coded data blocks to field DCT coded data blocks directly on the DCT domain, and DCT domain interlacing is the inverse operation. We extend DCT domain inverse motion compensation operation (the process of constructing an intracoded block in the DCT domain from an intercoded block using the DCT domain values of the reference blocks and motion vector informations [4]) to handle the enhanced modes of motion types and the different coding formats of the data blocks involved in the motion compensation. We then describe how the reduced image sequences are generated using these DCT domain operations. Note that although we use these operations for the purpose of reduced image sequence generation, they are also fundamental in manipulating MPEG-2 video streams in the compressed domain.

We describe our method of reduced image sequence generation in the context of DC image sequence (a DC image is formed from block-wise averages of 8 × 8 spatial blocks; for a frame-coded I frame, each DC coefficient is proportional to the average of the 8 × 8 block). For the MPEG-2 video with a typical spatial resolution of 708 × 480, a DC image has the resolution of 88 × 60. Our observations and experiments show that DC images from MPEG-2 video (typically at a resolution of 708 × 480) are sufficient for video browsing and common global image analysis tasks such as shot detection. The method proposed in this paper could easily be adapted also to generate 1/2 DC images when we only need a spatial resolution of 44 × 30 and also to DC+2AC images of spatial resolution of 176 × 120.

To cope with the higher data rates required in MPEG-2 video, we also present several new methods to speed up the process of reduced image sequence generation. We describe an approximation technique, called DC+2AC approximation, which significantly speeds up the extraction process with little degradation in quality. To further speed up the process, we provide two methods for the fast DCT domain inverse motion compensation: 1) utilizing the permutation matrixes and 2) utilizing shared information. One important advantage of these methods is that they can be used to speed up not only DC image sequence generation but also the generation of other types of reduced image sequences or the image sequences of full resolution. We show that each of these methods will result in about 40% savings for the DC+2AC approximation. In addition, the two methods can be combined to yield a speedup of up to 60%. Another advantage is that the methods are easy to implement in both hardware and software. Furthermore, the idea of utilizing shared information is independent of any processor or computation model and can be used on top of any optimized algorithms to achieve further speedup.

This paper is organized as follows. In Section II, we review the method proposed in [1] and [3] for extracting spatially reduced images from MPEG-1 video. We briefly review the additional functionalities of MPEG-2 which pose challenges in the extraction of reduced images in Section III. In Section IV, we set up the notations to be used for the rest of the paper. In Section V, we propose new formulations and tools for extracting reduced images from MPEG-2 video and study the extraction process for different frame types. In Section VI, we describe the approximation technique for the DC image sequence along with the fast algorithms for DCT domain inverse motion compensation. We present experimental results of our proposed algorithms on real MPEG-2 video and evaluate the quality of approximations in Section VII. Finally, Section VIII concludes the paper.

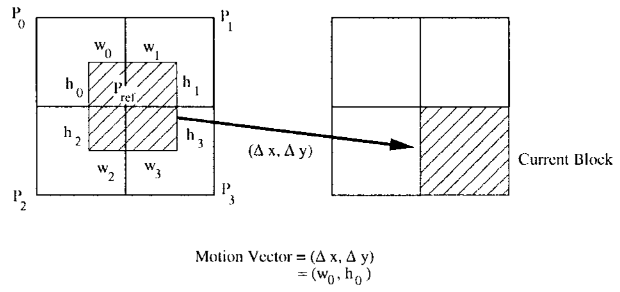


Fig. 1. Reference block (P_{ref}), motion vectors, and original blocks.

II. Review of Solutions for MPEG-1

In this section, we review the solution as presented in [1] and [3] for the reconstruction of spatially reduced images from MPEG-1 video. Two forms of reduced images are studied: DC and DC+2AC images. Due to space limitations, we only review the technique for DC image generation.

A DC image is obtained from block-wise averages of 8 × 8 blocks. For the I-frame of an MPEG-1 coded video, each pixel in the DC image corresponds to a scaled version of the DC coefficient of each DCT block. Each DC image is thus reduced 64 times compared to the original image. It is demonstrated in [1] that even at this resolution, global image features useful for specific class of image processing operations are well preserved.

The challenge is to also extract the DC images from P and B frames, which are coded using motion compensation to exploit temporal redundancy of video. A generic situation is shown in Fig. 1. Here, P_{ref} is the current block of interest, $P_0, \ldots, P_3$ are the four original neighboring blocks from which P_{ref} is derived, and the motion vector is $(\Delta x, \Delta y)$. The shaded regions in $P_0, \ldots, P_3$ are moved by $(\Delta x, \Delta y)$. We are thus interested in deriving the DC coefficients of P_{ref}.

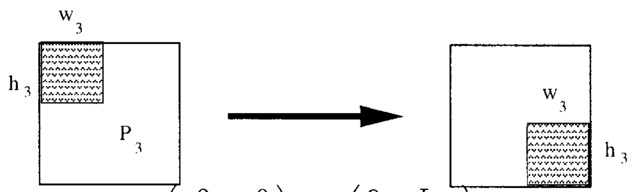
In [4], the relation of P_{ref} with respect to P_i is derived: if we represent each block as an 8 × 8 matrix, then we can describe in the spatial domain through the following matrix multiplications:

$$P_{\text{ref}} = \sum_{i=0}^{3} S_{i1} P_i S_{i2} \tag{1}$$

where S_{ij} are matrixes like

$$L_n = \begin{pmatrix} 0 & 0 \\ I_n & 0 \end{pmatrix} \quad \text{or} \quad R_n = \begin{pmatrix} 0 & I_n \\ 0 & 0 \end{pmatrix}.$$

Each I_n is an identity matrix of size n. An example of such an effect of S_{ij} on P is demonstrated in Fig. 2. The premultiplication shifts the subblock of interest vertically



$$\begin{pmatrix} 0 & 0 \\ I_{h_3} & 0 \end{pmatrix} P_3 \begin{pmatrix} 0 & I_{w_3} \\ 0 & 0 \end{pmatrix}$$

Fig. 2. Pre- and postmatrix multiplications to move subblocks.

TABLE I
MATRICES S_{i1} AND S_{i2}

Sub-block	Position	S_{i1}	S_{i2}
P_0	lower right	$\begin{pmatrix} 0 & I_{h_0} \\ 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 \\ I_{w_0} & 0 \end{pmatrix}$
P_1	lower left	$\begin{pmatrix} 0 & I_{h_1} \\ 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & I_{w_1} \\ 0 & 0 \end{pmatrix}$
P_2	upper right	$\begin{pmatrix} 0 & 0 \\ I_{h_2} & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 \\ I_{w_2} & 0 \end{pmatrix}$
P_3	upper left	$\begin{pmatrix} 0 & 0 \\ I_{h_3} & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & I_{w_3} \\ 0 & 0 \end{pmatrix}$

while postmultiplication shifts the subblock horizontally. Fast computation of 2D-DCT of (1) is studied in [5] based on factorization of the DCT matrix presented in [6]. We shall make generalization of the relationship of blocks as described in (1) later in the context of MPEG-2.

There are four possible locations of the subblock of interest: upper left, upper right, lower right, and lower left. The actions in terms of matrixes are tabulated in Table I.

While the value of S_{ij} is clear from Table I with the given values of h_i and w_i, we will sometimes write S_{ij} as a function of h_i and w_i. For example, $S_{01} = S_{01}(h_0, w_0) = S_{01}(h_0)$. This more precise way of writing the S_{ij} matrix will be needed when we discuss interlaced video.

Let us denote the 2D-DCT of an 8×8 block P as $DCT(P)$. We further denote an (i, j) component of the $DCT(P)$ as $(DCT(P))_{ij}$. Then, the linearity of the DCT operations also means that we can express the DC coefficient of $DCT(P_{\text{ref}})$ as

$$(DCT(P_{\text{ref}}))_{00} = \sum_{i=0}^{3} \left(\sum_{m=0}^{7} \sum_{l=0}^{7} w^i_{ml} (DCT(P_i))_{ml} \right) \quad (2)$$

for some weighting coefficients w^i_{ml}.

A key result of [1] is that

$$w^i_{00} = \frac{h_i w_i}{64} \quad (3)$$

i.e., the weight w^i_{00} is the ratio of overlaps of the block P_{ref} with block P_i.

An approximation, called the first-order approximation, approximates $(DCT(P_{\text{ref}}))_{00}$ by

$$DC(P_{\text{ref}})^1 = \sum_{i=0}^{3} \frac{h_i w_i}{64} DC(P_i). \quad (4)$$

This approximation requires only the motion vector informations and the DC values in the reference frames. It is shown in [1] that such approximation, when applied to B and P frames, yields good results in practice. Furthermore, the errors in the reconstruction can be analytically studied. Prediction at half-pixel accuracy is shown to have little effect on the quality of the DC approximation.

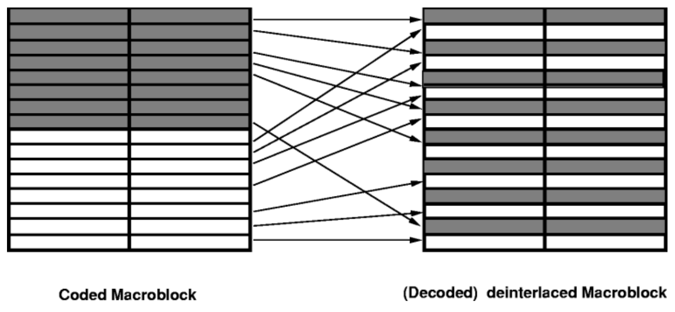


Fig. 3. Field DCT coding in a frame picture.

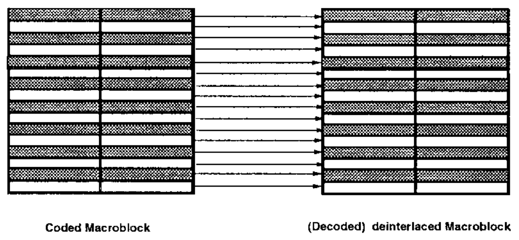


Fig. 4. Frame DCT coding in a frame picture.

III. REVIEW OF MPEG-2

MPEG-2 standard is intended for higher data rates and is targeted for compressing high-quality video, whereas MPEG-1 is intended for data rates on the order of 1.5 Mbit/s. For example, the advanced television systems committee (ATSC) digital television standard [7], [8] adopts MPEG-2 video compression standards for Advanced TV (ATV) video compression. MPEG-2 supports also the coding of interlaced video, which is not supported by MPEG-1. It also supports a much broader range of applications and modes of operations. MPEG-2 maintains all of the MPEG-1 video syntax and uses extensions for additional flexibility and functions, including interlaced video and scalable data streams.

In interlaced video, each frame is comprised of two fields, a top field and a bottom field. In MPEG-2, each field could be coded as a separate picture (field picture) or as a complete frame (frame picture). Frame pictures and field pictures could be mixed in a MPEG-2 data stream. The DCT coefficients for field pictures are always organized as independent fields. However, in a frame picture, DCT can be performed either on fields or frames on a macroblock basis. That is, a frame picture could have both frame-coded macroblocks and field-coded macroblocks. Figs. 3–5 show different formats of DCT-coded macroblocks possible in an MPEG-2 video.

Motion compensation (MC) in MPEG-2 is also more general than that in MPEG-1. There are two basic modes of MC's: field-based MC and frame-based MC. In field-based MC, two motion vectors are used to specify the translations of the top and bottom fields. In frame-based MC, only one motion vector

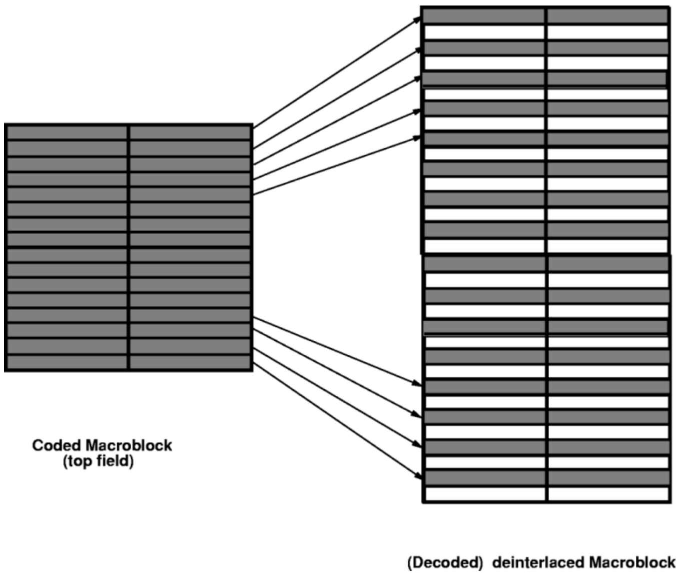


Fig. 5. A coded macroblock in a field picture.

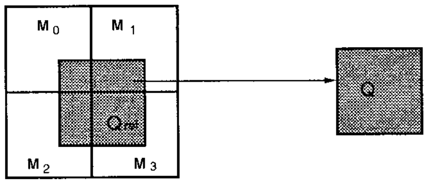


Fig. 6. Frame prediction in a frame picture or prediction in a field picture.

is used. Frame-based MC is similar to what is done in MPEG-1. The two cases are illustrated in Figs. 6 and 7. In field pictures, only field-based MC can be used, whereas in frame pictures, both frame-based and field-based MC are allowed.

Table II shows the percentage of macroblocks coded using different modes in a 450-frame Table-Tennis MPEG-2 sequence. It illustrates the mix of field- and frame-based modes, which is typical in MPEG-2 video sequences.

To improve prediction efficiency, MPEG-2 further supports the 16 × 8 motion-compensation mode. This mode allows a macroblock to be treated as an upper 16 × 8 region and a lower 16 × 8 region. Each region is then independently coded using MC. Another mode is the dual-prime motion prediction. It is a way of averaging the predictions from two adjacent fields of opposite parity when coding a given field or frame. It tends to reduce the noise in the data. For the rest of the paper, we will focus only on the field-based and frame-based MC because the treatments for 16 × 8 and dual-prime are similar.

MPEG-2 further allows four basic modes of bit stream scalability: data partitioning, signal-to-noise ratio (SNR) scalability, spatial scalability, and temporal scalability for a layered representation of the coded bit streams. The readers are referred to [9], [10], and [29] for more details on MPEG-2.

IV. Notations

For the rest of this paper, the following notational conventions are used. We represent an 8 × 8 (16 × 16) data block (macroblock) as an 8 × 8 matrix (16 × 16 matrix), or vice versa, when there is no confusion. We denote spatial domain blocks or macroblocks (or matrixes) with capital letters, (e.g., D, D', D_i, P, etc.) and the corresponding DCT domain blocks (or matrixes) by the same capital letters with hats (e.g., $\widehat{D_i}$, $\widehat{P_i}$, etc.). That is, for an 8 × 8 block represented by the matrix D, we have

$$\widehat{D} = DCT(D) = TDT^t \tag{5}$$

where T is the 8 × 8 DCT matrix with entries $t(i,j)$ (i denotes the ith row and j denotes the jth column) given by

$$t(i,j) = \frac{1}{2}\, k(i)\, \cos\frac{(2j+1)i\pi}{16} \tag{6}$$

where

$$k(i) = \begin{cases} \frac{1}{\sqrt{2}}, & i = 0; \\ 1, & \text{otherwise.} \end{cases} \tag{7}$$

We also explicitly use DCT to denotes the DCT domain value of a block when clearer notation is required. For example, $DCT(AB)$ denotes the 2D-DCT applied to the matrix product AB. We will also make constant use of the distributive property of 2D-DCT

$$DCT(AB) = DCT(A)DCT(B) = \widehat{A}\widehat{B}. \tag{8}$$

A noninterlaced 16 × 16 macroblock is denoted by a simple capital letter (e.g., D), whereas the corresponding interlaced macroblock is denoted by a primed capital letter (e.g., D'). Given a noninterlaced macroblock D, we denote the four component 8 × 8 blocks by D_0, D_1, D_2, and D_3, i.e., $D = \begin{pmatrix} D_0 & D_1 \\ D_2 & D_3 \end{pmatrix}$. Similarly, for the corresponding interlaced macroblock D', its four component blocks are denoted by D'_0, D'_1, D'_2, and D'_3, i.e., $D' = \begin{pmatrix} D'_0 & D'_1 \\ D'_2 & D'_3 \end{pmatrix}$. Each component in a block (matrix) is referenced by two indexes, i.e., A_{ij} represents the component at row i and column j in a block (matrix) A. To represent a row or column vector of a matrix, we use a $-$, i.e., A_{i-} represents the ith row vector and A_{-i} the ith column vector. We also use DC to denote the DC value and AC_{ij} to denote the $(i,j)th$, $(i,j) \neq (0,0)$, element of a given DCT block. That is, given a DCT block A, DC indicates A_{00}, and AC_{ij} indicates A_{ij}.

Consider a noninterlaced macroblock $D = \begin{pmatrix} D_0 & D_1 \\ D_2 & D_3 \end{pmatrix}$, where D_i, $i = 0,1,2,3$, is an 8 × 8 block. Similarly, let $D' = \begin{pmatrix} D'_0 & D'_1 \\ D'_2 & D'_3 \end{pmatrix}$ be the corresponding interlaced macroblock, i.e., D'_0 and D'_1 correspond to the top field of D, and D'_2 and D'_3 correspond to the bottom field of D. The relationship between D and D' can be described using a permutation matrix P as

$$D = PD' \tag{9}$$

where P is a 16 × 16 matrix, $P = [p_{ij}]$, $0 \leq i,j \leq 15$, such that

$$p_{ij} = \begin{cases} 1, & \text{if } (i = 2j \text{ and } 0 \leq j \leq 7) \quad \text{or} \\ & (i = 2j - 15 \text{ and } 8 \leq j \leq 15) \\ 0, & \text{otherwise.} \end{cases}$$

We will further denote for later reference $P = \begin{pmatrix} P_0 & P_1 \\ P_2 & P_3 \end{pmatrix}$, where each P_i is an 8 × 8 matrix.

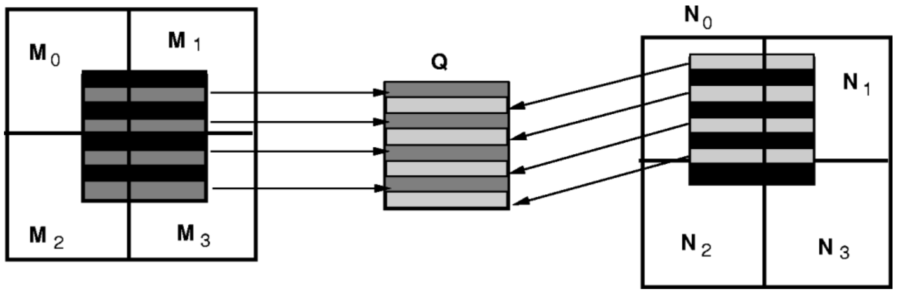


Fig. 7. Field prediction in a frame picture.

TABLE II
PERCENTAGE OF MACROBLOCKS UNDER DIFFERENT ENCODING MODES FOR A 450-FRAME TABLE-TENNIS MPEG-2 SEQUENCE

Frame type	I-coded macroblocks	Field DCT	Frame DCT	Field-based MC	Frame-based MC
I	100	27.6	72.4	-	-
B / P	0.18	54.5	45.5	22.2	77.8

V. CONSTRUCTION OF SPATIALLY REDUCED IMAGE SEQUENCES

Efficient extraction of spatially reduced images from MPEG-2 compressed video imposes different challenges from MPEG-1 video primarily due to the needs to handle interlaced video and the mixture of frame and field-based encoding at intra- and interframe levels. While deinterlacing could be easily performed in the spatial domain, we would like to efficiently generate the spatially reduced images without having to go through the expensive inverse DCT operations. In this section, we lay down the mathematical framework for handling interlaced video in the DCT domains of the different frame types and efficient extraction of reduced images.

A. Deinterlacing in DCT Domain

We start by describing an important operation which is required in many steps of the process, namely DCT domain deinterlacing. Consider a field-coded macroblock (Fig. 3). The upper two 8 × 8 blocks consist of top fields while the bottom two blocks consist of bottom fields for a spatial domain 16 × 16 macroblock. We call the compressed domain operation which transforms a field-coded macroblock represented in the DCT domain to a frame-coded macroblock in the DCT a domain DCT domain deinterlacing.

Following the notations in Section IV, we can describe the deinterlacing of a macroblock D' in the DCT domain in terms of the four component 8 × 8 DCT blocks

$$\begin{aligned}\widehat{D_0} &= \widehat{P_0}\widehat{D'_0} + \widehat{P_1}\widehat{D'_2}\\ \widehat{D_1} &= \widehat{P_0}\widehat{D'_1} + \widehat{P_1}\widehat{D'_3}\\ \widehat{D_2} &= \widehat{P_2}\widehat{D'_0} + \widehat{P_3}\widehat{D'_2}\\ \widehat{D_3} &= \widehat{P_2}\widehat{D'_1} + \widehat{P_3}\widehat{D'_3}.\end{aligned} \tag{10}$$

That is, we can calculate the DCT domain values of a noninterlaced macroblock D, i.e., $\widehat{D_i}$, $i = 0,1,2,3$, without going through inverse DCT and DCT transformations, given the DCT domain values of an interlaced macroblock D', i.e., $\widehat{D'_i}$, $i = 0,1,2,3$. From each equation above, we can see that the construction of a noninterlaced DCT domain block requires two 8 × 8 matrix multiplications.

B. Deinterlacing and Interlacing to Reduced Resolution for I Frames

Given an I-frame, we can construct its DC image by applying the DCT domain deinterlacing operation. For a frame-coded macroblock, we can trivially extract the DC coefficient. For a field-coded macroblock, we need to first apply the DCT domain deinterlacing operation (10) on the two 8 × 8 blocks D'_0, D'_1 in the top field and D'_2, D'_3 in the bottom field.

From (10), computing the DC value of D'_i, i.e., $(\widehat{D_i})_{00}$, requires 16 multiplications. We will show below that the complexity can be significantly reduced using properties of 2D-DCT and the 16 × 16 permutation matrix P. First, we state the following observation.

Observation 1:

a) $\left(\widehat{P_0}\right)_{0-} = \left(\widehat{P_1}\right)_{0-}$

b) $\left(\widehat{P_2}\right)_{0-} = \left(\widehat{P_3}\right)_{0-}$

c) $\left(\widehat{P_0}\right)_{0-} + \left(\widehat{P_2}\right)_{0-} = (1,0,0,0,0,0,0,0)$

d) $\left(\widehat{P_0}\right)_{02} = \left(\widehat{P_0}\right)_{04} = \left(\widehat{P_0}\right)_{06} = 0.$

The proof is straightforward and thus omitted.

Consider the construction of the DC value of the first block D_0 of a noninterlaced block D, i.e., $(\widehat{D_0})_{00}$

$$\begin{aligned}\left(\widehat{D_0}\right)_{00} &= \left(\widehat{P_0}\widehat{D'_0}\right)_{00} + \left(\widehat{P_1}\widehat{D'_2}\right)_{00}\\ &= \left(\widehat{P_0}\right)_{0-}\left(\widehat{D'_0}\right)_{-0} + (\widehat{P_1})_{0-}(\widehat{D'_2})_{-0}\\ &= \left(\widehat{P_0}\right)_{0-}\left(\left(\widehat{D'_0}\right)_{-0} + \left(\widehat{D'_2}\right)_{-0}\right)\end{aligned} \tag{11}$$

where (11) follows from Observation 1(a). This implies that we can construct the DC values for a noninterlaced block D'_0

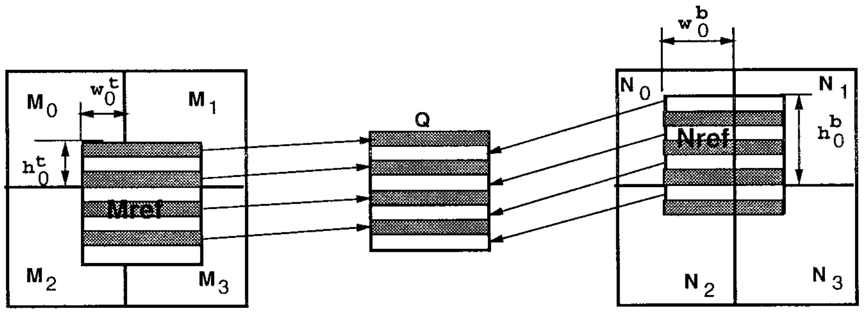


Fig. 8. Field prediction.

by eight multiplications. In addition, we have

$$\left(\widehat{P_0}\right)_{0-} = (0.5, 0.453, 0, -0.159, 0, 0.106, 0, -0.090) \quad (12)$$

so we can save three multiplications (since there are three zeros). In fact, the last two nonzero coefficients 0.106 and -0.090 can be practically ignored, implying that only three multiplications are needed. Given the DC value for D_0', we can get $(\widehat{D_2'})_{00}$ without any multiplications, as shown in the following:

$$\begin{aligned}\left(\widehat{D_2}\right)_{00} &= \left(\widehat{P_2}\right)_{0-}\left(\left(\widehat{D_0'}\right)_{-0} + \left(\widehat{D_2'}\right)_{-0}\right)\\ &= \left((1,0,0,0,0,0,0,0) - \left(\widehat{P_0}\right)_{0-}\right)\\ &\quad \cdot \left(\left(\widehat{D_0'}\right)_{-0} + \left(\widehat{D_2'}\right)_{-0}\right)\\ &\quad \text{(from Observation 1(c))}\\ &= \left(\left(\widehat{D_0'}\right)_{00} + \left(\widehat{D_2'}\right)_{00}\right)\\ &\quad - \left(\widehat{P_0}\right)_{0-}\left(\left(\widehat{D_0'}\right)_{-0} + \left(\widehat{D_2'}\right)_{-0}\right)\\ &= \left(\left(\widehat{D_0'}\right)_{00} + \left(\widehat{D_2'}\right)_{00}\right) - \left(\widehat{D_0}\right)_{00}.\end{aligned}$$

Likewise, the DC values for D_1 and D_3 can be computed by the following equations:

$$\begin{aligned}\left(\widehat{D_1}\right)_{00} &= \left(\widehat{P_0}\right)_{0-}\left(\left(\widehat{D_1'}\right)_{-0} + \left(\widehat{D_3'}\right)_{-0}\right)\\ \left(\widehat{D_3}\right)_{00} &= \left(\left(\widehat{D_1'}\right)_{00} + \left(\widehat{D_3'}\right)_{00}\right) - \left(\widehat{D_1}\right)_{00}.\end{aligned}$$

Therefore, the construction of a DC value for a noninterlaced block requires at most 2.5 multiplications ($5/2 = 2.5$). If we also ignore the last two nonzero coefficients of (12), we need only 1.5 multiplications ($3/2 = 1.5$).

The dual operation of DCT domain deinterlacing, i.e., DCT domain interlacing, can also be handled in a similar fashion. Moreover, the dual of Observation 1 holds, so that the interlacing of DC values can be done efficiently in the same way with 2.5 multiplications per block as in the deinterlacing operation [11].

C. DCT Domain Inverse Motion Compensation and Deinterlacing for P and B Frames

For macroblocks generated using motion compensations in P and B frames, we want to generate the DCT domain representations of the macroblock directly from the DCT information of the reference anchor blocks and motion vectors. The case of frame-based MC and frame-coded anchor blocks is identical to that of MPEG-1, as described by (1) (see Fig. 1). However, for field-based MC and/or field-coded anchor blocks, the situations are much more involved. Furthermore, the different anchor blocks used to predict an 8 × 8 block in a P and B frame could have different coding types. Likewise, the residues associated with a macroblock could be field or frame coded.

In the following sections, we describe how an 8 × 8 frame-coded block Q in the DCT domain of a P and B frame can be directly constructed from the DCT domain values of the reference anchor blocks and motion vectors of an MPEG-2 video. We will call this operation *DCT domain inverse motion compensation*. We will show that the more general setting required in MPEG-2 for inverse motion compensation is a generalization of (1) needed for MPEG-1.

1) Field-Based Prediction of Target Block Q: In a field-based MC, two motion vectors are used to predict the top and bottom field of a macroblock. We can set up an equation for the prediction of a field similar to the frame prediction's case. The prediction of a block Q in spatial domain using field-based MC is illustrated by Fig. 8.

In Fig. 8, the top field of the target block Q is predicted from four anchor 8 × 8 blocks M_0, M_1, M_2, M_3 and the bottom field is predicted from another set of four 8 × 8 anchor blocks N_0, N_1, N_2, N_3. Without loss of generality, we assume the motion vector for generating the top and bottom fields of Q are (w_0^t, h_0^t) and (w_0^b, h_0^b), respectively. Note that in the figure, the anchor blocks and the target block are noninterlaced spatial domain blocks. However, in DCT domain, these anchor blocks can belong to different anchor macroblocks, which can be coded in different formats, i.e., they can either be frame or field coded. We will show that the prediction of block Q in spatial domain can be described by

$$Q = \sum_{i=0}^{3} Q^{\mathcal{M}_i} + \sum_{i=0}^{3} Q^{\mathcal{N}_i} \quad (13)$$

$$Q^{\mathcal{M}_i} = \mathcal{P}_i^t \mathcal{S}_{i1}^t \mathcal{M}_i \mathcal{S}_{i2}^t \quad \text{and} \quad Q^{\mathcal{N}_i} = \mathcal{P}_i^b \mathcal{S}_{i1}^b \mathcal{N}_i \mathcal{S}_{i2}^b. \quad (14)$$

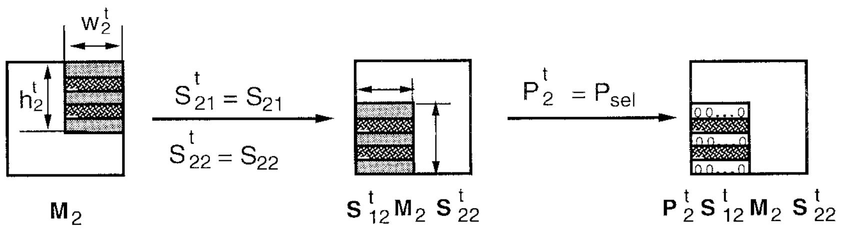


Fig. 9. Field-based motion compensation and anchor block is frame coded.

$\mathcal{M}_i$ and $\mathcal{N}_i$ represent the anchor blocks in either interlaced or noninterlaced format. For each anchor block $\mathcal{M}_i$ or $\mathcal{N}_i$, $Q^{\mathcal{M}_i}$ or $Q^{\mathcal{N}_i}$ denotes the contribution of $\mathcal{M}_i$ or $\mathcal{N}_i$ to the prediction of the target Q and is computed by $\mathcal{P}_i^t\mathcal{S}_{i1}^t\mathcal{M}_i\mathcal{S}_{i2}^t$ or $\mathcal{P}_i^b\mathcal{S}_{i1}^b\mathcal{N}_i\mathcal{S}_{i2}^b$, where $\mathcal{P}_i^t$, $\mathcal{P}_i^b$, $\mathcal{S}_{ij}^t$ and $\mathcal{S}_{ij}^b$, $0 \le i \le 3$ and $1 \le j \le 2$, are what we will describe below. The target block Q is formed by the summation of the contribution from each anchor block.

From (13) and (14), the DCT domain representation of Q is given by

$$\widehat{Q} = \sum_{i=0}^{3} \widehat{Q^{\mathcal{M}_i}} + \sum_{i=0}^{3} \widehat{Q^{\mathcal{N}_i}} \tag{15}$$

$$\begin{aligned} &= DCT\left(\sum_{i=0}^{3} \mathcal{P}_i^t\mathcal{S}_{i1}^t\mathcal{M}_i\mathcal{S}_{i2}^t\right) + DCT\left(\sum_{i=0}^{3} \mathcal{P}_i^b\mathcal{S}_{i1}^b\mathcal{N}_i\mathcal{S}_{i2}^b\right)\\ &= \sum_{i=0}^{3} \widehat{\mathcal{P}_i^t}\widehat{\mathcal{S}_{i1}^t}\widehat{\mathcal{M}_i}\widehat{\mathcal{S}_{i2}^t} + \sum_{i=0}^{3} \widehat{\mathcal{P}_i^b}\widehat{\mathcal{S}_{i1}^b}\widehat{\mathcal{N}_i}\widehat{\mathcal{S}_{i2}^b}\\ &= \sum_{i=0}^{3} DCT(\mathcal{P}_i^t\mathcal{S}_{i1}^t)\widehat{\mathcal{M}_i}\widehat{\mathcal{S}_{i2}^t} + \sum_{i=0}^{3} DCT(\mathcal{P}_i^b\mathcal{S}_{i1}^b)\widehat{\mathcal{N}_i}\widehat{\mathcal{S}_{i2}^b}. \end{aligned} \tag{16}$$

To compute the contributions $Q^{\mathcal{M}_i}$ or $Q^{\mathcal{N}_i}$, $0 \le i \le 3$, three steps are involved. First, from each anchor block $\mathcal{M}_i$ or $\mathcal{N}_i$, the subblock which contributes to the prediction of Q, called participating subblock hereinafter, is identified. Second, the identified participating subblock is shifted vertically and horizontally to the appropriate corner. This is done by multiplying the $\mathcal{S}_{ij}^t$ and $\mathcal{S}_{ij}^b$, $0 \le i \le 3$ and $1 \le j \le 2$. Third, each row of the shifted participating subblock is put to the correct row position corresponding to Q by multiplying the $\mathcal{P}_t^i$ and $\mathcal{P}_b^i$. The $\mathcal{S}_{ij}^t$, $\mathcal{S}_{ij}^b$, $\mathcal{P}_i^t$, and $\mathcal{P}_i^b$ are decided depending on the representation of $\mathcal{M}_i$ and $\mathcal{N}_i$, i.e., field coded or frame coded. Note that this setting is similar to that in MPEG-1, the difference being that field-based MC is involved and that the anchor blocks can be field or frame coded.

We now considers two separate cases for the anchor blocks: i) anchor noninterlaced and ii) anchor interlaced.

Anchor Noninterlaced: When $\mathcal{M}_i$ ($\mathcal{N}_i$) represents a noninterlaced block, $\mathcal{M}_i = M_i$ ($\mathcal{N}_i = N_i$). In this case, we first shift the participating subblock of M_i (N_i) and construct the contributing block after appropriate field selection [in this case, the top (bottom) field following Fig. 8]. Fig. 9 illustrates the procedure to construct $Q^{\mathcal{M}_2}$.

The subblock is simply identified from the horizontal and vertical displacement (w_i^t, h_i^t) or (w_i^b, h_i^b). The location of the subblock within the anchor block and its horizontal and vertical sizes decide the amount and direction of shifting in the same way as in MPEG-1. Therefore, the shifting for MPEG-1, denoted as S_{ij} in Table I, are used for $\mathcal{S}_{ij}^t$ and $\mathcal{S}_{ij}^b$, i.e., $\mathcal{S}_{ij}^t = S_{ij}(h_i^t, w_i^t)$ and $\mathcal{S}_{ij}^b = S_{ij}(h_i^b, w_i^b)$. In Fig. 9, the participating subblock is a $h_i^t \times w_i^t$ block in the upper right corner of the anchor block M_2 and is shifted to the lower left corner in the contributing block $Q^{\mathcal{M}_2}$.

The field selection in the contributing block is different for the top field prediction versus the bottom field prediction. In the top field prediction, every odd row is selected. However, in the bottom field selection, the selected odd rows are mapped to the even rows, i.e.,

$$\mathcal{P}_i^t = P_{\text{sel}}, \quad \mathcal{P}_i^b = P_{\text{swap}}P_{\text{sel}} \tag{17}$$

where

$$P_{\text{sel}} = \begin{pmatrix} 1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0 \end{pmatrix} \quad \text{and}$$

$$P_{\text{swap}} = \begin{pmatrix} 0&1&0&0&0&0&0&0\\ 1&0&0&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&1&0 \end{pmatrix}.$$

Anchor Interlaced: When $\mathcal{M}_i$ ($\mathcal{N}_i$) represents an interlaced block, $\mathcal{M}_i$ ($\mathcal{N}_i$) $= D_j'$, in some interlaced macroblock D', $D' = \begin{pmatrix} D_0' & D_1' \\ D_2' & D_3' \end{pmatrix}$. In this case, identification of the participating subblock from the anchor block is more involved. Given an interlaced 8×8 block, call the upper half of the block (i.e., the first four rows) the upper region and the lower half of the block (i.e., the last four rows) the lower region. Then, the participating subblock of each interlaced anchor block belongs to either the upper region or the lower region. Within each region, the participating subblock is displaced by $(w_i^t, \lceil h_i^t/2 \rceil)$ or $(w_i^b, \lceil h_i^b/2 \rceil)$ for $\mathcal{M}_i$ and for $\mathcal{N}_i$, when $i = 0, 1$ and displaced by $(w_i^t, \lfloor h_i^t/2 \rfloor)$ or $(w_i^b, \lfloor h_i^b/2 \rfloor)$ for $\mathcal{M}_i$ and for $\mathcal{N}_i$, when $i = 2, 3$. Fig. 10 illustrates a scenario of field-based MC and field-coded anchor block $\mathcal{M}_2$. In the figure, the motion vector is (h_2^t, w_2^t), and it is desired that the participating

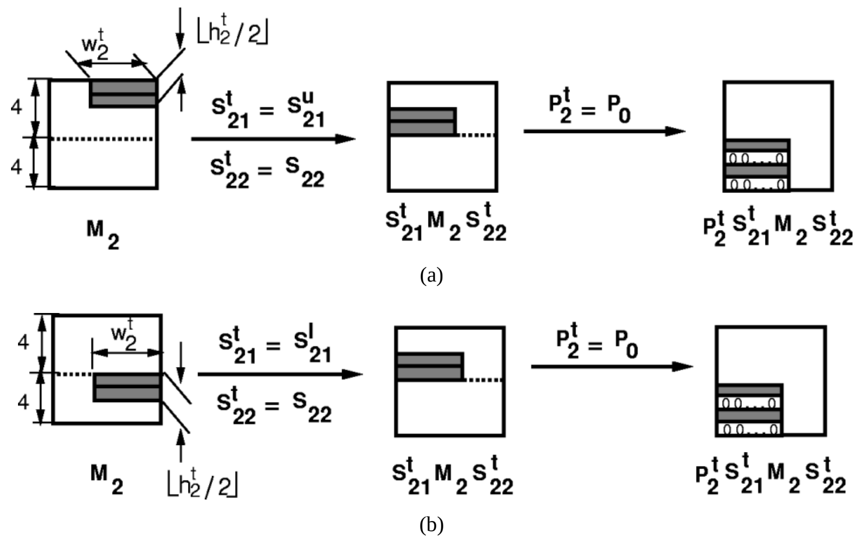


Fig. 10. Field-based motion compensation and anchor block is field coded. (a) Participating subblock comes from the upper region. (b) Participating subblock comes from the lower region.

TABLE III
SELECTION OF $\mathcal{S}_{i1}^t$, $\mathcal{S}_{i1}^b$, $\mathcal{S}_{i2}^t$, AND $\mathcal{S}_{i2}^b$ FOR VERTICAL AND HORIZONTAL SHIFTING WHEN AN ANCHOR BLOCK IS INTERLACED: $h' = h_i^t$ OR $h' = h_i^b$, $w' = w_i^t$ OR $w' = w_i^b$, AND 0_4 A 4×4 NULL MATRIX

Anchor block	$\mathcal{S}_{i1}^t$ $(\mathcal{S}_{i1}^b)$		$\mathcal{S}_{i2}^t$ $(\mathcal{S}_{i2}^b)$
	S_{i1}^u	S_{i1}^l	
$\mathcal{M}_0$ $(\mathcal{N}_0)$	$\begin{pmatrix} \begin{matrix} 0 & I_{\lceil h'/2 \rceil} \\ 0 & 0 \end{matrix} & 0_4 \\ 0_4 & 0_4 \end{pmatrix}$	$\begin{pmatrix} 0_4 & \begin{matrix} 0 & I_{\lceil h'/2 \rceil} \\ 0 & 0 \end{matrix} \\ 0_4 & 0_4 \end{pmatrix}$	$S_{02}(w')$
$\mathcal{M}_1$ $(\mathcal{N}_1)$			$S_{12}(w')$
$\mathcal{M}_2$ $(\mathcal{N}_2)$	$\begin{pmatrix} \begin{matrix} 0 & 0 \\ I_{\lfloor h'/2 \rfloor} & 0 \end{matrix} & 0_4 \\ 0_4 & 0_4 \end{pmatrix}$	$\begin{pmatrix} 0_4 & \begin{matrix} 0 & 0 \\ I_{\lfloor h'/2 \rfloor} & 0 \end{matrix} \\ 0_4 & 0_4 \end{pmatrix}$	$S_{22}(w')$
$\mathcal{M}_3$ $(\mathcal{N}_3)$			$S_{32}(w')$

subblock in $\mathcal{M}_2$ contributes to the top field of Q. In Fig. 10(a), the contributing subblock comes from the upper region, while in Fig. 10(b), the contributing subblock comes from the lower region.

The direction and amount of horizontal shifting is decided by w_i^t or w_i^b, in the same way as in the anchor noninterlaced case, i.e., $\mathcal{S}_{i2}^t = S_{i2}(w_i^t)$ and $\mathcal{S}_{i2}^b = S_{i2}(w_i^b)$. The vertical shifting, however, is different for each region as shown in Table III. When the participating subblock belongs to the upper region of an interlaced anchor block, S_{i1}^u is selected for both $\mathcal{S}_{i1}^t$ and $\mathcal{S}_{i1}^b$, whereas S_{i1}^l is selected when the subblock belongs to the lower region.

Thus, the shifted participating subblock in $\mathcal{S}_{i1}^t \mathcal{M}_i \mathcal{S}_{i2}^t$ or $\mathcal{S}_{i1}^b \mathcal{M}_i \mathcal{S}_{i2}^b$ is still interlaced. Therefore, each row of it needs to be mapped to the right row position by multiplying the two component P_0 and P_1 of the 16×16 permutation matrix P used in (10). Here, $\mathcal{P}_i^t = P_0$, $0 \le i \le 3$ for the top field prediction and $\mathcal{P}_i^b = P_1$, $0 \le i \le 3$ for the bottom field prediction.

Participating Subblocks in Special Positions: The discussion so far has been made by assuming that four anchor blocks are involved in the prediction of a field in a target block. However, there are cases when the participating subblock is positioned in distinguished positions in the anchor block. In such cases, the number of anchor blocks that need to be considered is reduced. By separately treating these cases, we can accordingly achieve the reduction in the required computation steps.

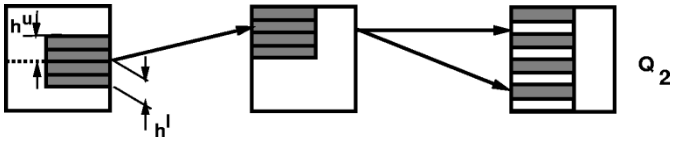


Fig. 11. Participating subblock across upper and lower regions of an interlaced anchor block.

In an anchor interlaced case, a participating subblock may be formed across four rows of the anchor block, as shown in Fig. 11. These four rows can be the four rows on the upper region, lower region, or taken across the upper and lower regions.

When the four rows are taken solely from upper or lower region, the matrix for vertical shifting $\mathcal{S}_{i1}^t$ or $\mathcal{S}_{i1}^b$ can be defined consistently with Table III, i.e., $\mathcal{S}_{i1}^u = \left(\begin{smallmatrix} I_4 & 0_4 \\ 0_4 & 0_4 \end{smallmatrix}\right)$ $\mathcal{S}_{i1}^l = \left(\begin{smallmatrix} 0_4 & I_4 \\ 0_4 & 0_4 \end{smallmatrix}\right)$. This case corresponds to the case when the subblock is aligned on both the top and bottom of the noninterlaced anchor blocks. However, when the four rows are taken across the upper and lower regions, there are two anchor blocks in the corresponding noninterlaced representation. Those two noninterlaced anchor blocks belong to a macro block and therefore the top or bottom field of the two forms an interlaced block. Suppose h^u rows are taken from upper region and h^l rows are taken from lower region. The matrix for vertical shifting $\mathcal{S}_{i1}^t$ or $\mathcal{S}_{i1}^b$ is

$$\begin{pmatrix} 0 & I_{h^u} & 0 & 0 \\ 0 & 0 & I_{h^l} & 0 \\ \multicolumn{2}{c}{0_4} & \multicolumn{2}{c}{0_4} \end{pmatrix}.$$

Another special case is that the participating subblock taken from eight columns of the corresponding anchor block. In this case, there is no need for horizontal shifting, i.e., $\mathcal{S}_{i2}^t = I$ or $\mathcal{S}_{i2}^b = I$. A similar situation can occur for the vertical shifting in the case of anchor noninterlaced.

2) Frame-Based Prediction of Target Block Q: In a frame prediction, a motion vector is used to predict both the top and bottom field of a target block. The prediction of a spatial domain block can be generally described by

$$Q = \sum_i Q^{\mathcal{M}_i}. \tag{18}$$

If an anchor block $\mathcal{M}_i$ is noninterlaced, it contributes both to the top and bottom fields of the target block. The contribution can be written as

$$Q^{\mathcal{M}_i} = S_{i1}\mathcal{M}_i S_{i2}$$

where S_{ij} for $i = 0,1,2,3$ and $j = 1,2$ are the matrixes for shifting participating subblocks, as described in Section II. When the anchor block $\mathcal{M}_i$ is interlaced, $\mathcal{M}_i$ contributes either to the top field or the bottom field of the target block, and there could be more than four anchor blocks. In this case, the contribution $Q^{\mathcal{M}_i}$ can be computed as in field based prediction (anchor interlaced case), i.e.,

$$Q^{\mathcal{M}_i} = \mathcal{P}_i^t \mathcal{S}_{i1}^t \mathcal{M}_i \mathcal{S}_{i2}^t \quad \text{or} \tag{19}$$

$$Q^{\mathcal{M}_i} = \mathcal{P}_i^b \mathcal{S}_{i1}^b \mathcal{M}_i \mathcal{S}_{i2}^b. \tag{20}$$

3) Unifying Equation for the Contribution of an Anchor Block: From the previous sections, we see that a noninterlaced target block $\widehat{Q}$ in the DCT domain can be constructed by adding the DCT domain values of contributing blocks $\widehat{Q^{\mathcal{M}_i}}$ from each anchor block $\mathcal{M}_i$ and there are up to eight anchor blocks. Despite the differences in the mode of motion compensations and coding types of the anchor blocks, we can write the following general equation describing the actions in the spatial domains:

$$Q = \sum_i Q^{\mathcal{M}_i} \tag{21}$$

$$Q^{\mathcal{M}_i} = \mathcal{P}_i \mathcal{S}_{i1} \mathcal{M}_i \mathcal{S}_{i2} \tag{22}$$

where

$$\begin{aligned}(\mathcal{P}_i, \mathcal{S}_{i1}, \mathcal{S}_{i1}), &= (\mathcal{P}_i^t, \mathcal{S}_{i1}^t, \mathcal{S}_{i2}^t) \text{ if top field prediction}\\ &= (\mathcal{P}_i^b, \mathcal{S}_{i1}^b, \mathcal{S}_{i2}^b) \text{ if bottom field prediction}\\ &= (I, S_{i1}, S_{i2}) \text{ if frame prediction and anchor noninterlaced.}\end{aligned}$$

D. Extracting Reduced Images from P and B Frames

To compute select DCT coefficients of Q, we note that $\widehat{\mathcal{S}_{ij}^t}$, $\widehat{\mathcal{S}_{ij}^b}$, $\widehat{\mathcal{P}_i^t}$, and $\widehat{\mathcal{P}_i^b}$, $0 \le i \le 3$ and $1 \le j \le 2$, can be precomputed. The contribution of each anchor block to each component in $\widehat{Q}$ can be written as follows:

$$\left(\widehat{Q^{\mathcal{M}_i}}\right)_{kl} = \sum_{m=0}^{7}\sum_{n=0}^{7} DCT(\mathcal{P}_i\mathcal{S}_{i1})_{km}\left(\widehat{\mathcal{M}_i}\right)_{mn}\left(\widehat{\mathcal{S}_{i2}}\right)_{nl} \tag{23}$$

$$= \sum_{m=0}^{7}\sum_{n=0}^{7} w_{kl}(i,m,n)\left(\widehat{\mathcal{M}_i}\right)_{mn} \tag{24}$$

where

$$w_{kl}(i,m,n) = DCT(\mathcal{P}_i\mathcal{S}_{i1})_{km}\left(\widehat{\mathcal{S}_{i2}}\right)_{nl}. \tag{25}$$

Then

$$(\widehat{Q})_{kl} = \sum_i \left(\widehat{Q^{\mathcal{M}_i}}\right)_{kl}. \tag{26}$$

From (24), we see that prediction of a component DCT value of a contributing block requires 64 multiplications resulting in a maximum of 64 × 8 multiplications for the prediction of a component value in a target block, since there is a maximum of eight possible contributing anchor blocks.

In addition, we will need to construct the DC and select AC values of the target block, which could potentially be used as future anchor blocks. Unlike in MPEG-1, blocks in B frame could be used as an anchor block (field-based motion compensation for prediction of the other field). This means that we need to maintain the DC and select AC values of most predicted blocks in P and B frames. In the Section VI, we will show how approximation techniques can lead to fast algorithms with little sacrifice in image quality.

VI. Fast Algorithms for Approximate Reconstruction of DC Images

As we have seen, the exact construction of a DC image sequence directly from MPEG-2 without full decoding can still be computationally expensive due to:

1) the requirement to deinterlace (or interlace) whole blocks before applying the inverse motion compensation operation;
2) the requirement to reconstruct the DC and select AC coefficients for each block which could serve as future anchor block;
3) the cost of DCT inverse motion compensation.

In this section, we propose novel solutions to reduce the computational cost. First, we present a scheme to approximate the DC images in Sections VI-A and VI-B. We generate the DC image sequence by maintaining only three coefficients for each 8 × 8 anchor block: DC, AC_{01}, and AC_{10}. We call an 8 × 8 DCT block in which all other coefficients except these three are zeros a DC+2AC block. In sections VI-C and VI-D, we identify a special property of a permutation matrix and a technique to facilitate the inverse motion compensation process using the property. Finally, in Section VI-E, we describe a method to identify and utilize the shared information in the prediction of the target blocks within a macroblock. Note that, as previously mentioned, the ideas presented in Sections VI-D and E can also be used for DCT domain inverse motion compensation at the full resolution and also for the generation of any type of reduced images.

A. Deinterlacing of DC+2AC Blocks

Given DC+2AC blocks, the DCT domain deinterlacing operation in (10) can be represented as follows:

$$\left(\widehat{D_i}\right)_{jl} \approx \sum_{k+l\le 1}\left(\left(\widehat{P_0}\right)_{jk}\left(\widehat{D_0'}\right)_{kl} + \left(\widehat{P_1}\right)_{jk}\left(\widehat{D_2'}\right)_{kl}\right) \tag{27}$$

$$\left(\widehat{D_{i+2}}\right)_{jl} \approx \sum_{k+l\le 1}\left(\left(\widehat{P_2}\right)_{jk}\left(\widehat{D_0'}\right)_{kl} + \left(\widehat{P_3}\right)_{jk}\left(\widehat{D_2'}\right)_{kl}\right) \tag{28}$$

where $0 \leq i \leq 1$ and $0 \leq j + l \leq 1$. Applying Observation 1, $(\widehat{D_0})_{jl}$ and $(\widehat{D_2})_{jl}$ can be computed as follows:

$$\begin{aligned}
\left(\widehat{D_0}\right)_{00} &\approx \left(\widehat{P_0}\right)_{00}\left(\left(\widehat{D'_0}\right)_{00} + \left(\widehat{D'_2}\right)_{00}\right)\\
&\quad + \left(\widehat{P_0}\right)_{01}\left(\left(\widehat{D'_0}\right)_{10} + \left(\widehat{D'_2}\right)_{10}\right)\\
\left(\widehat{D_0}\right)_{01} &\approx \left(\widehat{P_0}\right)_{00}\left(\left(\widehat{D'_0}\right)_{01} + \left(\widehat{D'_2}\right)_{01}\right)\\
\left(\widehat{D_0}\right)_{10} &\approx \left(\widehat{P_0}\right)_{10}\left(\widehat{D'_0}\right)_{00} + \left(\widehat{P_0}\right)_{11}\left(\widehat{D'_0}\right)_{10}\\
&\quad + \left(\widehat{P_1}\right)_{10}\left(\widehat{D'_2}\right)_{00} + \left(\widehat{P_1}\right)_{11}\left(\widehat{D'_2}\right)_{10}\\
&= \left(\widehat{P_0}\right)_{10}\left(\left(\widehat{D'_0}\right)_{00} - \left(\widehat{D'_2}\right)_{00}\right) + \left(\widehat{P_0}\right)_{11}\left(\widehat{D'_0}\right)_{10}\\
&\quad + \left(\widehat{P_1}\right)_{11}\left(\widehat{D'_2}\right)_{10}\\
\left(\widehat{D_2}\right)_{00} &\approx \left(\left(\widehat{D'_0}\right)_{00} + \left(\widehat{D'_2}\right)_{00}\right) - \left(\widehat{D_0}\right)_{00}\\
\left(\widehat{D_2}\right)_{01} &\approx \left(\left(\widehat{D'_0}\right)_{01} + \left(\widehat{D'_2}\right)_{01}\right) - \left(\widehat{D_0}\right)_{01}\\
\left(\widehat{D_2}\right)_{10} &\approx \left(\widehat{P_2}\right)_{10}\left(\widehat{D'_0}\right)_{00} + \left(\widehat{P_2}\right)_{11}\left(\widehat{D'_0}\right)_{10}\\
&\quad + \left(\widehat{P_3}\right)_{10}\left(\widehat{D'_2}\right)_{00} + \left(\widehat{P_3}\right)_{11}\left(\widehat{D'_2}\right)_{10}\\
&= \left(\widehat{P_2}\right)_{10}\left(\left(\widehat{D'_0}\right)_{00} - \left(\widehat{D'_2}\right)_{00}\right) + \left(\widehat{P_2}\right)_{11}\left(\widehat{D'_0}\right)_{10}\\
&\quad + \left(\widehat{P_3}\right)_{11}\left(\widehat{D'_2}\right)_{10}.
\end{aligned}$$

This approximation requires six multiplications and seven additions to deinterlace a DC+2AC block and three more multiplications and seven more addition operations for the second block, resulting in nine multiplications and 14 additions for two blocks.

B. Inverse Motion Compensation on DC+2AC Blocks

To proceed, we first define the following terms.

Definition 1: Given an 8 × 8 block B represented as an 8 × 8 matrix, and two 8 × 8 matrixes A and C, we define $\widetilde{DCT(ABC)}_{kl}$ as:

$$\widetilde{DCT(ABC)}_{kl} = \sum_{m+n\leq 1} v_{kl}(m,n)\widehat{B}_{mn}$$

where $v_{kl}(m,n) = \widehat{A}_{km}\widehat{C}_{nl}$, and $0 \leq k + l \leq 1$.

With the above definition, the DC+2AC block of a P and B frame can be approximated by the sum of contributing DC+2AC anchor blocks, i.e.,

$$\begin{aligned}
DCT(Q)_{kl} &\approx \sum_i \widetilde{DCT(Q^{\mathcal{M}_i})}_{kl} && (29)\\
&= \sum_i \sum_{m+n\leq 1} w_{kl}(i,m,n)(\widehat{\mathcal{M}_i})_{mn} && (30)
\end{aligned}$$

where $w_{kl}(i,m,n)$ is defined in (25). We call (30) the DC+2AC approximation to $DCT(Q)$. Therefore, the approximate prediction of a DC or an AC value requires a maximum of 12 multiplications and nine additions for frame prediction and a maximum of 24 multiplications and 18 additions for field prediction, resulting in a total of 36 or 72 multiplications for the prediction of a DC+2AC block.

C. The Permutation Matrix for Fast Inverse Motion Compensation of DC+2AC Block

We discuss in this section a property of the 8 × 8 permutation matrixes that will facilitate fast computation of DCT domain inverse motion compensation of a DC+2AC block. This property is exploited in Sections VI-D–E.

The computation of $w_{kl}(i,m,n)$ in (30) can be simplified by taking advantage of a property of the permutation matrix, which we state as follows.

Observation 2: For any 8 × 8 permutation matrix $\mathcal{P}$

a) $DCT(\mathcal{P})_{0-} = (1,0,0,0,0,0,0,0)$ and

b) $DCT(\mathcal{P})_{-0} = (1,0,0,0,0,0,0,0)^t$.

Proof: We then have the following equation:

$$DCT(\mathcal{P})_{i,j} = \sum_{l,m} t(i,l)(\mathcal{P})_{l,j}t(j,m) \tag{31}$$

where $t(j,m)$ is the entry in the DCT matrix and is defined in (6). For $i = 0$, we have

$$t(0,j) = \frac{1}{\sqrt{8}} \tag{32}$$

so that

$$\begin{aligned}
DCT(\mathcal{P})_{0,j} &= \frac{1}{\sqrt{8}} \sum_{l,m} (\mathcal{P})_{l,j}t(j,m)\\
&= \frac{1}{\sqrt{8}} \sum_{m} t(j,m) && (33)\\
&= \begin{cases} 1, & j = 0;\\ 0, & \text{otherwise}\end{cases} && (34)
\end{aligned}$$

where (33) follows from the fact that $\mathcal{P}$ is a permutation matrix, and (34) results from

$$\sum_{n=0}^{7} t(m,n) = \begin{cases} \sqrt{8} & m = 0;\\ 0, & \text{otherwise.}\end{cases} \tag{35}$$

The case for $j = 0$ is the same. □

The above observation states that the first row and column of $DCT(\mathcal{P})$ consist of all zero entries except the first row and first column which consists of a one.

Let B be an 8 × 8 data block represented as a matrix, A an 8 × 8 matrix, and $\mathcal{P}$ and $\mathcal{P}'$ are two 8 × 8 permutation matrixes. We can write

$$\begin{aligned}
\widetilde{DCT(\mathcal{P}BA)}_{kl} &= \sum_{m+n\leq 1} v^0_{kl}(m,n)(\widehat{B})_{mn},\\
v^0_{kl}(m,n) &= \widehat{\mathcal{P}}_{km}\widehat{A}_{nl} && (36)\\
\widetilde{DCT(AB\mathcal{P})}_{kl} &= \sum_{m+n\leq 1} v^1_{kl}(m,n)(\widehat{B})_{mn},\\
v^1_{kl}(m,n) &= \widehat{A}_{km}\widehat{\mathcal{P}}_{nl} && (37)\\
\widetilde{DCT(\mathcal{P}B\mathcal{P}')}_{kl} &= \sum_{m+n\leq 1} v^2_{kl}(m,n)(\widehat{B})_{mn},\\
v^2_{kl}(m,n) &= \widehat{\mathcal{P}}_{km}\widehat{\mathcal{P}'}_{nl} && (38)
\end{aligned}$$

where $k + l \leq 1$ and $m + n \leq 1$. Tables IV–VI show the values of nine coefficients used in the approximation,

TABLE IV
$\widehat{\mathcal{P}}_{km}\widehat{A}_{nl}$—PREMULTIPLICATION BY A PERMUTATION MATRIX: FIVE MULTIPLICATIONS

$(k,l)\setminus(m,n)$	(0,0)	(0,1)	(1,0)
(0,0)			0
(0,1)			0
(1,0)	0		0

TABLE V
$\widehat{A}_{km}\widehat{\mathcal{P}}_{nl}$—POSTMULTIPLICATION BY A PERMUTATION MATRIX: FIVE MULTIPLICATIONS

$(k,l)\setminus(m,n)$	(0,0)	(0,1)	(1,0)
(0,0)		0	
(0,1)	0		
(1,0)		0	0

TABLE VI
$\widehat{\mathcal{P}}_{km}\widehat{\mathcal{P}'}_{nl}$—PRE- AND POSTMULTIPLICAITON BY PERMUTATION MATRICES: TWO MULTIPLICATIONS

$(k,l)\setminus(m,n)$	(0,0)	(0,1)	(1,0)
(0,0)	1	0	0
(0,1)	0		0
(1,0)	0	0	

i.e., $v^0_{kl}(m,n)$ (Table IV), $v^1_{kl}(m,n)$ (Table V), and $v^2_{kl}(m,n)$ (Table VI), for each (k,l) and (m,n) pairs. Tables IV and V state that three values (DC, AC_{01}, and AC_{10}) of $\widehat{DCT(\mathcal{P}BA)}$ or $\widehat{DCT(AB\mathcal{P})}$ can be computed with five multiplications, saving four multiplications over the general case, as four entries are zero. Similarly, Table VI states that three values of $\widehat{DCT(\mathcal{P}B\mathcal{P}')}$ can be computed only with two multiplications, as one entry is a one and six entries are zeros.

Now consider the prediction of (a field of) a target block Q from four anchor blocks $\mathcal{M}_i$, $0 \le i \le 3$. When vertically or horizontally neighboring anchor blocks have the same coding format, the equations for DCT domain motion prediction can be rearranged so that they include permutation and thereby the computational cost for the same computation can be reduced. To see how this can be done, first note that the matrixes for horizontal shifting $\mathcal{S}_{i2}$ have the following properties:

$$\mathcal{S}_{02}(w_0) = \mathcal{S}_{22}(w_2), \quad \mathcal{S}_{12}(w_1) = \mathcal{S}_{32}(w_3) \tag{39}$$

$$\mathcal{S}_{02}(w_0) + \mathcal{S}_{12}(w_1) = P^0 = \begin{pmatrix} 0 & I_{8-w_0} \\ I_{w_0} & 0 \end{pmatrix} \tag{40}$$

and P^0 is an 8×8 permutation matrix.

D. Neighboring Anchor Blocks with the Same Coding Format

We show in this section how we can take advantage of the results in the previous section for faster computation by considering the contributions from more than one anchor block at a time. There are two cases: i) two neighboring anchor blocks with the same coding format and ii) all four neighboring anchor blocks with the same coding format. They are treated in Sections VI-D1 and 2, respectively.

1) Two Neighboring Anchor Blocks with the Same Coding Format: Suppose that two upper anchor blocks $\mathcal{M}_0$ and $\mathcal{M}_1$ belong to macroblocks coded in the same format. In this case, regardless of coding format or motion prediction mode

$$\mathcal{S}_{01} = \mathcal{S}_{11}, \quad \mathcal{P}_0 = \mathcal{P}_1. \tag{41}$$

Using these relationships, the prediction of $Q^{\mathcal{M}_0} + Q^{\mathcal{M}_1}$ can be rewritten as follows:

$$\begin{aligned} Q^{\mathcal{M}_0} + Q^{\mathcal{M}_1} &= \mathcal{P}_0\mathcal{S}_{01}\mathcal{M}_0\mathcal{S}_{02} + \mathcal{P}_1\mathcal{S}_{11}\mathcal{M}_1\mathcal{S}_{12} && (42) \\ &= \mathcal{P}_0\mathcal{S}_{01}(\mathcal{M}_0\mathcal{S}_{02} + \mathcal{M}_1\mathcal{S}_{12}) \\ &= \mathcal{P}_0\mathcal{S}_{01}(\mathcal{M}_0\mathcal{S}_{02} + \mathcal{M}_1(P^0 - \mathcal{S}_{02})) \\ &= \mathcal{P}_0\mathcal{S}_{01}(\mathcal{M}_0 - \mathcal{M}_1)\mathcal{S}_{02} + \mathcal{P}_0\mathcal{S}_{01}\mathcal{M}_1 P^0. && (43) \end{aligned}$$

While the DC+2AC approximation to the first term in (43) requires nine multiplications, the DC+2AC approximation to the second term can be computed with five multiplications, from Table V. Compared to (42), three extra additions are needed to compute $\widehat{\mathcal{M}_0} - \widehat{\mathcal{M}_1}$. However, as four terms are zero (from Table V), four additions can also be saved, resulting in saving four multiplication operations and one addition operation.

Likewise, when the two bottom anchor blocks $\mathcal{M}_2$ and $\mathcal{M}_3$ are coded in the same format, the equation for the prediction of $Q^{\mathcal{M}_2} + Q^{\mathcal{M}_3}$ can be rewritten in a similar fashion saving the same amount of computation. In this case, the fact that $\mathcal{S}_{21} = \mathcal{S}_{31}$ and $\mathcal{P}_2 = \mathcal{P}_3$ will be used.

When vertically neighboring anchor blocks are coded with the same formats, similar saving can be obtained. However, the reduction in the computation depends on the prediction mode and coding formats of the anchor blocks. For example, when the two anchor blocks $\mathcal{M}_0$ and $\mathcal{M}_2$ are coded in the same format, we have $\mathcal{P}_0 = \mathcal{P}_2$, and

$$\begin{aligned} Q^{\mathcal{M}_0} + Q^{\mathcal{M}_2} &= \mathcal{P}_0\mathcal{S}_{01}\mathcal{M}_0\mathcal{S}_{02} + \mathcal{P}_2\mathcal{S}_{21}\mathcal{M}_2\mathcal{S}_{22} \\ &= \mathcal{P}_0\mathcal{S}_{01}(\mathcal{M}_0 - \mathcal{M}_2)\mathcal{S}_{02} + \mathcal{P}_0 X\mathcal{M}_2\mathcal{S}_{02} && (44) \end{aligned}$$

where $X = \mathcal{S}_{01} + \mathcal{S}_{21}$. Note in the case of frame-based MC and noninterlaced anchor blocks, $P_0 = I$ and X is a permutation matrix, resulting in the same saving as in the horizontally neighboring case. In general, if we have interlaced anchor blocks, X will not be a permutation matrix.

2) All Four Anchor Blocks with the Same Coding Format: When all four anchor blocks are coded in the same format, we can write

$$\begin{aligned} Q &= \sum_{i=0}^{3} Q^{\mathcal{M}_i} \\ &= \mathcal{P}_0\mathcal{S}_{01}(\mathcal{M}_0 - \mathcal{M}_1 - \mathcal{M}_2 + \mathcal{M}_3)\mathcal{S}_{02} \\ &\quad + \mathcal{P}_0\mathcal{S}_{01}(\mathcal{M}_1 - \mathcal{M}_3)P^0 \\ &\quad + \mathcal{P}_0 X(\mathcal{M}_2 - \mathcal{M}_3)\mathcal{S}_{02} + \mathcal{P}_0 X\mathcal{M}_3 P^0. && (45) \end{aligned}$$

In (45), two terms include the permutation matrix P^0. Therefore, the DC+2AC approximation of Q can be computed

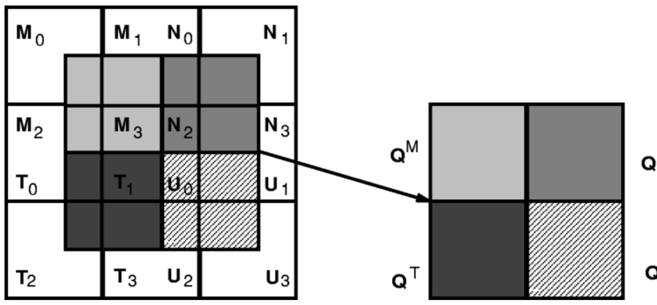


Fig. 12. Prediction of a 16 × 16 macroblock.

with 28 $(= 9+9+5+5)$ multiplications. It also requires nine extra addition operations to compute $R_1 = \widehat{\mathcal{M}}_1 - \widehat{\mathcal{M}}_3$, $R_2 = \widehat{\mathcal{M}}_2 - \widehat{\mathcal{M}}_3$, and $\widehat{\mathcal{M}}_0 - R_1 - R_2$; eight additions are saved by zero coefficients. It saves eight multiplication operations and requires one more addition operation. In the case of frame-based MC and noninterlaced anchors, P_0X is also a permutation matrix. In this case, the number of multiplications is reduced to 21 $(= 9+5+5+2)$, saving 15 multiplication and also five addition operations.

E. Macroblock Based Prediction: Reusing Intermediate Computation

In many cases, a set of neighboring target blocks are predicted using the same motion vector. For example, four 8 × 8 luminance data blocks (or chrominance data blocks in 4:4:4 chrominance format) within a macroblock share the same motion vector in both the field and frame prediction. In 4:2:2 chrominance format, two vertically neighboring blocks of chrominance share the same motion vector. In such cases, the same anchor block can be shared across the multiple target blocks. This means that there are possibly shared information in the predictions of multiple target blocks. In this section, we apply the techniques proposed in [11] for approximate reconstruction of inverse motion compensated DC+2AC block based on shared information across target blocks within the same macroblock.

Fig. 12 shows the prediction of a 16 × 16 macroblock Q. Each of the four 8 × 8 target blocks $Q^{\mathcal{M}}$, $Q^{\mathcal{N}}$, $Q^{\mathcal{T}}$, and $Q^{\mathcal{U}}$ is predicted from four anchor blocks, $\mathcal{M}_i$, $\mathcal{N}_i$, $\mathcal{T}_i$, and $\mathcal{U}_i$, $i = 0, 1, 2, 3$, respectively. For the prediction of each target block, the DCT domain values of the four contributing anchor blocks need to be computed and added. However, note that the predictions of four target blocks are strongly correlated to each other and do not have to be computed independently. Consider the following two observations.

First, although there are totally 16 contributing anchor blocks, there are only nine different anchor blocks, and five of them are shared among multiple target blocks, since the target blocks belong to the same macroblock and are predicted using the same motion vector. For example, the target blocks $Q^{\mathcal{M}}$ and $Q^{\mathcal{N}}$ share two anchor blocks, i.e., $\mathcal{M}_1 = \mathcal{N}_0$ and $\mathcal{M}_3 = \mathcal{N}_2$. Similarly, $Q^{\mathcal{M}}$ and $Q^{\mathcal{T}}$ share $\mathcal{M}_2 = \mathcal{T}_0$ and $\mathcal{M}_3 = \mathcal{T}_1$, etc. Second, the vertical and horizontal displacements of each participating subblock within the anchor block are identical across the four target blocks, $Q^{\mathcal{M}}$, $Q^{\mathcal{N}}$, $Q^{\mathcal{T}}$, and $Q^{\mathcal{U}}$. That is, $w^{\mathcal{M}_i} = w^{\mathcal{N}_i} = w^{\mathcal{T}_i} = w^{\mathcal{U}_i}$ and $h^{\mathcal{M}_i} = h^{\mathcal{N}_i} = h^{\mathcal{T}_i} = h^{\mathcal{U}_i}$ for $0 \leq i \leq 3$, where w's and h's superscribed by the anchor block (such as $w^{\mathcal{M}_i}$ $h^{\mathcal{M}_i}$) denote the horizontal and vertical displacements of the motion vectors associated with each anchor block.

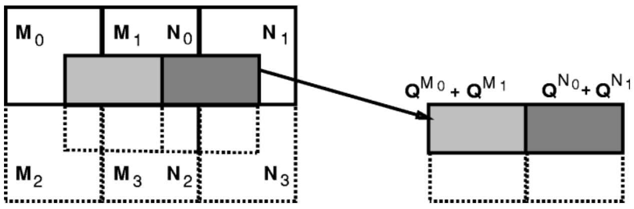


Fig. 13. Prediction of two horizontally neighboring regions.

1) Computation of Contribution of Two Neighboring Regions: Consider the prediction of upper regions of two horizontally neighboring target blocks $Q^{\mathcal{M}}$ and $Q^{\mathcal{N}}$ as shown in Fig. 13. The upper regions of the target blocks can be computed as the addition of two contributing blocks, i.e., $Q^{\mathcal{M}_0} + Q^{\mathcal{M}_1}$ and $Q^{\mathcal{N}_0} + Q^{\mathcal{N}_1}$, respectively, from the anchor blocks $\mathcal{M}_0$, $\mathcal{M}_1$ $(= \mathcal{N}_0)$, and $\mathcal{N}_1$. Suppose that $\mathcal{M}_0$, $\mathcal{M}_1$ $(= \mathcal{N}_0)$, and $\mathcal{N}_1$ belong to the macroblocks coded in the same format. We further assume that $\widehat{Q^{\mathcal{M}_0}} + \widehat{Q^{\mathcal{M}_1}}$ has already been computed using (43) and $\widehat{Q^{\mathcal{N}_0}} + \widehat{Q^{\mathcal{N}_1}}$ is currently sought for.

By changing the order of the anchor blocks, we can rewrite (43) as

$$Q^{\mathcal{M}_0} + Q^{\mathcal{M}_1} = \mathcal{P}_0\mathcal{S}_{01}(\mathcal{M}_1 - \mathcal{M}_0)\mathcal{S}_{12} + \mathcal{P}_0\mathcal{S}_{01}\mathcal{M}_0P^0. \quad (46)$$

The corresponding equation for the target block $Q^{\mathcal{N}}$ is:

$$Q^{\mathcal{N}_0} + Q^{\mathcal{N}_1} = \mathcal{P}_0\mathcal{S}_{01}(\mathcal{N}_1 - \mathcal{N}_0)\mathcal{S}_{12} + \mathcal{P}_0\mathcal{S}_{01}\mathcal{N}_0P^0. \quad (47)$$

Note that the DCT domain value for the second term of (47), i.e., $\widehat{\mathcal{P}_0}\widehat{\mathcal{S}_{01}}\widehat{\mathcal{N}_0}\widehat{P^0}$ has already been computed as $\widehat{\mathcal{P}_0}\widehat{\mathcal{S}_{01}}\widehat{\mathcal{M}_1}\widehat{P^0}$ using (43). Only $\widehat{\mathcal{P}_0}\widehat{\mathcal{S}_{01}}(\widehat{\mathcal{N}_1} - \widehat{\mathcal{N}_0})\widehat{\mathcal{S}_{12}}$ needs to be additionally computed to get $\widehat{Q^{\mathcal{N}_0}} + \widehat{Q^{\mathcal{N}_1}}$. Therefore, nine additional multiplications are needed. Since (46) needs 14 multiplication (from Section VI-D1), a total of 23 $(= 14+9)$ multiplications are needed to construct the DC+2AC approximations of $\widehat{Q^{\mathcal{M}_0}} + \widehat{Q^{\mathcal{M}_1}}$ and $\widehat{Q^{\mathcal{N}_0}} + \widehat{Q^{\mathcal{N}_1}}$.

A similar idea can be applied to any pair of regions which are vertically or horizontally neighboring as long as the anchor blocks are coded in the same format. For example, consider the prediction of $\widehat{Q^{\mathcal{M}_0}} + \widehat{Q^{\mathcal{M}_2}}$ and $\widehat{Q^{\mathcal{T}_0}} + \widehat{Q^{\mathcal{T}_2}}$. We can use (44) for $Q^{\mathcal{M}_0} + Q^{\mathcal{M}_2}$. As for $Q^{\mathcal{T}_0} + Q^{\mathcal{T}_2}$

$$Q^{\mathcal{T}_0} + Q^{\mathcal{T}_2} = \mathcal{P}_0\mathcal{S}_{21}(\mathcal{T}_2 - \mathcal{T}_0)\mathcal{S}_{02} + \mathcal{P}_0X\mathcal{T}_0\mathcal{S}_{02}. \quad (48)$$

The second terms of the two equations are the same since $\mathcal{M}_2 = \mathcal{T}_0$. Note that this is true, whether or not X is a permutation matrix. In this case the total number of multiplications to predict two regions is 27 $(= 9+9+9)$. In the frame prediction and anchor progressive case, it becomes 23 $(= 9+5+9)$.

2) Prediction of Neighboring Blocks: A similar idea can be applied in the prediction of pairs of (vertically or vertically) neighboring blocks or the prediction of a whole macroblock. In the following, we show the prediction of the whole macroblock composed of $Q^{\mathcal{M}}$, $Q^{\mathcal{N}}$, $Q^{\mathcal{T}}$, and $Q^{\mathcal{U}}$.

From [11], we can write $Q^{\mathcal{N}}$, $Q^{\mathcal{T}}$, and $Q^{\mathcal{U}}$ as

$$\begin{aligned} Q^{\mathcal{N}} = & \overbrace{\mathcal{P}_0\mathcal{S}_{01}(\mathcal{N}_1-\mathcal{N}_0-\mathcal{N}_3+\mathcal{N}_2)\mathcal{S}_{12}}^{\text{new}} \\ & + \mathcal{P}_0\mathcal{S}_{01}(\mathcal{N}_0-\mathcal{N}_2)P^0 \\ & + \overbrace{\mathcal{P}_0X(\mathcal{N}_3-\mathcal{N}_2)\mathcal{S}_{12}}^{\text{new}} + \mathcal{P}_0X\mathcal{N}_2P^0 \end{aligned} \tag{49}$$

$$\begin{aligned} Q^{\mathcal{T}} = & \overbrace{\mathcal{P}_0\mathcal{S}_{21}(\mathcal{T}_2-\mathcal{T}_3-\mathcal{T}_0+\mathcal{T}_1)\mathcal{S}_{02}}^{\text{new}} \\ & + \overbrace{\mathcal{P}_0\mathcal{S}_{21}(\mathcal{T}_3-\mathcal{T}_1)P^0}^{\text{new}} \\ & + \mathcal{P}_0X(\mathcal{M}_2-\mathcal{M}_3)\mathcal{S}_{02} + \mathcal{P}_0X\mathcal{M}_3P^0 \end{aligned} \tag{50}$$

$$\begin{aligned} Q^{\mathcal{U}} = & \overbrace{\mathcal{P}_0\mathcal{S}_{21}(\mathcal{U}_3-\mathcal{U}_2-\mathcal{U}_1+\mathcal{U}_0)\mathcal{S}_{12}}^{\text{new}} \\ & + \mathcal{P}_0\mathcal{S}_{21}(\mathcal{T}_3-\mathcal{T}_1)P^0 \\ & + \mathcal{P}_0X(\mathcal{N}_3-\mathcal{N}_2)\mathcal{S}_{12} + \mathcal{P}_0X\mathcal{M}_3P^0. \end{aligned} \tag{51}$$

In (49), the second and the fourth terms are the same as those of (45) and can be reused, since $\mathcal{N}_0 = \mathcal{M}_1$ and $\mathcal{N}_2 = \mathcal{M}_3$. Similarly, in (50), the third and fourth terms are the same of those of (45). In (51), the second and fourth terms are the same as those of (50), and the third term is the same that of (49). Therefore, for the prediction of the whole macroblock, only five additional terms marked "new" need to be computed after computing the four terms in (45).

In DC+2AC approximation, $\widehat{Q^{\mathcal{N}}}$ can be computed with 18 $(9+9)$ additional multiplications (14 additional multiplications for frame prediction and anchor noninterlaced case). Since one of the two new terms of (50) includes a permutation matrix, $\widehat{Q^{\mathcal{T}}}$ can be computed with only 14 $(= 9+5)$ multiplications. For $\widehat{Q^{\mathcal{U}}}$, we need only nine additional multiplications for the first term. Therefore, for the DC+2AC approximation of the whole macroblock, 69 multiplications are used (58 multiplications in frame prediction and anchor noninterlaced), resulting in 17.25 (14.5) multiplications per block. This means that about 50% (or 60%) of improvement over the number of the multiplications over the original method, which requires 36 multiplications.

The overall speed of our algorithms will vary on the input data sequences because of the possible diversity in the mixture of encoding options. For example, DC images can be generated a lot faster when there are more I macroblocks. The process will be also faster when there is more uniformity in the coding formats of the anchor blocks.

Our proposed techniques can be generalized to other types of reduced images. For instance, to generate 1/2 DC images, which are spatially reduced 16 times in each dimension, we can use the same DCT domain inverse motion compensation model in Section V, similar approaches to approximate reconstruction as in Sections VI-B and VI-C, and fast algorithms based on shared information as in Section VI-D. The only difference is that we can afford to use less information than we did for DC reconstruction. Thus, we may not need to extract AC_{01} or AC_{10} as we have done for DC images. Similarly, to extract DC+2AC images (reduced eight times in each dimensions) across all image frames, we can take on similar approaches, except that we will probably need more coefficients beyond

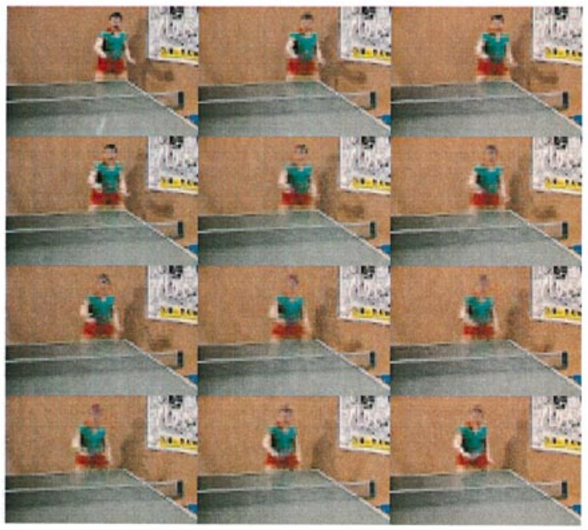


Fig. 14. DC image sequence approximated by DC+2AC.

AC_{01} and AC_{10}. The framework of DCT domain inverse motion compensation, fast algorithms based on matrix properties and shared information is general enough to support extraction of DCT blocks even up to full resolutions. The task of fast extraction of reduced images boils down to careful dropping of higher order AC coefficients to the extent that the image quality is preserved. Readers should be able to extrapolate other forms of reduced images from our detailed treatment of fast DC image extraction.

VII. Experiments and Results

In this section, we show sample DC images and demonstrate the accuracy of approximations. The approximation is evaluated on the same 450-frame Table-Tennis sequence at a spatial resolution 704 × 480 used in Table II. The sequence exhibits object motions and is compressed as frame pictures with frame pattern IBBPBBPBBPBB. The sequence uses the color format 4:2:0. We show a GOP consisting of 12 DC images in Fig. 14.

A. Accuracy of DC Image Sequence

We evaluate the accuracy of the reduced image sequence as follows. We first measure the difference between exact DC values and the approximated DC values. To see the effect of each approximation step, we separately collect the static for I, P, and B frames. We also separately measure the accuracy for luminance and chrominance blocks. Lastly, to see the effect of the approximate DCT domain deinterlacing (Section VI-A), we repeat the measurement twice. In the first measurement, exact DC+2AC blocks are constructed for I frames, whereas in the second measurement, the approximate deinterlacing is used for I frames.

Figs. 15 and 16 show the cumulative percentage of the number of blocks as a function of error ranges, i.e., a point (e, c) on the plots means that there are c % of blocks with absolute errors $\leq e$. In Fig. 15, the results are obtained by using the exact DCT deinterlacing for I frames. Fig. 16 shows the result obtained when the approximate DCT deinterlacing is

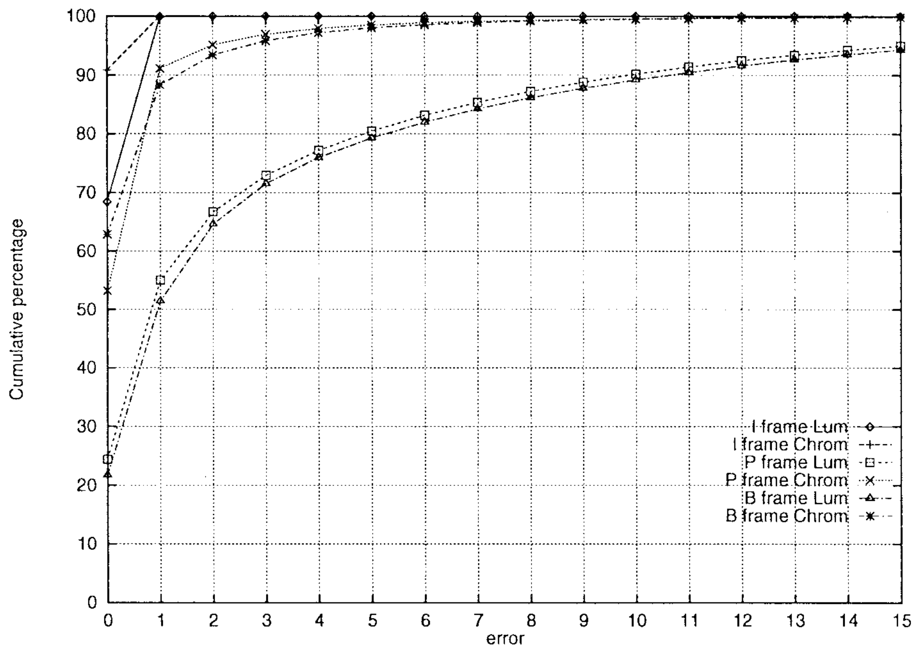


Fig. 15. Cumulative errors: DC + 2AC blocks for I frames are accurately constructed.

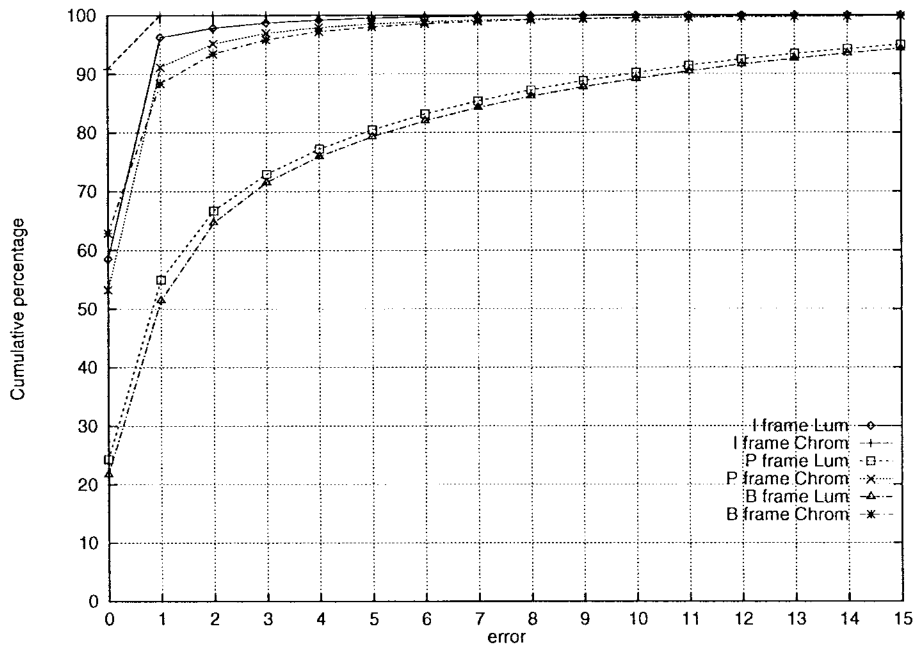


Fig. 16. Cumulative errors: DC + 2AC blocks for I frames are approximated by using DC, AC_{10}, and AC_{01}.

used for I frames. In the figures, we show the accuracy in the error range [0, 15]. In both cases, about 95% of blocks belong to this range in all I, P, and B frames and in both luminance and chrominance blocks.

In both figures, the approximation of chrominance blocks shows high accuracy. DC values in chrominance blocks in I frames are directly taken without approximation. Therefore, the errors for chrominance blocks in I frames were bounded within [0,1]. DC values for the chrominance blocks in P frames and B frames are computed without the DCT domain deinterlacing operation. In both figures, 95% of blocks were bounded in the error range [0, 3].

The accuracy of the luminance values in I frame shows the effect of DCT domain deinterlacing. Since we do not approximate but exactly compute using five values (DC, AC_{10}, AC_{30}, AC_{50}, and AC_{70}[1]) to extract DC values in I frames in Fig. 15, the error is bounded in [0,1]. In Fig. 16, we used only DC and AC_{10} values, but still more than 95% of blocks are approximated within error range [0, 1].

Estimation of luminance values in P and B frames shows the effect of the approximated DCT domain deinterlacing as well as DC+2AC approximation to DCT inverse motion

[1]From Observation 1, DCT domain deinterlacing does not require AC_{20}, AC_{40}, AC_{60} since they are multiplied by zero.

compensation. In the experiment, we also maintain field-coded DC, AC_{10}, and AC_{01} values. As shown in the figures, the accuracy in these values are relatively lower than other values, due to the approximations involved. However, the result still shows that 80% of blocks were approximated within error range [0, 5]. Comparing the two figures, approximated construction of DC values in I frames (Fig. 16) does not reduce the quality of DC images in P and B frames.

The accuracy of the DC images from P and B frames also depends on the distribution of intracoded macroblocks versus intercoded macroblocks within them. In the Table-Tennis sequence used in our measurement, only 0.18% of all the macroblocks in P and B frames are intracoded. Based on the fact that the DC images from I frames are more accurate than those from P or B frames as shown in the figures, DC images from P and B frames will be more accurate with more intracoded macroblocks.

VIII. Conclusions

In this paper, we study the extraction of spatially reduced image sequences from MPEG-2 compressed video sequences. The process is much more complex than in MPEG-1 due to extended functionality and syntax allowed in MPEG-2. More importantly, the process gets more complicated since different options can be mixed in a video sequence. The proposed method is based on a new class of DCT domain operations. DCT domain deinterlacing (interlacing) is used to convert field-DCT (frame-DCT) coded macroblocks to frame-DCT (field-DCT) coded ones. We also derive new DCT domain inverse motion compensation operation to handle the mixture of different types of motion compensation, such as field based MC and frame based MC, as well as the different types of coding formats for anchor frames. We further develop fast algorithms for those operations to facilitate the process of the reduced image generation and demonstrate the effectiveness of the algorithms.

Acknowledgment

The authors thank Dr. J. Kouloheris and Dr. R. Rajagopalan for helpful discussions related to the MPEG-2 standards. The authors also thank the anonymous reviewers for their valuable comments which improved this paper.

References

[1] B. L. Yeo and B. Liu, "On the extraction of DC sequences from MPEG compressed video," in *Proc. Int. Conf. Image Processing*, vol. II, 1995, pp. 260–263.
[2] ISO, ISO/IEC JTC1 CD 13818 *Generic Coding of Moving Pictures and Associated Audio*, 1994.
[3] B. L. Yeo, "Efficient processing of compressed images and video," Ph.D. dissertation, Elect. Eng. Dept., Princeton University, Princeton, NJ, Jan. 1996. [Online]. Available WWW: http://www.ee.princeton.edu/˜yeo/Thesis.
[4] 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.
[5] N. Merhav and V. Bhaskaran, "A fast algorithm for DCT domain inverse motion compensation," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, vol. 4, May 1996, pp. 2307–2310.
[6] Y. Arai, T. Agui, and M. Nakajima, "A fast DCT-SQ scheme for images," *Trans. IEICE*, vol. E71, pp. 1095–1097, Nov. 1988.
[7] *ATSC Digital Television Standard*, ATSC Standard A/53, 1995.
[8] *Guide to the Use of the ATSC Digital Television Standard*, ATSC Standard A/54, 1995.
[9] J. L. Mitchell, W. B. Pennebaker, C. E. Foog, and D. J. Le Gall, *MPEG Video Compression Standard*. London, U.K.: Chapman & Hall, 1996.
[10] B. G. Haskell, A. Puri, and A. N. Netravali, *Digital Video: An Introduction to MPEG-2*, London, U.K.: Chapman & Hall, 1997.
[11] J. Song and B. L. Yeo, "Fast DCT domain inverse motion compensation based on shared information in a macoblock," *IEEE Trans. Circuits Syst. Video Technol.*, to be published.
[12] J. Song, "Structured Composite Multimedia Documents: Design and Presentation in a Distributed Environment," Ph.D. dissertation, Dept. Comp. Sci., Univ. Maryland, College Park, MD, 1997.

Junehwa Song (M'99) received the B.S. degree from Seoul National University, Seoul, Korea, in 1988, the M.S. degree from the State University of New York in 1990, and the Ph.D. degree from the University of Maryland, College Park, in 1997, all in computer science.

He is currently a Research Staff Member at IBM T. J. Watson Research Center. He has been working on various aspects of distributed multimedia systems and compressed video processing.

Dr. Song was awarded an IBM Graduate Fellowship from 1995 to 1997.

Boon-Lock Yeo, photograph and biography not available at the time of publication.