LOW BIT RATE VIDEO CODING USING DCT-BASED FAST DECIMATION/INTERPOLATION AND EMBEDDED ZEROTREE CODING

H. A. Ilgin and L. F. Chaparro

Department of Electrical and Computer Engineering
University of Pittsburgh
Pittsburgh, PA 15261, USA

ABSTRACT

In this paper, we present a low bit rate video coding procedure in the Discrete Cosine Transform (DCT) domain that is based in fast decimation and interpolation and embedded zerotree coding algorithms. The theory for the decimation/interpolation in the DCT domain is given, and we show how to obtain fast algorithms. Motivated by the compositing of several video sources in multi-point video conferencing, we come up with a new method for single video streams that uses the proposed DCT-decimation in the encoder and the corresponding interpolation in the decoder. The decimated video frames are encoded using an embedded zerotree and an adaptive arithmetic coder to improve the quality of the decoded video and at an exact bit rate. The decimation and interpolation can be done with integer or rational factors, allowing flexibility in choosing the bit rate. Our codec performance is illustrated using different decimation factors, and showing that this codec is particularly efficient for low bit rates.

1. INTRODUCTION

Recently, there has been a growing interest in developing fast algorithms for processing video streams, using H.26x or MPEG compression standards, directly in the DCT domain. It has been shown [3], [4] that the DCT –or compressed domain– is more efficient than the space domain for image resizing. In this paper, we focus on the development of a general approach for DCT decimation and interpolation for integer [4] as well as rational factors and its application, together with embedded zerotree coder [1], [2], to improve the compression of a video stream. This procedure is shown to be very efficient for low bit rate video.

The encoding methodology is shown in Fig. 1. Video frames are encoded in the DCT domain, and after motion estimation and compensation are realized, the DCT error frames are decimated. The decimation and interpolation in the DCT domain basically consists in converting, for an integer N, an $N \times N$ array of 8×8 DCT blocks (typical block size in H.26x and MPEG) into one of size $8N \times 8N$. For efficiency it is possible to use only $q \times q$ ($q < 8$) low frequency coefficients of each block. To insure no aliasing, the $8N \times 8N$ block is then masked to obtain the low-frequency coefficients and thus the decimated DCT block (See Fig. 2). This transformation is slightly more complicated in the case of a rational factor N.

The transformation of an array of DCT blocks into one DCT block was recently introduced [5], and we show a simpler way to obtain it in a matrix form. It is shown to be orthogonal and to have certain properties, derived from the DCT matrices, that permits the development of fast decimation/interpolation algorithms. Using the DCT-decimator in the encoder we are able to encode several frames of a video stream in the size corresponding to one frame. To exploit the dependencies of the DCT coefficients by the zerotree coder, they are arranged into a hierarchical subband structure similar to that in the wavelet transform. This frame is then coded using a DCT-based embedded zerotree (DCT-EZT) algorithm and using an adaptive arithmetic coder. In the decoder, we apply the DCT-interpolation to recover the original DCT frames using the zerotree decoder.

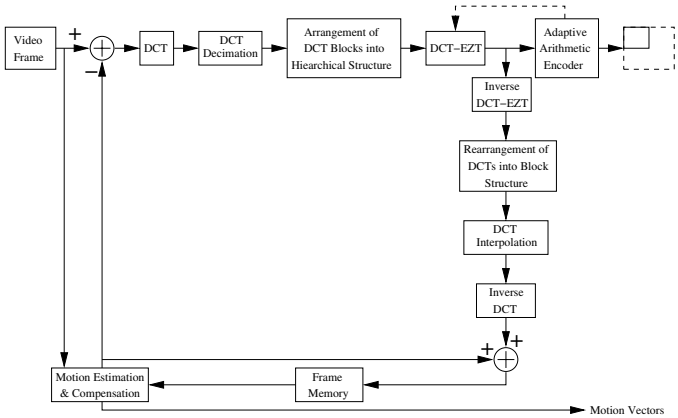

Fig. 1. Proposed encoder

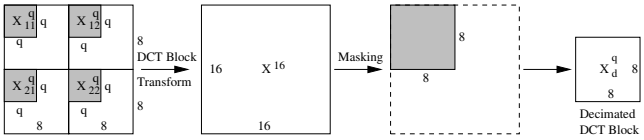

Fig. 2. Proposed DCT decimation

In the next two sections we explain DCT block transformation and a way to make the transformation faster. We also give the details of improved decimation and interpolation in the DCT domain. Simulations and conclusions are given in Section 4 and 5, respectively.

2. DCT BLOCK-TRANSFORMATION

Suppose we have an array of $N \times N$ DCT blocks, N integer, each block of size 8×8, and we wish to transform it into one DCT block of dimension $8N \times 8N$ by means of an orthonormal matrix

transformation. This corresponds to finding the inverse DCT of each of the 8×8 blocks, arranging them into one space array of dimension $8N \times 8N$, and finding its $8N$ DCT. A more efficient way has been proposed in [5], and we will show here a simpler matrix approach. The problem is to find a matrix T^{8N}, that is orthonormal i.e., $(T^{8N})^t T^{8N} = I$, and such that

$$X^{8N} = T^{8N} \begin{bmatrix} X^8_{11} & \cdots & X^8_{1N} \\ & \cdots & \\ X^8_{N1} & \cdots & X^8_{NN} \end{bmatrix} (T^{8N})^t,$$

where $\{X^8_{ij}\}_{i,j=1,\cdots,N}$ are 8×8 DCT blocks and X^{8N} is an $8N$ two-dimensional DCT block. Since the transformation matrix T^{8N} is unique for any set of $\{X^8_{ij}\}_{i,j=1,\cdots,N}$ and X^{8N}, consider the following simple case. Let $X^8_{ii} = I^8$, 8×8 identity matrix, and $X^8_{ij} = \mathbf{0}$, $i \neq j$. The DCT [1] of each X_{ii} is I^8 and the expected X^{8N} is I^{8N}, $8N \times 8N$ identity matrix, therefore

$$I^{8N} = S^{8N} \begin{bmatrix} (S^8)^t S^8 & \cdots & 0 \\ \cdots & (S^8)^t S^8 & \cdots \\ 0 & \cdots & (S^8)^t S^8 \end{bmatrix} (S^{8N})^t,$$

where S^8 is 8×8 DCT matrix, and by its orthonormality $(S^8)^t S^8 = I^8$. Thus, the transformation matrix is

$$T^{8N} = S^{8N} \begin{bmatrix} (S^8)^t & \cdots & 0 \\ \cdots & (S^8)^t & \cdots \\ 0 & \cdots & (S^8)^t \end{bmatrix}, \tag{1}$$

and it is orthonormal as $I^{8N} = T^{8N}(T^{8N})^t$.

To obtain the forward transformation, let

$$S^{8N} = \begin{bmatrix} S^{8N}_1 & S^{8N}_2 & \cdots & S^{8N}_N \end{bmatrix}, \tag{2}$$

so that the transformation matrix becomes

$$T^{8N} = \begin{bmatrix} T^{8N}_1 & T^{8N}_2 & \cdots & T^{8N}_N \end{bmatrix}, \tag{3}$$

and the sub-blocks $\{T^{8N}_i = S^{8N}_i (S^8)^t\}$ are of dimension $8N \times 8$. The representation of X^{8N} in terms of the $\{X^8_{ij}\}$, or the forward transformation, is then

$$X^{8N} = \sum_{i=1}^{N} \sum_{j=1}^{N} T^{8N}_i X^8_{ij} (T^{8N}_j)^t. \tag{4}$$

The inverse transformation of each of the 8×8 DCT blocks is given by

$$X^8_{ij} = (T^{8N}_i)^t X^{8N} T^{8N}_j, \tag{5}$$

using the orthonormality of the transformations.

[1] The $M \times M$ DCT can be expressed in a matrix form by letting $x = \{x(n,m)\}^{M-1}_{n,m=0}$, the DCT matrix $S^M = \{s(k,n)\}^{M-1}_{k,n=0} = \{0.5c(k)\cos((2n+1)k\pi/(2M))\}$ as $X = \{X(k,l)\}^{M-1}_{k,\ell=0} = S^M x (S^M)^t$. The inverse DCT is $x = (S^M)^t X S^M$.

2.1. Fast Transformation

For simplicity consider the integer case $N = 2$. The results can be extended to other integer factors. We show that although in this case $\{T^{16}_i\}_{i=1,2}$ are sparse (half of their entries are zeros and ones) it is possible to obtain sparser matrices for the forward transformation. In fact, as first indicated in [4], the odd rows of $\{S^{16}_i\}_{i=1,2}$ coincide with the odd rows of S^8, and due to the orthonormality of S^8, the matrices $\{T^{16}_i = S^{16}_i (S^8)^t\}_{i=1,2}$ are such that

$$J_0 S^{16}_1 = \begin{bmatrix} S^8 \\ Y \end{bmatrix}$$

where Y is a "don't-care" array, and J_0 is a permutation matrix that separates the odd and the even rows. Thus, the sub-block T^{16}_1 as defined above for $N = 2$, is given by

$$T^{16}_1 = J^t_0 \begin{bmatrix} I^8 \\ Y (S^8)^t \end{bmatrix}$$

showing that half of its entries are zero or one. Similarly for T^{16}_2. Thus the transformation matrices are sparse. Furthermore, it can be shown the following symmetry exists between T^{16}_1 and T^{16}_2:

$$T^{16}_2(i,j) = (-1)^{i+j} T^{16}_1(i,j) \tag{6}$$

for $i = 1, \cdots, 16$ and $j = 1, \cdots, 8$. In fact, by definition

$$\begin{aligned} T^{16}_2 &= S^{16} \begin{bmatrix} 0 \\ I^8 \end{bmatrix} (S^8)^t \\ &= S^{16} J^{16}_i (S^{16})^t S^{16} \begin{bmatrix} I^8 \\ 0 \end{bmatrix} (S^8)^t S^8 (J^8_i)^t (S^8)^t \\ &= J^{16}_m \, T^{16}_1 J^8_m \end{aligned}$$

where J^M_i, $M = 8, 16$, is a permutation matrix,

$$J^M_i = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \end{bmatrix},$$

and

$$J^M_m = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & (-1)^{M+1} \end{bmatrix}$$

is the DCT of J^M_i as can be easily verified. Due to this symmetry, instead of T^{16}_1 and T^{16}_2 we consider their sum and difference,

$$C^{16}_1 = 0.5\,[T^{16}_1 + T^{16}_2], \;\; C^{16}_2 = 0.5\,[T^{16}_1 - T^{16}_2], \tag{7}$$

which according to the above symmetry gives that $C^{16}_1(i,j) = T^{16}_1(i,j)$ when $(i+j)$ are even and zero otherwise. Likewise, $C^{16}_2(i,j) = T^{16}_1(i,j)$ when $(i+j)$ is odd and zero otherwise. Thus,the sparseness and symmetry of the $\{T^{16}_i\}$ matrices makes the $\{C^{16}_i\}$ matrices very sparse, allowing a very efficient forward transformation. Replacing the T^{16}_i in terms of the C^{16}_i matrices in the direct transformation we have

$$\begin{aligned} X^{16} &= [X+Y](C^{16}_1)^t + [X-Y](C^{16}_2)^t \\ X &= C^{16}_1(X^8_{11} + X^8_{21}) + C^{16}_2(X^8_{11} - X^8_{21}) \\ Y &= C^{16}_1(X^8_{12} + X^8_{22}) + C^{16}_2(X^8_{12} - X^8_{22}) \end{aligned} \tag{8}$$

3. DECIMATION AND INTERPOLATION IN THE DCT DOMAIN

Decimation in the space domain consists of low-pass filtering (to avoid aliasing) followed by downsampling, and the bandwidth of the filter depends on the decimation rate. In the DCT domain, masking instead of filtering permits us to obtain the decimated DCT block. Applying the direct transformation we obtain a 16×16 DCT block from the given four 8×8 DCT blocks. Masking the X^{16} matrix to extract the low-frequency coefficients we obtain the decimated DCT array:

$$X_d = [I^8 \;\; 0] X^{16} [I^8 \;\; 0]^t = \sum_{i=1}^{2} \sum_{j=1}^{2} F_i^{16} X_{ij}^8 (F_j^{16})^t \qquad (9)$$

where $F_i^{16} = [I^8 \;\; 0] T_i^{16}$. Again the above equation can be efficiently implemented using the sum and difference representations of T_i^{16} defining

$$D_1^{16} = [I^8 \;\; 0] C_1^{16}, \;\; D_2^{16} = [I^8 \;\; 0] C_2^{16} \qquad (10)$$

The quality of the decimation can be quantified by interpolating the decimated frame to obtain a smooth version of the original frame, and finding the error between this and the original frame. To obtain the smooth-out blocks,$\{\tilde{X}_{ij}^8\}$, consider the forward transformation of the 16×16 block

$$\begin{bmatrix} X_d & 0 \\ 0 & 0 \end{bmatrix} = \sum_{i,j} T_i^{16} \tilde{X}_{ij}^8 (T_j^{16})^t$$

Applying the inverse transformation obtained before we have interpolated blocks:

$$\tilde{X}_{ij}^8 = (T_i^{16})^t \begin{bmatrix} X_d & 0 \\ 0 & 0 \end{bmatrix} T_j^{16}, \;\; i,j = 1,2 \qquad (11)$$

where T_j^{16}, as explained before, are 16×8 matrices.

3.1. Improved Decimation

Typically, the high frequency coefficients in a DCT block are zero, and even when they are set to zero its inverse DCT values are not very different from the original ones. Consider then that the DCT blocks to be decimated have $q \times q$ ($1 \le q \le 8$) low-frequency components and the rest are zero

$$X_{ij}^8 \;\; = \;\; \begin{bmatrix} X_{ij}^q & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} I^q \\ 0 \end{bmatrix} X_{ij}^q \begin{bmatrix} I^q & 0 \end{bmatrix}$$

Replacing these blocks in (9) gives

$$X_d^q = \sum_{i,j} G_i^{16} X_{ij}^q (G_j^{16})^t \qquad (12)$$

where $G_i^{16} = F_i^{16} [I^q \;\; 0]^t$, $i = 1,2$, $1 \le q \le 8$. Again, these $8 \times q$ matrices are sparse, but as before, sparser matrices can be used:

$$E_1^{16} = D_1^{16} [I^q \;\; 0]^t, \;\; E_2^{16} = D_2^{16} [I^q \;\; 0]^t$$

These matrices are sparser than the C and the D matrices (especially for small values of q). For example, for $N = 2$ and $q = 4$, 69% of the entries of each E matrix are zeros. For this case the number of computations to decimate an image is 1.25 multiplications/pixel and 1.25 additions/pixel which is the same as Dudag's in [4]. As expected, the larger q is (i.e., $4 \le q \le 8$), the better the interpolation (see Table 1), but the more complex the implementation. Decimation by an integer $N > 2$ proceeds in a very similar way to decimation by 2. In each case, we transform an array of $N \times N$ DCTs of size 8×8 into one array of size $8N \times 8N$, and then mask it to obtain the decimated 8×8 DCT. Just as before, it is also possible when $N > 2$ to represent each 8×8 block with a $q \times q$ coefficients ($q < 8$) and the rest zero for faster implementation. The computational complexity depends on the value of q, and the error between the original block and the interpolated block, as measured by PSNR, is better for larger values of q. It is also possible to decimate by a rational number, e.g., $N = 2/3$ or $N = 3/4$. In this case the algorithm requires additional computations. When $N = 2/3$, we first need to transform 3×3 array of 8×8 DCT blocks into one of size 24×24. The masking to get the $2/3$ gives us a DCT block of dimension 16×16 which needs then to be converted into four 8×8 blocks. These additional computation increases the complexity of the decimation for rational factors.

Table 1. PSNR (dB) results of the proposed method and Dugad's ($N = 2$)

Sample Frame	Dugad et. al	$q = 4$	$q = 8$
Miss America	38.97	38.94	39.43
Salesman	30.51	30.46	30.96
Foreman	32.73	32.68	33.18
Hall	28.52	28.47	29.04
News	29.73	29.66	30.37

4. SIMULATIONS

We illustrate our decimation/interpolation algorithm with a Salesman CIF (Common Intermediate Format) video sequence. In the figures, regular encoding stands for H.263 video coding. We replace regular quantization with embedded zerotree coding in H.263, consequently adaptive arithmetic coding is used. This way we use the same encoding methodology to encode DCT coefficients and data symbols with the proposed system. We also use full pixel motion estimation and compensation for simplicity, although half pixel motion estimation and compensation can be used to improve quality.

Considering only intra-frames, the effect of the decimation factor N in the reconstructed image quality for given bit rates is shown in Fig. 3. We also use only $q \times q$, $q < 8$ part of DCT blocks for each decimation case as stated particularly in the same figure. The effects of each decimation factor differ with the number of bits used to encode each frame. For lower bit rates, we obtain better PSNRs than H.263 for reconstructed intraframes as shown in Fig. 3. Notice that for higher decimation factors ($N = 4$) the PSNR is initially better than for lower factors, but as the bit rate increases the opposite is true. Also from Fig. 3 the PSNR saturations occur after a particular bit rate. Thus the maximum required bit rate for a particular decimation factor can be taken as the bit rate corresponding to where the PSNR graphs coincide.

Considering both intra- and inter-frames, average PSNR comparisons of the proposed method with decimation factor $N = 2$ and regular encoding are shown in Fig. 4. As seen from this figure, average PSNRs of a video sequence strongly depends on the

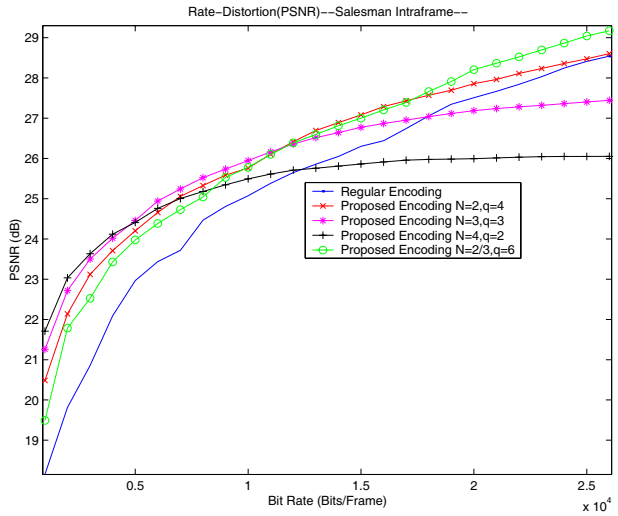

Fig. 3. Rate-distortion performances of the proposed encoding vs. regular encoding for Salesman intraframe

bit rate of the intra-frame. If an intra-frame is encoded with a low bit rate, which is our aim in this work, the proposed method gives better results then regular encoding. In Fig. 4, from top to bottom, intra-frames are encoded with 10000, 15000, and 20000 bits. We encode 30 frames/second and each GOP (Group of Picture) includes an intra-frame and 49 inter-frames. On the graphs on the top and in the middle, our method gives better PSNRs from 7.47 kilobits/second up to 45 kilobits/second. However, efficiency of the proposed method at higher bit rates decreases as displayed on the graphs in Fig. 4. Beside objective results, the reconstructed frames are also subjectively better than the ones from regular coding at lower bit rates.

5. CONCLUSION

In this paper we introduce fast decimation and interpolation for low bit rate video coding. Depending on the desired bit rate any integer or rational decimation factor can be chosen to encode video sequences. In the receiver side corresponding interpolation is realized. Reconstructed video quality is improved regarding to regular encoding. However as the bit rate increases, efficiency of the decimation and interpolation of the proposed coding decreases. Our method is well suitable for low bit rate video applications such as videoconferencing. The decimation/interpolation method we propose is not complex and is flexible, so the delay introduced by the decimation is negotiable.

6. REFERENCES

[1] Shapiro, J. M., "Embedded Image Coding Using Zerotrees of Wavelet Coefficients," *IEEE Trans. Sig. Proc.*, pp. 3445-3462, Dec. 1993.

[2] A. Said, 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.

[3] Y. Noguchi, D. G. Messerschmitt, and S.-F. Chang, "MPEG Video Compositing in the Compressed Domain," *Proc. IEEE Intl. Symp. Circ. and Sys.*, Vol. 2, pp. 596-599, May 1996.

[4] Dugad, R., Ahuja, N., "A Fast Scheme for Image Size Change in the Compressed Domain," *IEEE Trans. Circ. and Sys. for Vid. Tech.*, pp.461-474, Apr. 2001.

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

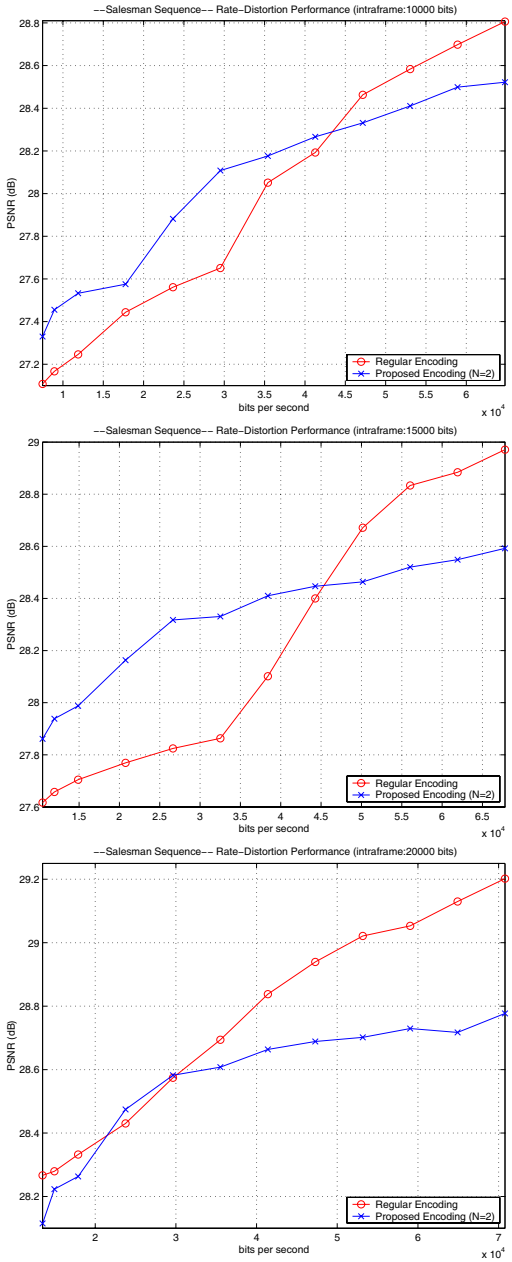

Fig. 4. Rate-distortion performances of the proposed encoding vs. regular encoding for video sequences