An Integrated Approach for Downscaling MPEG Video

Sudhir Porwal[1] and Jayanta Mukherjee[2]

[1] Image Analysis Center
Defence Electronics Applications Laboratory
Dehradun, India 248001
[2] Dept. of Computer Science
Indian Institute of Technology
Kharagpur, India 721302

Abstract. Digital video databases are widely available in compressed format. In many applications such as video browsing, picture in picture, video conferencing etc. data transfer at lower bit rate is required. This requires downscaling of the video before transmission. The conventional spatial domain approach for downscaling video is computationally very expensive. The computation can greatly be reduced if downscaling and inverse motion compensation (IMC) are performed in Discrete Cosine Transform (DCT) domain. There are many algorithms in the literature to perform IMC in the DCT domain. In this paper, we propose an efficient integrated technique to perform IMC and downscaling in DCT domain. This new approach results in significant improvement in computational complexity.

1 Introduction

Due to the day to day advancement in multi-media based applications, more and more video data are available in digital formats. A digital video is typically stored in the compressed form to reduce storage space and transmission time. International Organization for Standardization (ISO) has proposed MPEG standards [1] for video compression. According to MPEG standard, a video stream consists of a number of GOPs and each GOP is a pre-specified sequence of different kinds of frames. There are three such different types of frames, namely, I, P and B. The I frames are intra coded frame and they are followed by P and B kinds of frames which are known as inter coded frames. They are also called motion compensated frames. There are quite a few good review papers [2] [3] [4] dealing with this standard.

There are applications as video browsing, picture in picture, video conferencing which requires video to be transcoded at lower bit rates and of reduced frame size. A straightforward method to perform this transcoding is to decode each frame in the input video, downscale each frame spatially and re-encode at a lower bit rate. This technique is known as spatial domain downscaling. But the spatial domain technique is very time consuming and computationally inefficient to meet the requirement of real time applications. The DCT/IDCT operations and motion estimation during re-encoding of downscaled video are main bottlenecks of this approach. The computation time can greatly be reduced if we perform downscaling in the DCT domain itself, which eliminates the requirement of costly IDCT operation. There exists many algorithms [5] [6] [7] [8] [9] [10] [11] that provide different approaches for image/video downscaling in the DCT domain. As stated earlier, many frames in an MPEG stream (MPEG-1, MPEG-2) are motion compensated to achieve higher degree of compression. One has to reconstruct these motion compensated frames using inverse motion compensation(IMC) techniques before downscaling. This problem of inverse motion compensation (IMC) in DCT domain was studied in the work of Chang and Messerschmit [9], and subsequently in [12], [13], [14], [15]. Merhav[12] has proposed an excellent scheme to perform IMC in the DCT domain. He has also proposed an efficient computational model for performing this task with the help of factorization of the DCT and IDCT matrices that correspond to fast eight point winograd DCT/IDCT [16]. In [14], the shared information in a macroblock is used to speed up the process of IMC. It shows about 19% and 13.5% improvement over the method presented in [12], [13] respectively. In [15], a Macroblockwise Inverse Motion Compensation (MBIMC) scheme is presented to predict a complete macroblock in single step. This work is extension to the work of Merhav [12]. The method presented in [15] has shown 27% improvement over the Merhav approach.

In this paper, we propose a different approach for video downscaling by combining the downscaling and IMC as a composite operation. This approach is extention of our previous work [15] reported as Macroblockwise Inverse Motion Compensation (MBIMC) scheme. This work formulate a single expression for downscaling and inverse motion compensation. This reduces the computational complexity. We have computed the complexities in terms of multiplication and addition operations. Since multiplication operation is always costlier than addition operation, we assume in our work that a single multiplication is equivalent to 3 machine instructions and addition as a single machine instruction (as considered in [17]). The results are recorded using video stream containing I and P frames only (n=3, m=1 GOP structure is used). However the proposed approach can easily be extended to videos containing B frames also. In the next section (section II), we discuss in brief about MBIMC scheme [15]. In the section III, the integrated approach for downscaling a video with IMC is discussed. Subsequently, results are presented and discussed in section IV.

2 Conversion of a P Frame to an I Frame

In an MPEG video stream, motion compensated frame (P) can be converted to an intra (I) frame by performing the IMC operation. In motion compensation, each macroblock M in current frame is predicted from the previous encoded frame. If predicted macroblock is M' then error E is computed as $E = M - M'$ and finally this error block E is encoded in the video stream. A motion compensated frame is called inter coded frame. The inverse motion compensation is required to convert an inter coded frame (P-frames) to an intra coded frame (I-frames).

During motion compensation, the best matching reference macroblock M' may not be aligned to any macroblock of its reference frame. In general, the predicted macroblock M' may intersect with nine 8×8 blocks (see Figure 2). Our aim is to compute DCT(M) of current macroblock using the fact $M = M' + E$. As DCT(E) is available directly from the compressed stream, we have to compute DCT(M').

In [15], the MBIMC scheme is presented to perform IMC for a macroblock. We discuss the MBIMC scheme here for the sake of completeness.

2.1 Macroblockwise Inverse Motion Compensation (MBIMC)

In MPEG video stream, motion estimation and compensation are performed for each macroblock. A macroblock contains four 8×8 DCT blocks. A motion vector is generated for each macroblock and each 8×8 block in the macroblock shares the same motion vector.

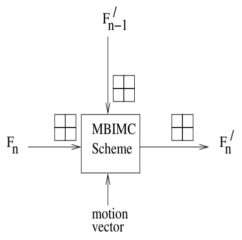

Fig. 1. Block diagram of MBIMC scheme

The functionality of MBIMC scheme is shown in Figure 1. The F'_{n-1} represents the $(n-1)^{th}$ intra/reference frame. F_n is the n^{th} motion compensated inter frame in the video sequence. The MBIMC scheme uses the motion vector and macroblocks from reference and current (inter) frames to perform the IMC for complete macroblock. The IMC operation converts an inter macroblock in to Intra macroblock which can easily be downscaled by any downscaling algorithm.

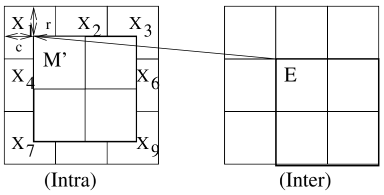

Fig. 2. Macroblockwise inverse motion compensation (MBIMC)

As shown in Figure 2, M' is the predicted macroblock in the reference frame which starts from the location (r, c). Macroblock M' does not always align with the block boundaries and intersects with nine 8×8 DCT blocks in general. In MBIMC scheme, the M' is computed from the nine 8×8 DCT blocks. If $x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9$ are the adjacent blocks in spatial domain then a 16×16 block from the 24×24 block can be extracted using the Eq. (1).

$$m' = c_r \begin{bmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 \end{bmatrix} c_c \tag{1}$$

Where m' is the predicted macroblock in spatial domain and c_r is a 16×24 matrix, c_c is 24×16 matrix. These matrices are different for different values of r and c (refer Figure 2). Since $1 \leq r \leq 8$ and $1 \leq c \leq 8$, there can be eight different c_r and c_c matrices which can be pre-computed and stored.

Since we have DCT blocks $X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9$ and we have to extract macroblock M' from these nine DCT blocks. The macroblock M' is a group of four adjacent 8×8 DCT blocks. To achieve this, Eq. (1) is expressed in the DCT domain as follows:

$$\mathbf{M'} = \begin{pmatrix} S_8 & 0 \\ 0 & S_8 \end{pmatrix} \left\{ c_r \begin{bmatrix} S_8^t & 0 & 0 \\ 0 & S_8^t & 0 \\ 0 & 0 & S_8^t \end{bmatrix} \begin{bmatrix} X_1 & X_2 & X_3 \\ X_4 & X_5 & X_6 \\ X_7 & X_8 & X_9 \end{bmatrix} \begin{bmatrix} S_8 & 0 & 0 \\ 0 & S_8 & 0 \\ 0 & 0 & S_8 \end{bmatrix} c_c \right\} \begin{pmatrix} S_8^t & 0 \\ 0 & S_8^t \end{pmatrix} \tag{2}$$

Here $'0'$ represents a 8×8 matrix of zeros. The matrix multiplication inside the curly braces results in a 16×16 matrix, which represent the spatial domain block. The premultiplication of $\begin{pmatrix} S_8 & 0 \\ 0 & S_8 \end{pmatrix}$ and post multiplication of $\begin{pmatrix} S_8^t & 0 \\ 0 & S_8^t \end{pmatrix}$ results in a 16×16 macroblock containing four 8×8 DCT blocks. Let us define $\mathbf{S}$ and $\mathbf{S^t}$ as shown below.

$$\mathbf{S} = \begin{bmatrix} S_8 & 0 & 0 \\ 0 & S_8 & 0 \\ 0 & 0 & S_8 \end{bmatrix} \text{ and } \mathbf{S^t} = \begin{bmatrix} S_8^t & 0 & 0 \\ 0 & S_8^t & 0 \\ 0 & 0 & S_8^t \end{bmatrix}$$

Then $\mathbf{S^t}$ can be written using the well known factorization of 8 point DCT matrix S as shown in Eq. (3).

$$S_8 = DPB_1B_2MA_1A_2A_3 \tag{3}$$

Here D is a diagonal matrix, P is a permutation matrix, B_1, B_2, A_1, A_2, A_3 are sparse matrices of zeros and ones and M is sparse matrix of real numbers. The details can be seen in [16].

$$\mathbf{S^t} = \underbrace{\begin{bmatrix} (MA_1A_2A_3)^t & 0 & 0 \\ 0 & (MA_1A_2A_3)^t & 0 \\ 0 & 0 & (MA_1A_2A_3)^t \end{bmatrix}}_{\mathbf{Q^t}} \underbrace{\begin{bmatrix} B_2^t & 0 & 0 \\ 0 & B_2^t & 0 \\ 0 & 0 & B_2^t \end{bmatrix}}_{\mathbf{B2^t}} \underbrace{\begin{bmatrix} B_1^t & 0 & 0 \\ 0 & B_1^t & 0 \\ 0 & 0 & B_1^t \end{bmatrix}}_{\mathbf{B1^t}} \underbrace{\begin{bmatrix} P^t & 0 & 0 \\ 0 & P^t & 0 \\ 0 & 0 & P^t \end{bmatrix}}_{\mathbf{P^t}} \underbrace{\begin{bmatrix} D^t & 0 & 0 \\ 0 & D^t & 0 \\ 0 & 0 & D^t \end{bmatrix}}_{\mathbf{D^t}}$$

Similarly, equation for $\mathbf{S}$ is factorized also.

Using the above mentioned notations, we can rewrite Eq. (2) as given below.

$$\mathbf{M'} = \begin{pmatrix} S_8 & 0 \\ 0 & S_8 \end{pmatrix} \left\{ \mathbf{c_r Q^t B2^t B1^t P^t D^t} \begin{bmatrix} X_1 & X_2 & X_3 \\ X_4 & X_5 & X_6 \\ X_7 & X_8 & X_9 \end{bmatrix} \mathbf{DPB1B2Qc_c} \right\} \begin{pmatrix} S_8^t & 0 \\ 0 & S_8^t \end{pmatrix} \tag{4}$$

The Eq. (4) is used to perform IMC for a macroblock in MBIMC scheme. The MBIMC scheme can be referred in detail in [15].

3 Video Downscaling and IMC : Integrated Scheme

Interestingly, the IMC in the DCT domain and downscaling can also be clubbed together in to a single step. The functionality of integrated scheme is described in Figure 3. F'_{n-1} is $(n-1)^{th}$ downscaled reference frame and F_n is next inter frame in the video sequence. Each downscaled reference frame (F'_{n-1}) is upscaled (using [5]) by a factor of two, to reconstruct the next frame (F_n) in the video sequence. The $(n-1)^{th}$ upscaled frame is represented as F''_{n-1}. It may be noted that the upsampling of I frames is not required as it is already available from the compressed video stream. The integrated scheme takes the four 8×8 blocks from F_n and F''_{n-1}, performs IMC and downscaling and generate 8×8 downscaled intra block F'_n.

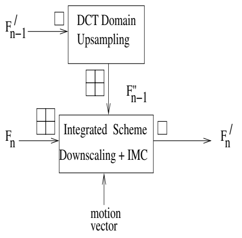

Fig. 3. Integrated scheme for downscaling+IMC

An equation can be derived to perform IMC and downscaling operations simultaneously. But in the integrated approach, we have to perform upsampling of the downscaled frame to convert next inter frame in the video sequence in to intra frame. This upsampling requires additional computation. if we combine the downscaling operation with inverse motion compensation, the Eq. (4) can be written as

$$\mathbf{X'} = \left(\begin{smallmatrix} S_8 & 0 \\ 0 & S_8 \end{smallmatrix}\right) \left\{ \mathbf{d} \left\{ \mathbf{c_r Q^t B2^t B1^t P^t D^t} \begin{bmatrix} X_1 & X_2 & X_3 \\ X_4 & X_5 & X_6 \\ X_7 & X_8 & X_9 \end{bmatrix} \mathbf{DPB1B2Qc_c} \right\} \mathbf{d^t} \right\} \left(\begin{smallmatrix} S_8^t & 0 \\ 0 & S_8^t \end{smallmatrix}\right) \tag{5}$$

where $\mathbf{d}$ is a downscaling filter as shown in Eq. (6).

$$\mathbf{d} = 0.5 \begin{bmatrix} 1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1 \end{bmatrix}_{8\times16} \tag{6}$$

The $\mathbf{d}$ and $\mathbf{d^t}$ matrix multiplication with internal 16×16 IMC block, will yield the 8×8 block. Here DCT represent the 8-point DCT of the resultant block. $\mathbf{X'}$ is the IMC and downscaled version of the actual reference macroblock which can directly be added to downscaled error macroblock $\mathbf{E}$ to get the desired downscaled intra macroblock. An efficient scheme is derived to perform matrix multiplication as given in Eq. (5).

Let us represent

$$\mathbf{J_r} = \mathbf{dc_r Q^t} \qquad 1 \le r \le 8$$
$$\mathbf{K_c} = \mathbf{Qc_c d^t} \qquad 1 \le c \le 8$$

$(\mathbf{dc_r})$ and $(\mathbf{c_c d^t})$ multiplication will generate matrices of size 8×24 and 24×8, which can be calculated a priory (16 such matrices). The sizes of the $\mathbf{J_r}$ and $\mathbf{K_c}$ will be 8×24 and 24×8. Since $\mathbf{J_r}$ and $\mathbf{K_c}$ have similar structure, both require same number of operations to perform matrix multiplication with an arbitrary matrix of size 24×24.

It is observed that $\mathbf{J_r}$ has two different structures depending on whether $\mathbf{r}$ is even or odd (similarly it is true for $\mathbf{K_c}$). We will consider two cases when $\mathbf{r = 2}$ and $\mathbf{r = 5}$, to present our efficient computation model to perform the matrix multiplication of $\mathbf{J_r}$ matrices with an arbitrary matrix. The matrix $\mathbf{J_2}$ is computed and given below.

$$\mathbf{J_2} = \begin{bmatrix}
2 & -2 & 0 & 0 & D & 2A & E & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 2 & -2A & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & -2 & 0 & 0 & -D & -2A & -E & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & A & 1 & C & 0 & -B & -1 & 1 & 1 & A & 1 & C & 0 & B & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 & D & 2A & E & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 2 & -2A & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 & -D & -2A & -E & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & A & 1 & C & 0 & -B & -1 & 1 & 1 & A & 1 & -C & 0 & B & 1
\end{bmatrix}$$

Where A = 0.7071, B = 0.9239, C = 0.3827, D = .5412 and E = 1.3066

For computing $\mathbf{u} = \mathbf{J_2 v}$ where $\mathbf{u} = (\mathbf{u_1}, \dots \mathbf{u_8})^{\mathbf{t}}$ and $\mathbf{v} = (\mathbf{v_1}, \dots \mathbf{v_{24}})^{\mathbf{t}}$, an efficient an efficient scheme is provided below. (Here $'$ and $''$ are used to represents variables. It does not represent derivatives.)

$Y_1 = v_1 + v_2$	$Y_1' = v_9 + v_{10}$	$Y_1'' = v_{17} + v_{18}$
$Y_2 = v_1 - v_2$	$Y_2' = v_9 - v_{10}$	$Y_2'' = v_{17} - v_{18}$
$Y_3 = Av_3$	$Y_3' = Av_{11}$	$Y_3'' = Av_{19}$
$Y_4 = Dv_5$	$Y_4' = Dv_{13}$	$Y_4'' = Dv_{21}$
$Y_5 = Ev_7$	$Y_5' = Ev_{15}$	$Y_5'' = Ev_{23}$
$Y_6 = Cv_5$	$Y_6' = Cv_{13}$	$Y_6'' = Cv_{21}$
$Y_7 = Bv_7$	$Y_7' = Bv_{15}$	$Y_7'' = Bv_{23}$
$Y_8 = Av_6$	$Y_8' = Av_{14}$	$Y_8'' = Av_{22}$
$Y_9 = Y_6 - Y_7$	$Y_9' = Y_6' - Y_7'$	$Y_9'' = Y_6'' - Y_7''$

$$u_1 = 2Y_2 + Y_4 + 2Y_8 + Y_5$$
$$u_2 = 2Y_1 - 2Y_3 - 2v_4$$
$$u_3 = 2Y_2 - Y_4 - 2Y_8 - Y_5$$
$$u_4 = Y_1 + Y_3 + v_4 + Y_9 - v_8 + Y_1' + Y_3' + v_{12} - Y_9' + v_{16}$$
$$u_5 = 2Y_2' + Y_4' + 2Y_8' + Y_5'$$
$$u_6 = 2Y_1' - 2Y_3' - 2v_{12}$$
$$u_7 = 2Y_2' - Y_4' - 2Y_8' - Y_5'$$
$$u_8 = Y_1' + Y_3' + v_{12} + Y_9' - v_{16} + Y_1'' + Y_3'' + v_{20} - Y_9'' + v_{24}$$

Similar exercise could be carried out for J_4, J_6 and J_8 matrices. For $\mathbf{r} = 5$, $\mathbf{J_5}$ matrix is given below.

$$\mathbf{J_5} = \begin{bmatrix}
2 & 0 & -2A & -1 & -2B & -A & -2C & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 2A & 1 & 2C & -A & -2B & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 2A & 1 & -2C & A & 2B & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & -2A & -1 & 2B & A & 2C & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & -2A & -1 & -2B & -A & -2C & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 2A & 1 & 2C & -A & -2B & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 2A & 1 & -2C & A & 2B & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & -2A & -1 & -2B & A & 2C & 0
\end{bmatrix}$$

Where A = 0.7071, B = 0.9239 and C = 0.3827.

To compute $\mathbf{u} = \mathbf{J_5 v}$, we calculate according to following steps. (Here also $'$ and $''$ are used to represents variables. It does not represent derivatives.)

$Y_1 = 2Av_3 + v_4$	$Y_1' = 2Av_{11} + v_{12}$	$Y_1'' = 2Av_{19} + v_{20}$
$Y_2 = 2(B+C)(v_5 + v_7)$	$Y_2' = 2(B+C)(v_{13} + v_{15})$	$Y_2'' = 2(B+C)(v_{21} + v_{23})$
$Y_3 = 2Cv_5$	$Y_3' = 2Cv_{13}$	$Y_3'' = 2Cv_{21}$
$Y_4 = 2Bv_7$	$Y_4' = 2Bv_{15}$	$Y_4'' = 2Bv_{23}$
$Y_5 = Av_6$	$Y_5' = Av_{14}$	$Y_5'' = Av_{22}$
$Y_6 = Y_2 - Y_3 - Y_4$	$Y_6' = Y_2' - Y_3' - Y_4'$	$Y_6'' = Y_2'' - Y_3'' - Y_4''$
$Y_7 = Y_5 + v_8$	$Y_7' = Y_5' + v_{16}$	$Y_7'' = Y_5'' + v_{24}$
$Y_8 = Y_3 - Y_4 - Y_7$	$Y_8' = Y_3' - Y_4' - Y_7'$	$Y_8'' = Y_3'' - Y_4'' - Y_7''$

$$u_1 = 2v_1 - Y_1 - Y_6 - Y_5$$
$$u_2 = 2v_1 + Y_1 + Y_8$$
$$u_3 = 2v_9 + Y_1' - Y_8'$$
$$u_4 = 2v_9 - Y_1' + Y_6' + Y_5'$$
$$u_5 = 2v_9 - Y_1' - Y_6' - Y_5'$$
$$u_6 = 2v_9 + Y_1' + Y_8'$$
$$u_7 = 2v_{17} + Y_1'' - Y_8''$$
$$u_8 = 2v_{17} - Y_1'' + Y_6'' + Y_5''$$

Similar exercise can be done for J_1, J_3 and J_7 matrices. By developing similar implementation schemes of matrix multiplication, the number of operations required to perform matrix multiplication of J_i matrices with an arbitrary matrix of size 24 × 24 are computed and shown in Table 1. The K_c matrices will also require same number of operations to perform matrix multiplication due to similar structures.

Table 1. Multiplication complexities of $\mathbf{J_i}$ matrices in integrated scheme

Matrix	Computations/column
J_1	10m + 34a
J_2	17m + 44a
J_3	14m + 38a
J_4	16m + 43a
J_5	15m + 41a
J_6	17m + 44a
J_7	14m + 38a
J_8	15m + 44a

Let us now compute the total computational complexity for Inverse Motion Compensation and downscaling of a block. Consider a case when $\mathbf{r} = \mathbf{c} = \mathbf{6}$ in Eq. (5). This requires maximum number of computations to perform IMC and downscaling (refer Table 1). It is considered for finding the computational requirements in the worst case. Total operations required to perform IMC and downscaling (when $\mathbf{r} = \mathbf{c} = \mathbf{6}$) using Eq. (5) for each macroblock are 856m + 3496a. It requires 3.34 multiplications and 13.65 additions operations per pixel

Table 2. Computational complexities for downscaling a P frame from CIF resolution to QCIF resolution

Function	Complexity			
	Mults.	Adds	Shifts	Total Cost
Spatial Domain based downscaling				
Input CIF frame processing				
Inverse Quant. + IDCT (144m, 464a per 8 × 8 block)	228096	734976		
Inverse Motion Compensation (256a per 16 × 16 block)		101376		
Output QCIF frame processing				
Downscale by 2 (3a, 1s per pixel)		76032	25344	
Full search ME (±15 pels, 738048a per 16 × 16 block)		73066752		
Motion Compensation (256a per 16 × 16 block)		25344		
DCT + Quant. (144m, 464a per 8 × 8 block)	57024	183744		
Total	285120	74188224	25344	
Total Operation count (Add = 1 op, shift = 1 op, Mult. = 3ops)				**75068928**
DCT domain based downscaling (Using MBIMC)				
Input CIF frame processing				
Inverse Quant. (64m per 8 × 8 block)	101376			
IMC using MBIMC ((3.43m, 20.5a per pixel)	347720	2078208		
Output QCIF frame processing				
DCT Downscale by 2 (1.25m, 1,25a per pixel)	126720	126720		
AMVR (9m, 30a, 1shift per 16 × 16 block)	891	2970	99	
DCT domain MC (3.43m, 20.5a per pixel)	86930	519552		
Quant. (64m per 8 × 8 block)	25344			
Total	688981	2727450	99	
Total Operation count (Add = 1 op, shift = 1 op, Mult. = 3ops)				**4794492**
DCT domain based downscaling (Using Integrated Scheme)				
Input CIF frame processing				
Inverse Quant. (64m per 8 × 8 block)	101376			
IMC using Integrated Scheme ((3.34m, 13.65a per pixel)	338595	1383782		
DCT Upscale by 2 for intermediate frame processing (1.25m, 1,25a per pixel)	126720	126720		
Output QCIF frame processing				
AMVR (9m, 30a, 1shift per 16 × 16 block)	891	2970	99	
DCT domain MC (3.43m, 20.5a per pixel)	86930	519552		
Quant. (64m per 8 × 8 block)	25344			
Total	679856	2033024	99	
Total Operation count (Add = 1 op, shift = 1 op, Mult. = 3ops)				**4072592**

of the input video frame. The computations required per pixel in the integrated approach is less than the MBIMC scheme [15]. In the integrated approach, we have to upsample the resulting downscaled frame to reconstruct the next inter frame in the video sequence. This requirement will add extra computational cost to the integrated scheme. We have used the Dugad and Ahuja [5] approach for upsampling of the frames which requires 1.25 multiplications and 1.25 additions per pixel of the upsampled frame. If we assume that one multiplication is equivalent to three machine instructions and one addition is one machine instruction (refer [17]), the video downscaling using the integrated scheme shows 23% improvement over the MBIMC scheme (The method presented in [5] is used for downscaling with MBIMC scheme).

4 Results

We have implemented the MBIMC scheme presented in [15] and integrated scheme to perform IMC and used Dugad and Ahuja's approach for downscaling/upscaling wherever required. The integrated scheme discussed above performs downscaling and IMC in DCT domain and convert each interframe (P-frame) in to an intraframe (I-frame). We have compared the spatial domain video downscaling system with DCT domain based video downscaling system using the MBIMC scheme and Integrated scheme. The computational comparison of these schemes are shown in Table 2. In the DCT domain based video downscaling methods the AMVR [17] method is used for motion vector re-estimation. The

Table 3. Comparison of Different Technique

	PSNR (dB)								
	Spatial Domain			MBIMC Scheme			Integrated Scheme		
video	Y	U	V	Y	U	V	Y	U	V
Coastguard	25.17	32.54	32.55	26.84	42.69	44.07	26.68	42.38	43.78
Foreman	28.61	32.20	32.08	31.45	40.84	42.45	30.89	40.38	41.80
Container	25.76	32.49	32.78	27.03	40.95	40.46	26.49	39.88	39.20
Tennis	24.98	32.36	31.58	26.82	40.55	41.88	26.19	39.41	40.19

four different MPEG video streams with only I and P frames are used to record the results. These video streams are downscaled to QCIF resolution (at 500 kbps) from CIF resolution (at 1.5 mbps). To compute the PSNR, each frame from downscaled QCIF video stream is upsampled to CIF resolution and then compared with the original video frame. In spatial domain technique, frames are upsampled using 'bilinear' technique and in other DCT domain based techniques, frames are upsampled in DCT domain using Dugad and Ahuja's technique [5]. The average PSNR values for different video streams are shown in the Table 3.

In Table 3, we can observe that average PSNR values of Integrated approach are much higher than the spatial domain approach and very close to MBIMC scheme. It is obvious from Table2 that the MBIMC scheme is approximately 15 times faster than the spatial domain technique, however the integrated scheme shows 23% improvement over the MBIMC scheme.

5 Conclusion

In this paper, we propose a integrated scheme to perform inverse motion compensation (IMC) and downscaling directly on DCT blocks of an MPEG video stream. This scheme performs IMC and downscaling over a complete macroblock in a single step. It also uses the factorization of the DCT/IDCT matrices to reduce computational complexity of the IMC operation. A fast mathematical model is proposed to perform the computations efficiently. The integrated scheme shows 23% improvement for downscaling operation over the MBIMC scheme.

References

1. Coding of Moving Pictures and Associated Audio for Digital Storage Media up to 1.5 bits/s, ISO/IEC JTC1 CD 11172, 1992.
2. Coding of Moving and Associated Audio. Committee Draft of Standard ISO 11172: ISO/MPEG 90/176, Dec. 1990.
3. Video Codec for Audio Visual Services at px64 Kbits/s. CCITT Recommendation H. 261, 1990.
4. D. le Gall, MPEG : A video compression standard for multimedia applications, Commun. ACM, vol. 34, no. 4, pp. 47-58, Apr, 1991.
5. R Dugad and N. Ahuja, A fast scheme for image size change in the compressed domain, IEEE Trans. Circuits Syst. Video Technology, vol. 11, pp. 461-474, Apr. 2001.

6. Jayanta Mukherjee, Sanjit K. Mitra, Image resizing in the compressed domain using subband DCT, IEEE Trans. Circuits Syst. Video Technology, vol 12, No. 7, July 2002.
7. Jayanta Mukherjee, S. K. Mitra, Resizing of images in the DCT space by arbitrary factors, IEEE Int. Conf. on Image Processing (ICIP), Singapore, 2004, pp. 2801-2804.
8. Jayanta Mukherjee, S. K. Mitra, Arbitrary resizing of images in DCT space, Communicated to IEE Proc. Vis. Image Signal Process.
9. S. F. Chang and D. G. Messerschmitt, Manipulation and compositing of MC-DCT compress video, IEEE J. Select. Areas Commun., vol. 13, pp. 1-11, Jan. 1995.
10. B. C. Smith and L. Rowe, Algorithms for manipulating compressed images, IEEE Comput. Grap. Applicat. Mag., vol. 13, pp. 34-42, Sept. 1993.
11. Q. Hu and S. Panchanathan, Image/video spatial scalability in compressed domain, IEEE Trans. Ind. Electron., vol. 45, pp. 23-31, Feb. 1998.
12. N. Merhav and V. Bhaskaran, Fast algorithms for DCT-domain image downsampling and for inverse motion compensation, IEEE Trans. Circuits Syst. Video Technology, vol. 7, pp. 468-476, June 1997.
13. P. A. A. Assuncao and M. Ghanbari, "Transcoding of MPEG-2 video in the frequency domain," in ICASSP 1997, 1997, pp. 2633-2636.
14. Junehwa Song, Boon-Lock Yeo, "A Fast Algorithm for DCT-Domain Inverse Motion Compensation Based on Shared Information in a Macroblock," IEEE Trans. on Circuits and Systems for Video Technology, vol. 10, No. 5, Aug. 2000.
15. Sudhir Porwal, Jayanta Mukhopadhyay, "A Fast DCT Domain Based Video Downscaling System," IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP), Toulouse, France, 2006, pp. 885-888.
16. Y. Arai, T. Agui, and M. Nakajima, A fast DCT-SQ scheme for images, Trans. IEICE, vol. E 71, no. 11, p. 1095, Nov. 1998.
17. Bo Shen, Ishwar K. Sethi, Bhaskaran Vasudev, Adaptive Motion Vector Resampling for Compressed Video Downscaling, IEEE Trans. Circuits Syst. Video Technology, Vol. 9, No. 6, Sept., 1999.