Hakkı Alparslan ILGIN[1], Hakkı Gökhan İLK[1], Miroslav VOZNAK[2], Luis F. CHAPARRO[3]

Ankara University (1), VSB-Technical University of Ostrava (2), University of Pittsburgh (3)

Fast DCT Image Resizing and Video Compositing for Multi-Point Video Conferencing

Abstract— *Video conferencing is a real time process that requires fast operations. In this paper a Discrete Cosine Transform (DCT)-based video compositing structure for multi-point video conferencing with a fast DCT method for integer and rational image resizing is presented. Whole compositing process is kept in the DCT domain to avoid comprehensive operations. Our procedure does not only improve speed but also the quality of the resized video frames. Proposed method is illustrated and compared with other methods by means of simulation results with different video sequences.*

Streszczenie. *W artykule przedstawiono metodę składania obrazów, pochodzących z wielu źródeł, na potrzeby wideo-konferencji. Działanie oparte jest na dyskretnej transformacji kosinusowej, którą wykorzystano do całościowego i wymiernego skalowania obrazu. Opracowany algorytm poprawia szybkość działania oraz jakość skalowania. Dokonane zostało symulacyjne porównanie działania z innymi metodami. (**Szybkie skalowanie i składanie obrazów wideo z wykorzystaniem dyskretnej transformacji kosinusowej, na potrzeby wielopunktowej wideo-konferencji**).*

Keywords: Multi-point video conferencing, video compositing, image resizing, DCT decimation, DCT interpolation, significance map coding, zerotree coding, transcoding.
Słowa kluczowe: wielopunktowa wideo-konferencja, składanie obrazów wideo, skalowanie obrazu, dziesiątkowanie DCT, interpolacja DCT, kodowanie mapy istotności, zerotree, transkodowanie.

Introduction

Real time video coding applications demands low computational complexity. Being one of these applications, multi-point video conferencing involves compositing of video sequences which come into a node point. Decoding, decimation and re-encoding are the main processes of video compositing. Inverse and forward Discrete Cosine Transform (DCT) and recalculation of motion estimation and compensation are the most computationally expensive operations of the whole process. Regular approach consists of hybrid decoding, spatial domain decimation and compositing and hybrid encoding as seen in Fig. 1. DCT based video compositing methods which use DCT transcoders and decimation are performed for faster and efficient implementation [6-8]. In this paper, we consider DCT compositing approach. A general scheme for fast DCT image resizing, which is previously considered in [11] and [13], is also shown. We then use significance map coding, which became a popular quantization scheme with embedded zerotree wavelet encoding in [1] and SPIHT in [4], to encode the DCT coefficients obtained from the composited video frames. To decimate an image by N, NxN array of 8x8 DCT blocks is transformed into one DCT matrix of size 8Nx8N. Transformed matrix is then masked to obtain the low-frequency DCT coefficients, which represent the decimated video frame. Transformation of an array of DCT blocks into one DCT block was introduced in [14]. Very fast transformation matrices can be found leading to very fast transformations. Similar characteristics were first discussed in [13] for decimation by 2. In this paper a general approach is shown. Proposed algorithms allow fast decimation for integer or rational factors. When decimation factor is rational, an additional transformation to obtain an array of 8x8 blocks for the masked array is needed. Since high-frequency DCT coefficients are generally very small or zero, efficiency of our algorithm is increased by using only the low-frequency DCT coefficients.

To expedite the compositing, we use motion vector information from incoming videos to estimate -instead of to compute- the motion vectors for composited video. Likewise, by means of significance map encoder we adapt the quantizer to improve the image quality for a desired bit rate, thus attaining bit rate control. In the next section, we briefly review DCT compositing. Then a fast way to decimate and composite video frames in the DCT domain is shown. Significance map coding of DCT coefficients and its advantage is discussed in the quantization of composited videos section. Simulation results and conclusions are presented in the last two sections.

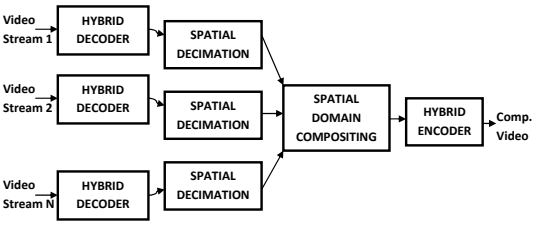

Fig. 1. Compositing in the spatial domain

Transcoding and DCT Compositing

Video streams from different sources are decoded by DCT domain transcoders, and then decimated and composited into one video stream as indicated in Fig. 2. Transcoders convert incoming video streams into consecutive DCT images which result in decreased processing time since there is no inverse DCT in the DCT transcoder. This is opposite of a conventional hybrid decoder, which requires expensive inverse DCT operations to reconstruct video frames in the spatial domain [6-8].

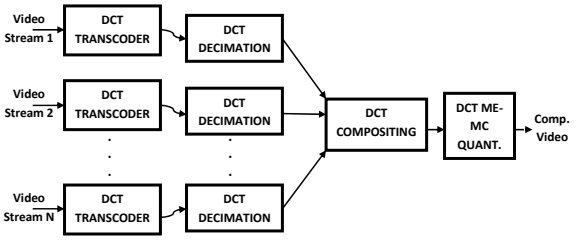

Fig. 2. Transcoding and compositing in the DCT domain

In DCT transcoder, staying in the DCT domain demands computing new best matched namely optimal DCT blocks for motion compensation. Optimal block may not coincide with previously computed DCT blocks. Therefore it needs to be calculated using adjacent DCT blocks. There exist three overlapping cases for an optimal block: (1) fully overlapping with one block, (2) partially overlapping with two vertical or horizontal blocks, and (3)

partially overlapping with four blocks as shown in Fig. 3. Here shaded blocks represent optimal blocks. Inverse DCT of the overlapping DCT blocks and then forward DCT of overlapping area are computed to obtain optimal DCT block for the last two cases in conventional way. However, operations do not remain completely in the DCT domain and increases the number of operations [7]. To decrease the computational complexity and thus to stay in the DCT domain, DCT windowing and shifting operations are used to obtain optimal DCT blocks as shown in Fig. 4. Using windowing and shifting matrices improves the speed of the DCT motion compensation [7, 9]. However sparser DCT windowing and shifting matrices as suggested by [10, 11] requires less operations.

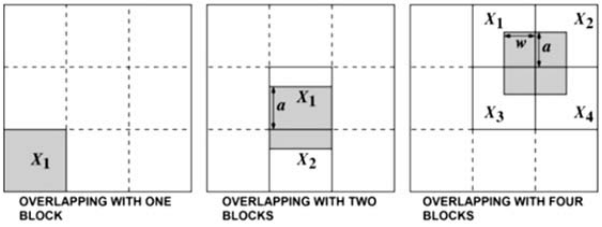

Fig. 3. Overlapping cases for an optimal block

Additionally the more the first case is met the faster the transcoding is implemented. In this case, motion vectors are either zero or integer multipliers of a block size, meaning that no computation is required to obtain the DCT of the optimal block.

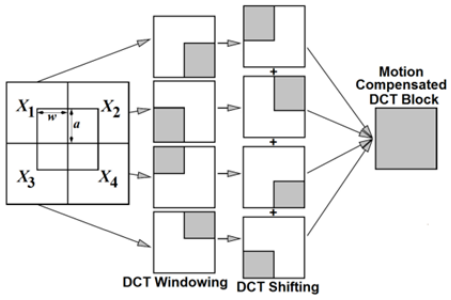

Fig. 4. DCT windowing and shifting for motion compensation in the DCT domain

The speed of the DCT compositing system is increased further by decreasing decimation/interpolation complexity using the proposed algorithms. Some examples illustrating decoding time improvements by regular [7] and fast [11] DCT transcoders are shown in Table 1.

Table 1. Average speed improvements by DCT transcoder and fast DCT transcoder over hybrid decoder

Video Sequence	Average Speedup (%)	
	DCT Transcoder	Fast DCT Transcoder
Claire	22.8	47.6
Salesman	33.3	43.3
Hall	33.7	43.1

Fast transcoders [11] are utilized to obtain reconstructed DCT frames. Decimation is done with the proposed low complexity algorithm. Decimated DCT frames are then put together to construct composited DCT frames. Here the absence of the forward DCT operation, which is a must in hybrid-domain compositing systems, is another improvement of the DCT domain approach [6]. Motion vectors associated with the blocks of the composited video are estimated using motion vectors of the input videos resulting in lower complexity motion estimation [12]. The quantization scheme is significance map coding of the DCT coefficients, which is explained in the corresponding section.

Transformation and Decimation of DCT Blocks

Composited video frame is formed of decimated subframes. Many of the video compositing standards require decimation by an integer factor, but some also require decimation by a rational factor to get the mixed views as shown in Fig. 5. To pursue operations in the compressed domain, it is thus necessary to have efficient decimation and interpolation methods. The first issue in the DCT decimation is the transformation of an array of DCT blocks into one block. Suppose we have an array of *N*x*N* DCT blocks, where the integer $N \geq 2$ is the decimation factor, each of size 8x8, and we wish to transform it into one DCT block of dimension 8*N*x8*N*. This corresponds to finding the inverse DCT of each of the 8x8 blocks, arranging them into one array of dimension 8*N*x8*N*, and finally finding its 8*N* two-dimensional DCT. A more efficient approach was proposed in [14], which as we show can be attained by means of an orthonormal matrix transformation. The problem is to find an orthonormal transformation matrix T^{8N}, i.e., $(T^{8N})^t\, T^{8N}=I^{8N}$, where I^{8N} is 8*N*x8*N* identity matrix and *t* is the matrix transform operators,

$$X^{8N} = T^{8N} \begin{bmatrix} X_{11}^{8} & \cdots & X_{1N}^{8} \\ \vdots & \ddots & \vdots \\ X_{N1}^{8} & \cdots & X_{NN}^{8} \end{bmatrix} (T^{8N})^t \tag{1}$$

where $\{X_{ij}^{8}\}_{i,j=1,\ldots,N}$ and X^{8N} are 8x8 and 8*N*x8*N* two-dimensional DCT blocks respectively. By definition, if the *M*x*M* DCT operation matrix is obtained as

$$S^M = \{s(k,n)\}_{k,n=0}^{M-1} = \{0.5c(k)\cos(\tfrac{(2n+1)k\pi}{2M})\} \tag{2}$$

then the two-dimensional DCT of the spatial domain matrix $x = \{x(n,m)\}_{n,m=0}^{M-1}$ is given by

$$X = \{X(k,l)\}_{k,l=0}^{M-1} = S^M x (S^M)^t. \tag{3}$$

Fig. 5. Composited frames; (top) six subframes, five with decimation factor *N*=3 and one with decimation factor *N*=2/3, (bottom) sixteen subframes with decimation factor *N*=4

The inverse DCT of the matrix *X* will be as follows:

$$\mathrm{x} = (\mathrm{S}^{\mathrm{M}})^{\mathrm{t}}\mathrm{X}\mathrm{S}^{\mathrm{M}}. \tag{4}$$

DCT transformation can be easily obtained by finding the DCT of the identity matrix I^{8N} (each $X_{ii}^{8} = I^{8}$ and $X_{ij}^{8} = 0, i \neq j$) using the forward DCT definition in (3) and orthonormality property of S^{M} so that

$$\mathrm{T}^{8\mathrm{N}} = \mathrm{S}^{8\mathrm{N}}\mathrm{diag}[(\mathrm{S}^{8})^{\mathrm{t}}]. \tag{5}$$

To obtain the forward block transformation, let the 8*N*x8*N* matrices S^{8N} and T^{8N} be expressed as follows

$$S^{8N} = [S_1^{8N}\ S_2^{8N}\ \cdots\ S_N^{8N}]$$

$$T^{8N} = [T_1^{8N}\ T_2^{8N}\ \cdots\ T_N^{8N}], \tag{6}$$

where the sub-blocks

$$\{T_i^{8N} = S_i^{8N}(S^{8})^{t}\} \tag{7}$$

have 8*N*x8 size. Thus the representation of X^{8N} in terms of the X_{ij}^{8}, or the forward block transformation, is as folllows:

$$\mathrm{X}^{8\mathrm{N}} = \sum_{\mathrm{i}=1}^{\mathrm{N}}\sum_{\mathrm{j}=1}^{\mathrm{N}}\mathrm{T}_{\mathrm{i}}^{8\mathrm{N}}\mathrm{X}_{\mathrm{ij}}^{8}(\mathrm{T}_{\mathrm{j}}^{8\mathrm{N}})^{\mathrm{t}}. \tag{8}$$

Therefore 8*N*x8*N* X^{8N} DCT block is obtained directly from an *N*x*N* array of 8x8 DCT blocks. The decimation can now be obtained by masking the resulting block to obtain more significant DCT coefficients. The inverse transformation of each of the 8x8 DCT blocks is given by

$$X_{ij}^{8} = (T_i^{8N})^{t}X^{8N}T_j^{8N} \tag{9}$$

using the orthonormality of the DCT transformation. This will allow us to perform interpolation.

To simplify the notation let us consider *N*=2. Equations for other cases for *N*>2 can be easily extended. Applying the direct transformation, we obtain a 16x16 DCT block from four 8x8 given DCT blocks. Masking the X^{16} matrix to extract the low-frequency coefficients we obtain the decimated DCT array as

$$\begin{aligned} X_d &= [I^8\ 0]X^{16}[I^8\ 0]^t \\ &= \sum_{i=1}^{2}\sum_{j=1}^{2}F_i^8X_{ij}^8(F_j^8)^t \end{aligned} \tag{10}$$

where $\{F_i^8 = [I^8\ 0]T_i^{16}\}_{i=1,2}$ are of size 8x8. The above equation can be efficiently implemented using the sum and difference representations of $\{T_i^{16}\}$;

$$\begin{aligned} D_1^{16} &= 0.5(T_1^{16} + T_2^{16}) \\ D_2^{16} &= 0.5(T_1^{16} - T_2^{16}) \end{aligned} \tag{11}$$

where D_1^{16} and D_2^{16} are 16x8 transformation matrices which are obtained as follows:

$$\begin{aligned} D_1^{16}(i,j) &= \begin{cases} T_1^{16}, (i+j)\ \text{even} \\ 0, (i+j)\ \text{odd} \end{cases} \\ D_2^{16}(i,j) &= \begin{cases} T_1^{16}, (i+j)\ \text{odd} \\ 0, (i+j)\ \text{even} \end{cases} \end{aligned} \tag{12}$$

Replacing the $\{T_i^{16}\}$ in terms of $\{D_i^{16}\}$ matrices in the direct transformation we obtain transformed matrix,

$$X^{16} = [Y + Z](D_1^{16})^{t} + [Y - Z](D_2^{16})^{t}$$

$$Y = D_1^{16}(X_{11}^8 + X_{21}^8) + D_2^{16}(X_{11}^8 - X_{21}^8)$$

$$Z = D_1^{16}(X_{12}^8 + X_{22}^8) + D_2^{16}(X_{12}^8 - X_{22}^8) \tag{13}$$

Forward block transformation in (13) demands less operations than the transformation in (8) since sparseness of $\{T_i^{16}\}$ matrices makes $\{D_i^{16}\}$ matrices sparser, which can be seen in (12). Accordingly decimated DCT block X_d is efficiently obtained as

$$X_d = [Y_d + Z_d](A_1^8)^{t} + [Y_d - Z_d](A_2^8)^{t}$$

$$Y_d = A_1^8(X_{11}^8 + X_{21}^8) + A_2^8(X_{11}^8 - X_{21}^8)$$

$$Z_d = A_1^8(X_{12}^8 + X_{22}^8) + A_2^8(X_{12}^8 - X_{22}^8) \tag{14}$$

where the $\{A_i^8 = [I^8\ 0]D_i^{16}\}_{i=1,2}$ matrices of size 8x8 are sparser than the $\{F_i^8\}$ matrices defined in (10), and make the decimation faster. Operations can be faster since high frequency AC coefficients in a DCT block are typically very small or zero. Therefore when the coefficients are set to zero, inverse DCT gives a very close result to the original spatial domain block. Accordingly consider that the DCT blocks to be decimated have *q*x*q* ($1 \leq q \leq 8$) nonzero low-frequency AC coefficients and we set the rest to zero, i.e.,

$$\mathrm{X}_{\mathrm{ij}}^{8} = \begin{bmatrix} \mathrm{X}_{\mathrm{ij}}^{\mathrm{q}} & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} \mathrm{I}^{\mathrm{q}} \\ 0 \end{bmatrix}\mathrm{X}_{\mathrm{ij}}^{\mathrm{q}}[\mathrm{I}^{\mathrm{q}}\ 0]. \tag{15}$$

Replacing these blocks in (10) gives decimated DCT block as follows:

$$X_d^q = \sum_{i=1}^{2}\sum_{j=1}^{2}G_i^8X_{ij}^q(G_j^8)^{t} \tag{16}$$

where X_d^q is 8x8 decimated DCT block and $\{G_i^8 = F_i^8[I^q\ 0]^t\}_{i=1,2}$, $1 \leq q \leq 8$. $\{G_i^8\}$ matrices of size $8\mathrm{x}q$ clearly have fewer entries than $\{F_i^8\}$. However decimated DCT block can be obtained faster if the decimation in (14) is used:

$$X_d^q = \left[Y_d^q + Z_d^q\right](B_1^q)^{t} + \left[Y_d^q - Z_d^q\right](B_2^q)^{t}$$

$$Y_d^q = B_1^q\left(X_{11}^q + X_{21}^q\right) + B_2^q\left(X_{11}^q - X_{21}^q\right) \tag{17}$$

$$\mathrm{Z}_{\mathrm{d}}^{\mathrm{q}} = \mathrm{B}_1^{\mathrm{q}}\left(\mathrm{X}_{12}^{\mathrm{q}} + \mathrm{X}_{22}^{\mathrm{q}}\right) + \mathrm{B}_2^{\mathrm{q}}\left(\mathrm{X}_{12}^{\mathrm{q}}\text{-}\mathrm{X}_{22}^{\mathrm{q}}\right)$$

where $\{B_i^q = A_i^8[I^q\ 0]^t\}_{i=1,2}$. Therefore $\{B_i^q\}$ matrices have fewer entries than $\{F_i^8\}$ and also $\{A_i^8\}$. The proposed DCT block transformation and decimation for *N*=2 is shown in Fig. 6.

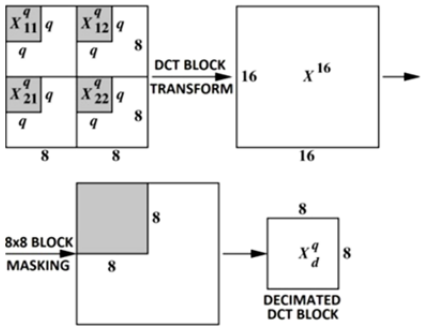

Fig. 6. Proposed integer DCT decimation for decimation factor *N*=2

When we compare the proposed method to others, our algorithm (when *q*=4) has the lowest complexity with 1.25 multiplications and 1.25 additions per pixel [17] as shown in Table 2.

Table 2. Computational complexity of decimation methods for *N*=2

Method	Multipl./Pixel	Add./Pixel
Prop. (*q*=4)	1.25	1.25
[13]	1.25	1.25
[16]	1.00	2.00
[15]	1.89	2.06
[7]	4.00	4.75
Spatial	3.44	9.82

Since the computational complexity of the decimation method in [13] is the same with ours, we compare it with our method in terms of Peak Signal to Noise Ratio (PSNR) in Table 3. As expected in our method, the larger the *q* is (i.e., 4≤*q*≤8), the better the interpolation as seen in Table 3, but the more complex the implementation. In Table 3, our method with first *q*=4 has the same number of operations for both decimation and interpolation as in [13]. As seen in the table, PSNR results of [13] and our method (*q*=4 (i)) are very close. Also, with a slight increase in the number of operations in the interpolation while keeping the complexity of the decimation as before, the PSNR values of the interpolated frames are improved (see *q*=4 (ii) in the table). For *q*=8, complexity of our algorithm increases.

Table 3. PSNR comparisons of the proposed resizing method with the method in [13] for *N*=2

Video Frame	[13]	*q*=4 (i)	*q*=4 (ii)	*q*=8
Miss America	38.97	38.88	39.13	39.43
Crew	33.65	33.51	33.70	34.04
Ice	33.43	33.17	33.60	34.11
Foreman	32.74	32.58	32.82	33.19
Hall	28.64	28.48	28.76	29.17
Container	27.32	27.11	27.51	28.02
Bus	27.28	27.00	27.31	27.73

When *N*>2, decimation proceeds in a very similar way to decimation by *N*=2. In general, for an integer decimation factor *N* we transform an array of *N*x*N* DCT blocks of size 8x8 into one array of size 8*N*x8*N*, and then mask it to obtain the decimated 8x8 DCT. For example for *N*=3, 24x24 DCT matrix is obtained as

$$X^{24} = (U+Y)(D_1^{24})^t + (W-Y)(D_2^{24})^t + (U-W)(D_3^{24})^t,$$

$$U = D_1^{24}(X_{11}^8 + X_{21}^8) + D_2^{24}(X_{31}^8 - X_{21}^8) + D_3^{24}(X_{11}^8 - X_{31}^8),$$

$$Y = D_1^{24}(X_{12}^8 + X_{22}^8) + D_2^{24}(X_{32}^8 - X_{22}^8) + D_3^{24}(X_{12}^8 - X_{32}^8),$$

$$W = D_1^{24}(X_{13}^8 + X_{23}^8) + D_2^{24}(X_{33}^8 - X_{23}^8) + D_3^{24}(X_{13}^8 - X_{33}^8). \quad (18)$$

When *N*>2, it is also possible to represent each 8x8 block with *q*x*q* top-left coefficients (*q*<8) for faster operation. Therefore for *N*=3, decimated 8x8 DCT block will be

$$X_d^q = (U_d^q + Y_d^q)(D_1^q)^t + (W_d^q - Y_d^q)(D_2^q)^t + (U_d^q - W_d^q)(D_3^q)^t,$$

$$U_d^q = D_1^q(X_{11}^q + X_{21}^q) + D_2^q(X_{31}^q - X_{21}^q) + D_3^q(X_{11}^q - X_{31}^q),$$

$$Y_d^q = D_1^q(X_{12}^q + X_{22}^q) + D_2^q(X_{32}^q - X_{22}^q) + D_3^q(X_{12}^q - X_{32}^q),$$

$$W_d^q = D_1^q(X_{13}^q + X_{23}^q) + D_2^q(X_{33}^q - X_{23}^q) + D_3^q(X_{13}^q - X_{33}^q). \quad (19)$$

As before computational complexity is related to the value of *q*, and PSNR is better for larger values of *q*. However for decimation factors, such as *N*=3 or *N*=4, with even small values of *q* the quality of the interpolated images does not differ visually or in terms of PSNR as shown in Table 4. For example, for *N*=4, the PSNR results are the same for *q*=4, 5, 6 and 8, and are very close to the PSNR for *q*=2 and 3. Thus only the 2x2 low-frequency parts of the 8x8 DCT blocks may be sufficient to decimate DCT blocks. This constitutes a significant computational saving.

Table 4. PSNR comparisons of a Claire frame for different decimation factors with several DCT block sizes

	Method					
N	*q*=2	*q*=3	*q*=4	*q*=5	*q*=6	*q*=8
2	28.99	30.62	33.93	34.10	34.16	34.16
3	28.90	30.12	30.32	30.33	30.32	30.33
4	28.84	29.25	29.30	29.30	29.30	29.30
2/3	28.66	30.30	34.10	35.04	37.07	37.08

PSNR comparisons for decimated/interpolated (*N*=2 and *N*=4, 1≤*q*≤8) frames from different video sequences are shown in Fig. 7 and Fig. 8. When *q*=1 only the DC values are used to obtain the decimated frames and AC coefficients are set to zero. In this case, it is still possible to see a coarse reconstructed image. For the case of *N*=2, *q*=4 is the most appropriate value, since there is no significant PSNR improvement for *q*>4 as seen in Fig. 7. This shows the efficiency of the proposed method which uses smaller portions of DCT blocks efficiently. When *N* is 4, only 2x2 portions of the DCT blocks are enough to resize the frames since PSNR does not increases dramatically after *q*=2 as seen in the graphic in Fig. 8. The same idea is valid for the other resizing options either integer or rational. It is also possible to decimate by a rational number such as *N*=2/3 or *N*=3/4. For instance when *N*=2/3, we first need to transform a 3x3 array of 8x8 DCT blocks into one of size 24x24. The masking to get 2/3 gives us a DCT block of dimension 16x16, that needs to be converted into final four 8x8 blocks. These required additional computations increase the complexity of the decimation for rational factors.

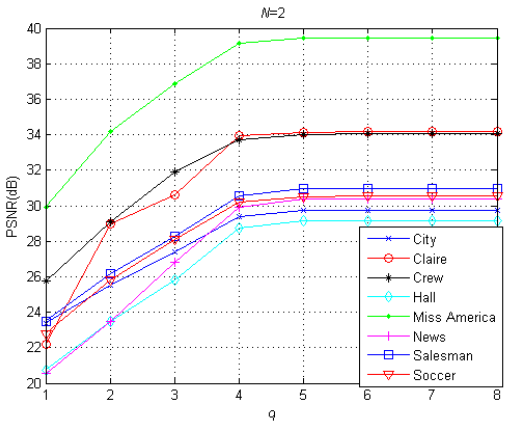

Fig 7. PSNR results for resizing by 2 using different DCT block sizes

In rational decimation, only *q*x*q* parts of DCT blocks need to be used to obtain 24x24 DCT block as shown in Fig. 9 similarly. This substantially reduces the number of operations. For instance, if all coefficients of the 8x8 DCT blocks are used, which corresponds to the case where *q*=8, the number of multiplications and additions per pixel to obtain the final four 8x8 DCT blocks are 21.58 and 20.81 respectively. If only 6x6 portions of the input DCT blocks are used (*q*=6), operations reduce to 18.38 multiplications and 17.16 additions per pixel. In terms of PSNR, there is almost no difference between *q*=6 and *q*=8 cases as shown in Table 4. For further computational saving *q* can be decreased with a slight PSNR decrease as seen in Table 4.

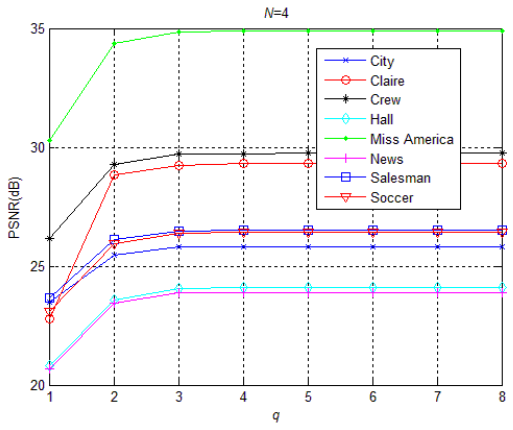

Fig 8. PSNR results for resizing by 4 using different DCT block sizes

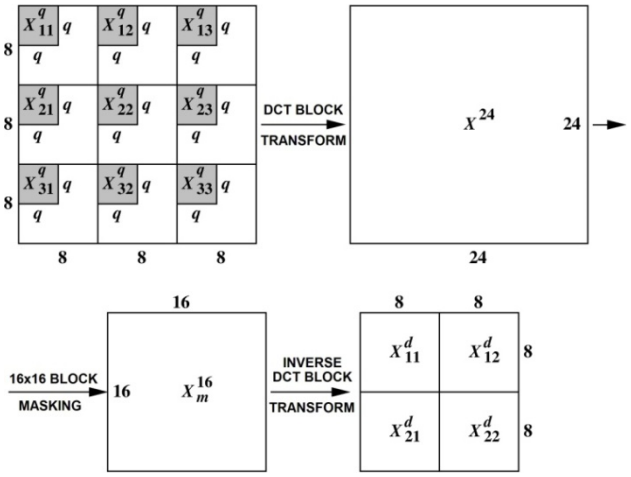

Fig. 9. Proposed rational DCT decimation (*N*=2/3)

Quantization of Composited Videos

We apply quantization basically by replacing quantization parameter scheme with significance map coding in order to improve the quality of the composited video. We apply DCT based embedded zerotree coding (DCT-EZT) which is previously used for wavelet [1] and then for DCT [2, 3, 19]. In [4], called SPIHT, compression ratio is improved. It has also efficient differential approach in a recent study [5]. To encode the DCT coefficients by a significance map coder they are rearranged into a hierarchical subband structure exploiting the dependency between them. We arrange video frames consisting of 8x8 DCT blocks into a 3-scale hierarchical structure resulting in ten subbands as shown in Fig. 10. The highest subband, LL_3, have all DC coefficients. AC coefficients are distributed into other subbands. Since the structure is the same as the wavelet coefficients HL_i, LH_i and HH_i includes horizontal-, vertical- and diagonal-detail AC-DCT coefficients. Gray arrows show importance order of the subbands in Fig. 10. Obtained symbols by the significance map encoder are also scanned in the same order. DC coefficients contain most of the energy of the frame. Therefore the quality of the decoded image depends in great part on the DC coefficients. Thus DC coefficients are firstly encoded and then the low to high frequency AC coefficients follow.

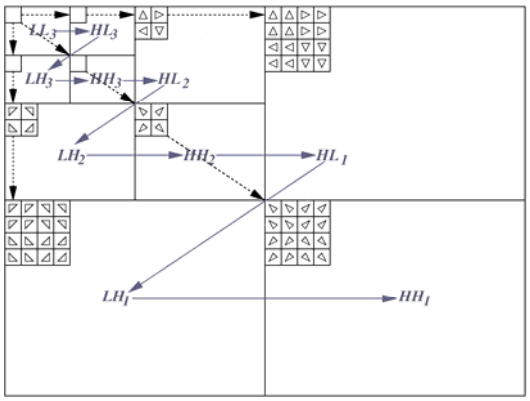

Fig. 10. Arranged DCT coefficients in 3-scale subband structure

Zerotree coding starts with measuring the significance of all coefficients by comparing them with an initial threshold given as

$$\mathrm{Th}_0 = 2^{\mathrm{n}} \tag{20}$$

where $n = \lfloor \log_2(\max(|X_{k,m}|)/2 \rfloor$ and $X_{k,m}$ is DCT coefficient of a *K*x*M* frame. If the magnitude of a coefficient is greater than the threshold, it is considered significant. The advantage of this type of coding is that if a group of coefficients related to each other as tree in different subbands is found to be insignificant, only one symbol is sent to decoder stating that the tree has coefficients lower than the threshold called zerotree.

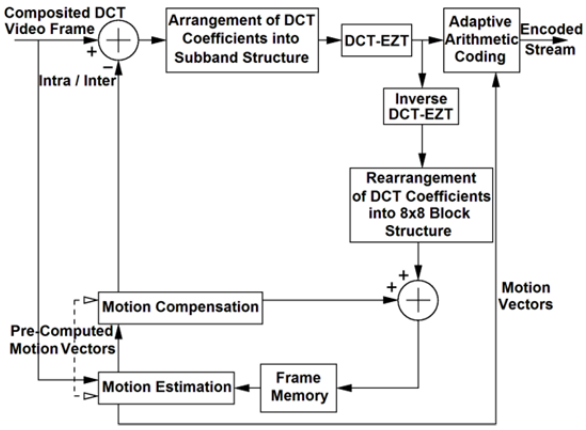

Fig. 11. Structure of DCT-EZT video coder

This contributes compression ratio [1]. After comparing all coefficients with threshold, a subordinate scan is performed to give more precision to significant coefficients [1-4]. At the next step *n* is decreased by 1, namely threshold is halved after subordinate scan finishes. Then insignificant coefficients are compared to this new threshold. Algorithm iteratively continues in the way as explained above. As soon as a symbol is outputted by the zerotree coder, it is encoded by an adaptive arithmetic encoder [18] to obtain embedded bit stream and more compression [1]. Process continues until the desired bit rate is reached. Proposed encoder structure is given in Fig. 11.

Simulations and Results

In the experiments, we first composited four different video sequences into one video stream. In this case frames of each video sequence are decimated by 2. The resulting composited video frames consist of four subframes. In the second compositing structure, we composited six different video sequences into one. Here composited frames are composed of five subframes decimated by 3, and one subframe decimated by rational factor 2/3. In the last case we composited sixteen frames decimated by 4 into one. The proposed algorithms can be also applied to other decimation and compositing cases. All operations are kept in the DCT domain as shown in Fig. 2. Each video sequence has an intra-frame and 49 inter-frames in CIF (Common Intermediate Format) which has 352x288 resolution. PSNR results shown in Table 5 are the average of the PSNR of total 50 frames. In all compositing structures we obtain better PSNR values by using DCT-EZT coding than by using regular scalar quantization scheme with different quantization parameters (QP) as shown in Table 5. The reconstructed frames are also visually better in DCT-EZT case. Namely blocking effects are lesser and details are better especially at low bit rates. It is further possible to improve the quality of the quantization process by using the approach in [5].

Table 5. PSNR results for composited videos

Structure	Method	PSNR (dB)		
4-subframe	*Regular*	*QP*=5	*QP*=10	*QP*=20
		36.25	32.48	28.25
	Zerotree	38.20	34.77	31.42
6-subframe	*Regular*	*QP*=5	*QP*=10	*QP*=20
		36.61	33.11	29.07
	Zerotree	39.00	35.50	32.78
9-subfame	*Regular*	*QP*=5	*QP*=10	*QP*=20
		35.30	31.40	26.99
	Zerotree	37.16	33.40	29.61

Conclusion

We perform compositing for multi-point video conferencing fully and efficiently in the DCT domain. Resizing processes including proposed decimation and interpolation in the DCT domain increases the quality of video frames by decreasing aliasing. To decrease the computational complexity *q* can be set to smaller values in decimation and interpolation without affecting the image quality significantly. Quantization parameter scheme is replaced by significance map coding that achieves bit rate control and better PSNR. Our methods can be used in latest video coding standards such as H.264 SVC to achieve efficient implementations at several scales. Bit rate allocation to specific subframes or objects can be applied in order to increase the quality of the interested video stream(s) or object(s) by the viewer sides. Also bit rate optimization to distribute bits efficiently among the composited frames of a GOP (Group of Frames) can increase the average PSNR of a video stream [20].

REFERENCES

[1] J. M. Shapiro, ``Embedded Image Coding Using Zerotrees of Wavelet Coefficients,'' IEEE Trans. Signal Proc., Vol. 41, No. 12, pp. 3445-3462, Dec. 1993.

[2] Z. Xiong, O. G. Guleryuz and M. T. Orchard, ``A DCT-Based Embedded Image Coder,'' IEEE Signal Processing Letters, Vol. 3, No. 11, pp. 289-290, Nov. 1996.

[3] D. M. Monro and G. J. Dickson, ``Zerotree Coding of DCT coefficients,'' IEEE Intern. Conf. Image Proc., Vol. 2, pp. 625-628, Oct. 1997.

[4] A. Said and W. A. Pearlman, ``A new, fast, and efficient image codec based on set partitioning in hierarchical trees,'' IEEE Trans. Circ. and Sys. for Vid. Tech., pp. 243-250, June 1996.

[5] Yang Hu, W. A. Pearlman, "Differential-SPIHT for Image Sequence Coding", Proc. IEEE Intl. Conf. Acoustic, Speech, and Signal Processing, p. 894, Dallas, TX, Apr. 2010.

[6] S.-F. Chang and D. G. Messerschmitt, ``A new approach to decoding and compositing motion-compensated DCT-based images,'' Proc. IEEE Intl. Conf. Acoustic, Speech, and Signal Processing, Vol. 5, pp. 421-424, Minneapolis, MN, Apr. 1993.

[7] S.-F. Chang and D. G. Messerschmitt, ``Manipulation and Compositing of MC-DCT Compressed Video,'' IEEE Journal on Selected Areas in Commun., Vol. 13, No. 1, pp. 1-11, Jan. 1995.

[8] Y. Noguchi, D. G. Messerschmitt and S.-F. Chang, ``MPEG Video Compositing in the Compressed Domain,'' Proc. IEEE Intl. Symp. Circuits and Systems, Vol. 2, pp. 596-599, May 1996.

[9] M. Song, A. Cai, J.-a. Sun, ``Motion estimation in DCT domain,'' Proc. IEEE Comm. Technology, ICCT'96, Vol. 2, pp. 670-674, May 1996.

[10] N. Merhav and V. Bhaskaran, ``A Fast Algorithm for DCT-domain Inverse Motion Compensation,'' Proc. IEEE International Conf. Acoustics, Speech, and Signal Processing, Vol. 4, pp. 2307-2310, May 1996.

[11] N. Merhav and V. Bhaskaran, ``Fast Algorithms for DCT-domain Image Downsampling and For Inverse Motion Compensation,'' IEEE Trans. Circuits and Syst. for Video Technology, Vol. 7, No. 3, pp. 468-476, June 1997.

[12] J. Youn and M.-T. Sun, ``A fast motion vector composition method for temporal transcoding,'' Proc. IEEE Intl. Symp. Circuits and Systems, Vol. 4, pp. 243-246, June 1999.

[13] R. Dugad and N. Ahuja, ``A Fast Scheme for Image Size Change in the Compressed Domain,'' IEEE Trans. Circuits and Syst. for Video Technology, pp.461-474, Apr. 2001.

[14] J. Jian and G. Feng, ``The Spatial Relationship of DCT Coefficients Between a Block and its Sub-blocks,'' IEEE Trans. Signal Proc., pp. 1160-1169, May 2002.

[15] H. Park, Y. Park and S.-K. Oh, ``L/M fold image resizing in block-DCT domain using symmetric convolution,'' IEEE Trans. Image Process., vol. 12, pp. 1016-1034, Sept. 2003.

[16] J. Mukherjee and S. K. Mitra, ``Arbitrary resizing of images in DCT space,'' IEE Proc.-Vis. Image Signal Process., vol. 152, pp. 155-164, Apr. 2005.

[17] H. A. Ilgin and L. F. Chaparro, ``Low-Bit Rate Video Coding Using DCT-based Fast Decimation/Interpolation and Embedded Zerotree Coding,'' IEEE Trans. Circuits and Syst. for Video Technology, Vol. 17, No. 7, pp. 833-844, July 2007.

[18] I. H. Witten, R. M. Neal and J. G. Cleary, ``Arithmetic Coding for Data Compression,'' Commun. ACM, Vol. 30, No. 6, pp. 520-540, June 1987.

[19] C.-K. Cheong, K.-S. Cho, and S.-W. Lee, “Signifcance tree image sequence coding with DCT-based pyramid structure,'' in *Proc. IEEE International Conf. Image Proc.*, Vancouver, Canada, Sep. 2000, Vol. 2, pp. 859-862.

[20] A. Ortega and K. Ramchandran, “Rate-distortion methods for image and video compression,'' *IEEE Signal Processing Magazine*, pp. 23-50, Nov. 1998.

Authors: *Assist. Prof. Dr. Hakkı Alparslan ILGIN, Prof. Dr. Hakkı Gökhan İLK, Ankara University, Electronics Eng. Dept., Dögol Caddesi, 06100 Tandogan, Ankara, Turkey, E-mail: ilgin@eng.ankara.edu.tr, ilk@eng.ankara.edu.tr; Assoc. Prof. Dr. Miroslav VOZNAK, Technical University of Ostrava, Department of Telecommunications, 17. listopadu 15/2172, 708 33 Ostrava-Poruba, Czech Republic, E-mail: voznak@ieee.org; Assoc. Prof. Dr. Luis F. Chaparro, University of Pittsburgh, Dept. of Electrical Eng., Pittsburgh, PA 15261, USA, E-mail: chaparro@ee.pitt.edu.*