From 8-Tap DCT to 4-Tap Integer-Transform for MPEG to H.264/AVC Transcoding

Bo Shen
Hewlett-Packard Laboratories
1501 Page Mill Road, 1181, Palo Alto, CA 94304
Email: boshen@hpl.hp.com

***Abstract*— H.264 uses a 4-tap integer transform (*DCT-like*) to produce transform coefficients. This is different than the 8-tap DCT that is used in most prior video coding standards such as MPEG-x. The paper propose two integer algorithms for transcoding from 8-tap DCT data to 4-tap DCT-like data. The approximate nature of the integer transcoding algorithms is quantitively evaluated. The results show that the approximation is hidden in most cases when bit rate reduction is considered.**

I. Introduction

One of the most important aspects of a transcoding algorithm is its capability of reusing already encoded information resides in the input object. Given that H.264 [1] is becoming the preferred video codec for video compression, there is a need to investigate on transcoding from MPEG-like objects (e.g., MPEG-1/2/4, H.263). Some effort has been witnessed (e.g., [3]) in reusing the motion information and exploring the more flexible motion compensation schemes offered by H.264. This paper investigates the possibility of reusing the transform data, specifically, DCT residual information in bit rate reduction and spatial-resolution reduction transcoding. The problem is unique since H.264 uses a 4-tap *DCT-like* integer transformation [2] to produce transform coefficients, while most MPEG-like video codecs use 8-tap DCT. The question is whether there is a transcoding approach to bridge the two with smaller computation load comparing to a re-encode approach, i.e., performing the inverse 8-tap DCT followed by the 4-tap integer transformation. This paper answers this question by proposing two integer transcoding solutions.

In Section II, two integer transcoding algorithms are proposed. In Section III, the approximate nature of these algorithms is investigated considering along with the bit rate reduction operation. The issues related to using these algorithms in MPEG-like to H.264 transcoding are also discussed. The concluding remarks are provided in Section IV.

II. Integer Transcoding Algorithm

Two algorithms are proposed with the first one handling the case of transcoding from one 8-tap DCT block to four 4-tap DCT-like blocks. This can be used to transcode MPEG-like video objects into H.264 format while reducing the output bit rate. The second algorithm handles the case of transcoding from one 8-tap DCT block to one 4-tap DCT-like block. It can be used in spatial resolution reduction along with bit rate reduction. Since H.264 uses spatial prediction prior to DCT-like transformation of intra macroblocks, which is not used in any MPEG-like formats, only transcoding of inter macroblocks is considered here. Note that the motion information from the input video can be completely reused in both cases, which indicates that the DCT residue can also be reused if we can find good solutions to transcode from 8-tap DCT to 4-tap DCT-like blocks. The data block we consider hereafter are all residual blocks in inter frames.

A. One 8-Tap DCT to Four 4-Tap DCT-like

Consider an 8×8 DCT residual block B from an input video object, it is reconstructed using the 8-tap inverse DCT:

$$\hat{b} = T_8' B T_8, \tag{1}$$

where T_8 is the 8-tap DCT matrix. Block $\hat{b}$ consists of four 4×4 blocks $\hat{b}_{11}$, $\hat{b}_{12}$, $\hat{b}_{21}$ and $\hat{b}_{22}$ in the order of left to right and up to bottom. Each 4×4 block can be derived from the 8×8 block through a pair of matrix multiplications:

$$\hat{b}_{ij} = e_i \hat{b} e_j', \tag{2}$$

where e_1 and e_2 are 4×8 matrices that are defined as the upper and lower half of an 8×8 identity matrix, respectively.

To produce 4×4 transform blocks for H.264, the 4-tap integer transformation (DCT-like) is applied:

$$\hat{B}_{ij} = T_4 \hat{b}_{ij} T_4', \tag{3}$$

where T_4 is the 4-tap integer transformation as defined in H.264 [1].

Plugging Eq. (1) into Eq. (2) and then in Eq. (3), we have:

$$\hat{B}_{ij} = T_4 e_i T_8' B T_8 e_j' T_4'. \tag{4}$$

Denoting

$$E_i = T_4 e_i T_8', \tag{5}$$

we have

$$\hat{B}_{ij} = E_i B E_j'. \tag{6}$$

After some manipulations, $\hat{B}_{ij}$ can be computed as follows:

$$\hat{B}_{11} = (W + X + Y + Z)/4 \tag{7a}$$
$$\hat{B}_{12} = (W + X - Y - Z)/4 \tag{7b}$$
$$\hat{B}_{21} = (W - X + Y - Z)/4 \tag{7c}$$
$$\hat{B}_{22} = (W - X - Y + Z)/4, \tag{7d}$$

0-7803-8554-3/04/$20.00 ©2004 IEEE.

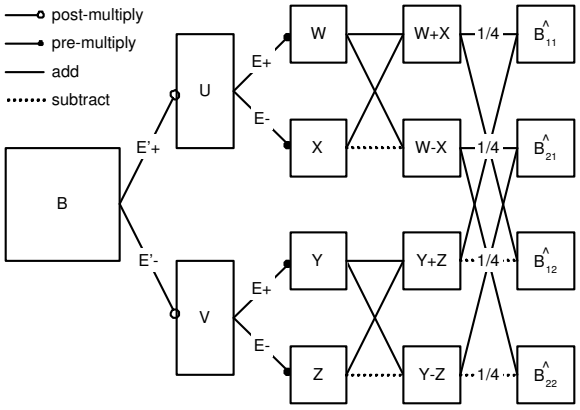

Fig. 1. Processing flow for one 8-tap to four 4-tap blocks

where

$$W = E_+ B E'_+ \tag{8a}$$
$$X = E_- B E'_+ \tag{8b}$$
$$Y = E_+ B E'_- \tag{8c}$$
$$Z = E_- B E'_-, \tag{8d}$$

and

$$E_+ = E_1 + E_2 \tag{9a}$$
$$E_- = E_1 - E_2. \tag{9b}$$

Eqs. (8) can be efficiently computed by denoting

$$U = B E'_+ \tag{10a}$$
$$V = B E'_-, \tag{10b}$$

and then:

$$W = E_+ U \tag{11a}$$
$$X = E_- U \tag{11b}$$
$$Y = E_+ V \tag{11c}$$
$$Z = E_- V. \tag{11d}$$

Fig. 1 depicts the block diagram of the computing process.

Now let us look into the complexities of the post- and pre-matrix multiplications shown in Fig. 1. Based on Eq. (5) and (9), E_+ and E_- are computed as:

$$E_+ = \begin{pmatrix} \sqrt{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5842 & 0 & 1.1257 & 0 & -0.5463 & 0 & 0.3051 \\ 0 & 0 & 0 & 0 & \sqrt{2} & 0 & 0 & 0 \\ 0 & 0.0740 & 0 & -0.0583 & 0 & 0.6564 & 0 & 1.2491 \end{pmatrix} \tag{12}$$

$$E_- = \begin{pmatrix} 0 & 1.2815 & 0 & -0.4500 & 0 & 0.3007 & 0 & -0.2549 \\ 0 & 0 & 1.4107 & 0 & 0 & 0 & -0.1003 & 0 \\ 0 & -0.1056 & 0 & 0.7259 & 0 & 1.0864 & 0 & -0.5308 \\ 0 & 0 & 0.1003 & 0 & 0 & 0 & 1.4107 & 0 \end{pmatrix}. \tag{13}$$

Due to the manipulations from Eq. 7 to Eq. 11 taking advantages of the symmetric property of the transformations, many entries in E_+ and E_- are zero, which reduces the transcoding complexity significantly. However, each matrix still contains at least 10 non-trivial elements, which causes one matrix multiplication (8×4 with 4×4) to have at least 40 multiplications. To reduce this complexity, we use a similar approach as introduced in [4] which is based on a factorization of the DCT transformation matrix. The factorization is considered along with the uniquely defined 4-tap DCT-like transformation used in H.264. Specifically, the 8-tap DCT matrix, T_8 can be factorized as follows [5]:

$$T_8 = D_8 P B_1 B_2 M A_1 A_2 A_3. \tag{14}$$

Please refer to [5] for definition of these matrices.

On the other hand, the 4-tap DCT-like transformation used in H.264, T_4 is also factorized into a diagonal matrix and an integer transformation matrix.

$$T_4 = D_4 C = \begin{pmatrix} a & & & \\ & b & & \\ & & a & \\ & & & b \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & d & -d & -1 \\ 1 & -1 & -1 & 1 \\ d & -1 & -1 & -d \end{pmatrix}, \tag{15}$$

where $a = 1/2, b = \sqrt{2/5}$ and $d = 1/2$. It represents an integer orthogonal approximation to the 4-tap DCT [2].

Plugging Eq. (14) and (15) into Eq.(5) and then Eq. (9), we now have the factorized version of E_+ and E_-:

$$E_+ = D_4 \overbrace{C(e_1 + e_2) A'_3 A'_2 A'_1 M'} B'_2 B'_1 P D_8 \tag{16a}$$
$$E_- = D_4 \underbrace{C(e_1 - e_2) A'_3 A'_2 A'_1 M'} B'_2 B'_1 P D_8. \tag{16b}$$

From this sequence of matrix multiplications, we find the products of the matrices within the under and over braces render sparse matrices [1]. Therefore, denoting:

$$E^d_+ = C(e_1 + e_2) A'_3 A'_2 A'_1 M' \tag{17a}$$
$$E^d_- = C(e_1 + e_2) A'_3 A'_2 A'_1 M', \tag{17b}$$

we have

$$E_+ = D_4 E^d_+ B'_2 B'_1 P D_8 \tag{18a}$$
$$E_- = D_4 E^d_- B'_2 B'_1 P D_8. \tag{18b}$$

The matrix multiplication with E_+ or E_- can now be carried out with D_4 absorbed in the quantization as already defined in H.264 [1], and D_8 absorbed in the inverse quantization. The matrix multiplications with P, B'_2 and B'_1 contain only trivial operations. Therefore, only E^d_+ and E^d_- contains non-trivial elements as shown in the following.

$$E^d_+ = \begin{pmatrix} 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -3.9197 & 0 & 1.6236 & 2 \\ 0 & 8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.3066 & 0 & -0.5412 & 1 \end{pmatrix} \tag{19a}$$

$$E^d_- = \begin{pmatrix} 0 & 0 & 0 & 0 & 2.1648 & 2\sqrt{2} & 5.2263 & 2 \\ 0 & 0 & 4.2426 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -2\sqrt{2} & 0 & 2 \\ 0 & 0 & -\sqrt{2} & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{19b}$$

Note that there are 8 non-zero coefficients in E^d_+, among them 5 are non-trivial, and 10 non-zero coefficients in E^d_-, among them 6 are non-trivial.

[1] Other possible combinations have been compared, none renders sparser ones.

We can further reduce the computation complexity by using basic integer operation (shift and/or add) to replace multiplications. Specifically, we can represent fractional numbers using 8-bit 2's complement format. Allowing at most 1 shift to replace a floating-point multiplication (1s approximiation), E_+^d and E_-^d can be approximated as:

$$E_+^d = \begin{pmatrix} 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -4 & 0 & 2 & 2 \\ 0 & 8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & -0.5 & 1 \end{pmatrix} \quad (20a)$$

$$E_-^d = \begin{pmatrix} 0 & 0 & 0 & 0 & 2 & 2 & 4 & 2 \\ 0 & 0 & 4 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -2 & 0 & 2 \\ 0 & 0 & -1 & 2 & 0 & 0 & 0 & 0 \end{pmatrix}. \quad (20b)$$

Alternatively, we can allow at most 2 shifts and 2 additions for each approximation to achieve a better precision (2s2a).

$$E_+^d = \begin{pmatrix} 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -4 & 0 & 1.625 & 2 \\ 0 & 8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.25 & 0 & -0.5 & 1 \end{pmatrix} \quad (21a)$$

$$E_-^d = \begin{pmatrix} 0 & 0 & 0 & 0 & 2.125 & 2.5 & 5.25 & 2 \\ 0 & 0 & 4.25 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -2.5 & 0 & 2 \\ 0 & 0 & -1.5 & 2 & 0 & 0 & 0 & 0 \end{pmatrix}. \quad (21b)$$

Using Eq. (20) or (21) for the matrix multiplications with E_+ and E_- as shown in Fig. 1 offers an integer transcoding solution which is much more efficient than the re-encoding approach. This integer transcoding produces approximated results, we will quantitively evaluate the quality degradation shortly.

Note that one can argue that the 8-tap DCT can also be approximated using integer operations only; therefore, combined with the 4-tap integer transformation defined by H.264, the re-encoding approach can also be carried out with only integer operations. However, this integer re-encoding approach produces more quality degradation since the approximation is introduced in the inverse transform stage without consideration of what follows in the rest of the re-encoding processes.

B. One 8-Tap DCT to One 4-Tap DCT-like

To produce 4-tap DCT data from 8-tap DCT data, one simple approach is to reuse the low-pass band of the 8-tap DCT data with a scale factor [7]. The inverse DCT of the 4×4 low-pass coefficients truncated from an 8×8 block provides a low-pass filtered version. However, H.264 uses a 4-tap DCT-like integer transformation, the truncating approach may not generate precise transcoding results. Without delving into the detail, it is shown that the following adjustment needs to be applied on the 4-tap low-pass band (B_4) that is truncated from the 8-tap DCT input (B)

$$\hat{B_4} = A' B_4 A, \quad (22)$$

where

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.9975 & 0 & 0.0709 \\ 0 & 0 & 1 & 0 \\ 0 & -0.0709 & 0 & 0.9975 \end{pmatrix}. \quad (23)$$

$\hat{B_4}$ can then be used as a H.264 transform block. Note that A is *almost* an identity matrix, which also indicates that the H.264 integer transformation is very similar to DCT.

In one particular case, the adjustment is not necessary if B_4 contains all zero coefficients in the 2nd and 4th row and column. In general, to avoid non-trivial multiplications, we can approximate A as an identity matrix. Therefore, B_4 can be approximately used as an H.264 transform block without the adjustment. The approximation results in an integer transcoding with some degradation in quality. Following a possibly coarser quantization for bit rate reduction, the degradation caused by this approximation may not be significant.

C. Discussion on MPEG to H.264 transcoding

Since H.264 uses inter-block pixel prediction for intra frames and all MPEG-like codecs do not, a direct 8-tap to 4-tap DCT-like method can not be used for intra block transcoding. Given that we want to avoid going back to the pixel domain, the adaptive decision on prediction mode based on pixel domain information is difficult to achieve. The transcoder can assume a suboptimal approach in which only DC prediction mode is used to facilitate this transcoding. To some extend, this may not introduce too much degradation given that the only information available to the transcoder is the compressed video which is already lossy compressed. Similar approach introduced in Section II-A can be used to derive an integer transcoding method for transcoding of intra blocks in DC prediction mode.

For inter frames, MPEG-like codecs encode all residue in 8×8 blocks. On the other hand, H.264 also supports 8×8 mode using a 8-tap integer transformation. When spatial resolution reduction is not required, it is ideal that the MPEG residual blocks, and therefore the motion information, can be reused. However, the 8-tap DCT-like transformation defined in H.264 is a scaled and integerized approximation to the 8-tap DCT. To reuse the MPEG DCT blocks, the same approach as outlined in Section II-B can be followed to find out the difference between 8-tap DCT blocks and 8-tap DCT-like blocks. Again, quantitive tests should be carried out to justify any approximation.

III. Performance Evaluation

Considering bit rate reduction, the approximation introduced by these integer transcoding methods can be hidden due to the fact that a coarser quantization stage always follows the 8-tap to 4-tap transcoding. We use a group of CIF resolution MPEG-1 sequences coded with three fixed QPs as input. All blocks in the inter frames (12-frame GOP) are forced to be coded in the predictive mode. The integer transcoding is applied on DCT residual blocks. The results are then subjected to requantization with QP factors ranging from 1 to 31.

We compute the mean square error (MSE) of the inverse integer transformed (as defined in H.264) non-zero residual blocks that are produced using integer transcoding methods and the MSE of those that are produced using the re-encoding method. Both MSEs are computed with reference to the

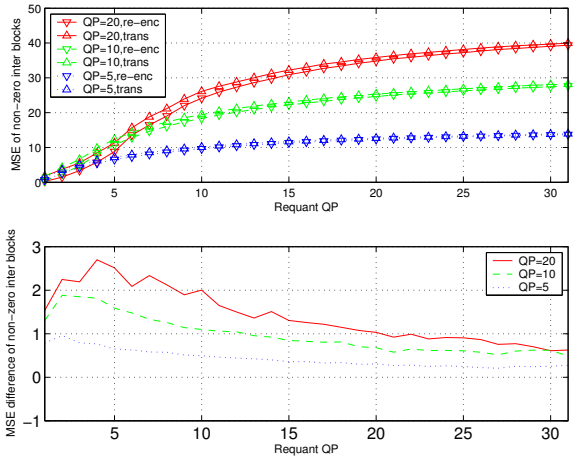

Fig. 2. One 8-tap to four 4-tap integer transcoding with 1s approximation

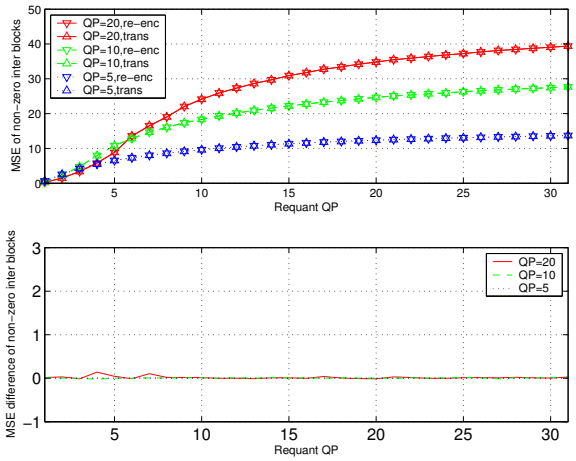

Fig. 3. One 8-tap to four 4-tap integer transcoding with 2s2a approximation

inverse DCTed input. In the one 8-tap to one 4-tap case, the reference is further bilinear down sampled by a factor of two. The results of the table-tennis sequence are shown here. Fig. 2 shows the results for the one 8-tap to four 4-tap integer transcoding with 1s approximation, and Fig. 3 with 2s2a approximation.

For 1s approximation, the MSE differences between re-encoded and transcoded results generally decrease when the requantization QP is increasing. In the case of QP=5 and requant QP=15, the MSE difference is about 0.4. Considering that the actual MSE of these non-zero blocks is about 12 as shown in the top figure of Fig. 2, the MSE difference corresponds to a PSNR degradation of about 0.14 dB. Note that the MSEs are collected over all non-zero inter blocks. If considering with zero inter blocks that cause no degradation in transcoding, the overall frame degradation is even smaller. On the other hand, it should be point out that these experiments do not consider drift accumulation over the sequence, and the handling of which is outside the scope of this paper. For 2s2a approximation, the MSE differences are very small and the transcoding produces almost identical results as re-encoding.

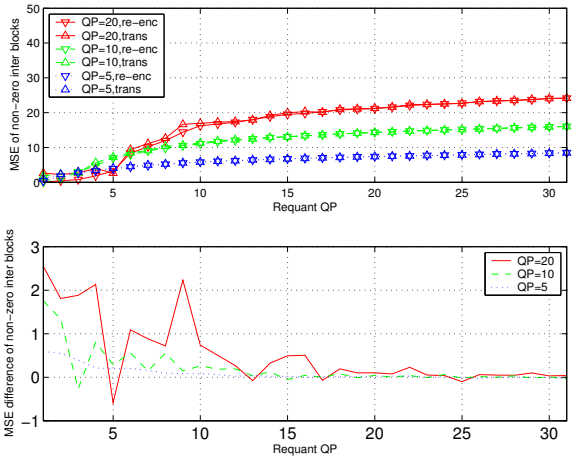

Fig. 4. One 8-tap to one 4-tap integer transcoding

Fig. 4 shows the results for the one 8-tap to one 4-tap case, which also corresponds to a spatial resolution reduction by a factor of two in each dimension. The general trend of decreasing MSE difference with increasing requantization QP is also observed. For all input QP factors, the MSE difference is smaller than 0.3 when the requantization QP is bigger than its corresponding input QP. The occasional zero-crossing of the MSE difference curves is due to the fact that the bilinear-downsampled version of the inverse DCTed input blocks is used as the reference, which is not mathematically equivalent to the inverse transformed version of the truncated-and-scaled blocks.

IV. Conclusion

Two integer algorithms for transcoding of 8-tap DCT data to 4-tap DCT-like data are proposed. The algorithms can be used to efficiently transcode residual blocks in MPEG-like video objects to H.264 format while reusing motion information. The transcoding only requires basic integer operations such as shift and add. We have shown that the error produced by the integer transcoding is mostly insignificant, especially when bit rate reduction is also considered as a goal of the transcoding.

References

[1] Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, *Joint Final Committee Draft (JFCD) of Joint Video Specification (ITU-T Rec. H.264 — ISO/IEC 14496-10 AVC)*, Nov. 2002.

[2] H. Malvar, A. Hallapuro, M.Karczewicz and L. Kerofsky, "Low-complexity transform and quantization with 16bit arithmetic for H.264," *Proc. of ICIP 2002*, Rochester, NY, Sept. 2002.

[3] H. Sun and Y.P. Tan, "Arbitrary downsizing video transcoding using H.264 standard," *Proc. of ICIP 2003*, Barcelona, Spain, Sept. 2003.

[4] N. Merhav and V. Bhaskaran, "A Fast Algorithm for DCT-Domain Image Downsampling", *Proc. ICASSP'96*, vol. II, pp. 2307-2310, Atlanta, May 1996.

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

[6] Y. Arai, T. Agui and M. Nakajima, "A Fast DCT-SQ Scheme for Images", *Trans. of IEICE*, vol. E71, no. 11, pp. 1095-1097, Nov. 1988.

[7] K. N. Ngan, "Experiments on two-dimensional decimation in time and orthogonal transform domains," *Signal Processing*, vol. 11, pp.249-263, 1986.