AN NEW COEFFICIENTS TRANSFORM MATRIX FOR THE TRANSFORM DOMAIN MPEG-2 TO H.264/AVC TRANSCODING

Gao Chen[1,2], *Shouxun Lin*[1], *Yongdong Zhang*[1], *Gang Cao*[1]

[1] Institute of Computing Technology, Chinese Academy of Sciences Beijing, 100080, China
[2] Graduate School of the Chinese Academy of Sciences, Beijing, 100039, China
Email: {chengao, sxlin, zhyd, gcao}@ict.ac.cn

ABSTRACT

In this paper, a fast transform method is proposed to convert MPEG-2 8-tap discrete cosine transform (DCT) coefficients to H.264/AVC 4-tap integer transform coefficients directly in the transform domain. The proposed transform method saves 16 operations for each 8×8 DCT block by utilizing a novel transform kernel matrix and a fast computing method for multiplication of this new matrix. The simulation results show that the proposed method causes only a very little quality degradation, which is completely negligible in practice with the maximum value lower than 8×10^{-3}dB, as compared with Jun Xin's method. Hence, it can be efficiently used in the transform-domain MPEG-2 to H.264 transcoding.

1. INTRODUCTION

The newest video-coding standard, known as H.264/AVC [1], jointly developed by the Joint Video Team of ISO/IEC MPEG and ITU-T VCEG, is highly efficient offering perceptually equivalent quality video at about 1/3 to 1/2 of the bit-rates offered by the MPEG-2 format [2]. Due to its superior compression efficiency, it is expected to replace MPEG-2 in digital video systems, but the complete migration to H.264 will take several years given the fact that MPEG-2 has been widely used in many applications nowadays, including DVD and digital TV. This creates an important need for transcoding technologies that transcode the widely available MPEG-2 compressed videos to H.264 compressed format and vice versa [3][4]

The transform domain transcoder is simpler than the one in the conventional pixel domain, since the former avoids the complete decoding and re-encoding which are computationally expensive. For this reason, there has been a great effort in recent time to develop fast algorithms that conduct MPEG-2 to H.264 transcoding in the transform domain [5][6]. Unlike other transform domain transcoding, such as H.263 to MPEG-2 transcoding, the DCT coefficients in MPEG-2 to H.264 transcoding cannot be reused directly and have to be converted to H.264 integer transform coefficients. This is because that H.264 and MPEG-2 are based on different transformation kernels to produce transform coefficients, that is H.264 uses a 4-tap integer transformation (we refer as IT thereinafter), while MPEG-2 uses 8-tap DCT to produce transform coefficients. Thus, one of the indispensable steps in the transform domain MPEG-2 to H.264 transcoding is to convert DCT coefficients to IT coefficients, i.e. DCT-to-IT transform. In the conventional pixel domain MPEG-2 to H.264 transcoder, there also exits the problem of DCT-to-IT transform. That is the de-quantized MPEG-2 DCT coefficients are first converted to pixels data through inverse DCT and are then transformed to IT coefficients through IT. This process is also called the pixel domain method [7][8]. Obviously, this method cannot be adopted in the transform domain transcoder since that the incoming MPEG-2 video sequence is already decoded to the pixel data.

The role of DCT-to-IT transform in MPEG-2 to H.264 transcoder is equal to the transform, such as DCT, in one of video encoder. There are many fast DCT algorithms have been proposed to implement the video encoder efficiently [9]. In order to implement the MPEG-2 to H.264 transcoder efficiently in the transform domain, we should try our best to speed up the process of DCT-to-IT transform. Jun Xin et al., [7] and Bo Shen [8] have derived two different transform kernel matrices for DCT-to-IT transform, and have showed their methods outperform the pixel domain method respectively. Although Jun Xin's method needs 64 less operations compared with Bo Shen's method, there still exits 19 nonzero elements in his transform kernel matrix, which cause DCT-to-IT transform to need 352 multiplications and 352 additions (for total of 704 operations). In this paper, we propose a novel transform kernel matrix based on the factorization of DCT. The new transform matrix saves 16 operations for each 8×8 DCT block compared with the Jun Xin's matrix.

The rest of the paper is organized as follows. We derive a novel transform kernel matrix based on the factorization of DCT in Section 2. Furthermore, we also propose a fast computing process for matrix multiplication using the new transform kernel matrix in Section 3. Simulation results are given in Section 4, and conclusions are provided in Section 5.

2. NOVEL TRANSFORM KERNEL MATRIX

Because Jun Xin's DCT-to-IT transform kernel matrix is more efficient than the one of Bo Shen, we adopt Jun Xin's matrix as our start point. At the same time, the factorized form of DCT matrix adopted by Bo Shen in [8] is also exploited to derive our new transform kernel matrix. The detailed process is presented in the following.

Let T_8 be the transform kernel matrix of DCT, H be the IT transform matrix and K represent the matrix: $K = \begin{pmatrix} H & 0 \\ 0 & H \end{pmatrix}$. The Jun Xin's DCT-to-IT transform kernel matrix S is given by

$$S = \begin{pmatrix} H & 0 \\ 0 & H \end{pmatrix} \times T_8{}^T = K \times T_8{}^T \qquad (1)$$

Where the superscript T denotes matrix transposition. For the proof and more details of the S, please refer to [7].

A factorization of DCT that is the fastest existing algorithm for 8-point DCT due to Arai, Agui, and Nakajima [9][10] is exploited to perform the factorization of the transform kernel matrix S. According to this factorization, T_8 is represented as $T_8 = DPB_1B_2MA_1A_2A_3$, where the matrices on the right-hand side are defined as in [9, pp. 53-57]. Thus, we can rewrite equation (1) into

$$S = K \times T_8{}^T = \overbrace{K \times A_3{}^T A_2{}^T A_1{}^T M^T} B_2{}^T B_1{}^T P^T D^T \qquad (2)$$

Where D is a diagonal matrix, and P is a permutation matrix. We observe that $D = D^T$ and $P = P^T$. So

$$S = \overbrace{K \times A_3{}^T A_2{}^T A_1{}^T M^T} \times (B_1B_2)^T \times P \times D \qquad (3)$$

After calculating and comparing all possible combinations of this sequence of matrix multiplications, we find that the product of the matrices within the over braces renders the sparest matrices. Then we define: $S^d = K \times A_3{}^T A_2{}^T A_1{}^T M^T$, and have

$$S = S^d \times (B_1B_2)^T \times P \times D \qquad (4)$$

DCT-to-IT transform now can be carried out multiplication by $(B_1B_2)^T$ and S^d in turn. The multiplication of matrix D can be ignored since it can be absorbed in MPEG-2 inverse quantization matrix without any changes in arithmetic complexity of the de-quantizer. The matrix P causes only changes in the order of the components, thus it can be ignored as well. The matrix (B_1B_2) only contains 0, 1 and –1. The matrix S^d can be used as the novel transform kernel matrix to perform DCT-to-IT transform. The matrix (B_1B_2) and S^d are shown in the following.

$$B_1B_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 \\ 0 & 0 & 0 & 0 & -1 & -1 & 0 & 1 \end{pmatrix} \qquad (5)$$

$$S^d = \begin{pmatrix} 4 & 0 & 0 & 0 & a & b & c & 1 \\ 0 & 0 & d & 4 & -e & 0 & f & 2 \\ 0 & 4 & 0 & 0 & 0 & -b & 0 & 1 \\ 0 & 0 & -b & 2 & g & 0 & -h & 1 \\ 4 & 0 & 0 & 0 & -a & -b & -c & -1 \\ 0 & 0 & -d & -4 & -e & 0 & f & 2 \\ 0 & 4 & 0 & 0 & 0 & b & 0 & -1 \\ 0 & 0 & b & -2 & g & 0 & -h & 1 \end{pmatrix} \qquad (6)$$

Where the const values a … h are (rounded off to four decimal places): a= 1.0824, b= 1.4142, c=2.6132, d= 4.2426, e= 3.9198, f= 1.6236, g= 1.3066, h= 0.5412.

3. FAST COMPUTING METHOD FOR MULTIPLICATION OF S^d

The sparseness and symmetry of S^d can be exploited to perform the multiplication of S^d. Let z be an 8-dimensional column vector, and the vector Z be the one-dimensional (1D) DCT-to-IT transform result of z. The following steps describe the fast computing method for multiply B_1B_2 and S^d to z, which is also depicted in Fig. 1 as a flow graph.

First step, multiplication by $(B_1B_2)^T$:

$x[1] = z[1]$
$x[2] = z[2]$
$x[3] = z[3] - z[4]$
$x[4] = z[3] + z[4]$
$x[5] = z[5] - z[8]$
$x[6] = z[6] + z[7] - z[5] - z[8]$
$x[7] = z[6] - z[7]$
$x[8] = z[5] + z[6] + z[7] + z[8]$

Second step, multiplication by S^d:

$m_1 = 4 \times x[1]$
$m_2 = a \times x[5] + b \times x[6] + c \times x[7] + x[8]$
$m_3 = f \times x[7] + x[8] + x[8] - e \times x[5]$
$m_4 = d \times x[3] + 4 \times x[4]$
$m_5 = 4 \times x[2]$
$m_6 = x[8] - b \times x[6]$

$m_7 = g \times x[5] - h \times x[7] + x[8]$
$m_8 = b \times x[3] - x[4] - x[4]$

$Z[1] = m_1 + m_2$
$Z[2] = m_3 + m_4$
$Z[3] = m_5 + m_6$
$Z[4] = m_7 - m_8$
$Z[5] = m_1 - m_2$
$Z[6] = m_3 - m_4$
$Z[7] = m_5 - m_6$
$Z[8] = m_7 + m_8$

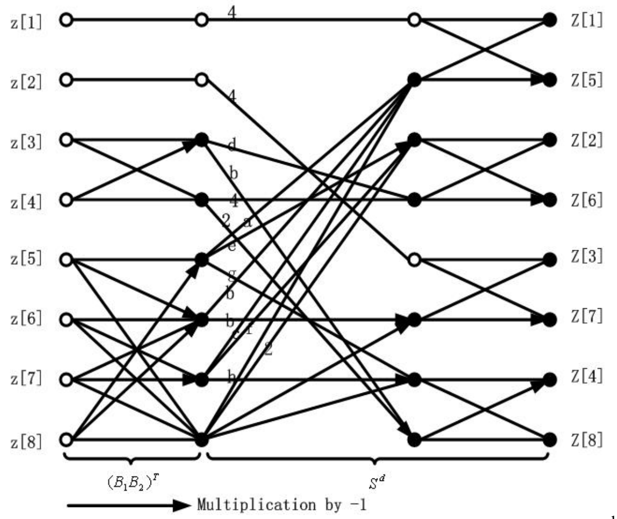


Figure 1. Flow graph for DCT-to-IT transform using S^d

When we count the number of operations in the flow graph, we get 13 multiplications and 30 additions. The two-dimensional (2D) case will be a repeated 1D application for very row and then for very column of an 8×8 DCT block. It follows that 2D $(B_1B_2)^T$ needs 160 (=16×10) additions and 2D S^d needs 208 (=16×13) multiplications and 320 (=16×20) additions, for total of 688 operations. Thus, our proposed method saves 144 multiplications but increases 128 additions compared with Jun Xin's method. In other words, our method saves 16 operations totally compared with Jun Xin's method. That also means that 128 multiplications in the process of matrix multiplications are replaced by additions.

Table 1 Number of operations for DCT-to-IT transform

Method	Add	Mul	Shift	Sum of Operations
Pixel domain	672	256	64	992
Bo Shen	352	352	64	768
Jun Xin	352	352	0	704
Our Proposed	480	208	0	688

The number of operations required in different methods for DCT-to-IT transform is tabulated in Table 1. Considering that the real-arithmetic multiplication operation is usually three to four times higher time overhead than the real-arithmetic addition operation in most processor, our proposed method saves more computational complexity and can be implemented more efficiently in ASIC, DSP and media processor.

4. SIMULATION RESULTS

We adopt MPEG software simulation group MPEG-2 software decoder [11] to decode MPEG-2 input video bit-streams. The decoded DCT coefficients are converted to IT coefficients using Jun Xin's method and our proposed method respectively. In order to avoid the influences of H.264 coding tools, such as intra prediction and variable block size motion compensation, the IT coefficients are directly subjected to H.264 quantization, inverse quantization and inverse IT processes (the same processes in the reference software H.264/AVC JM8.2 [12]) instead of H.264 re-encoding process to get the reconstructed pixels data. The H.264 re-quantization parameter ranges from 0 to 51 corresponding to the full H.264 quantization parameter range. The MPEG-2 bit-streams for simulation are compressed in the MPEG-2 encoder [11]. All MPEG-2 bit-streams are intra-coded with the frame rate of 30 frames/s with four different target bit-rates: 2.5 Mbps/s, 3.5 Mbps/s, 4.5 Mbps/s and 6 Mbps/s respectively. The block diagram of simulation setting is shown in Fig. 2.

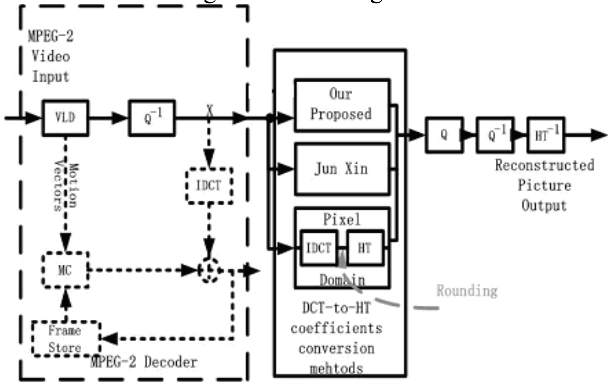
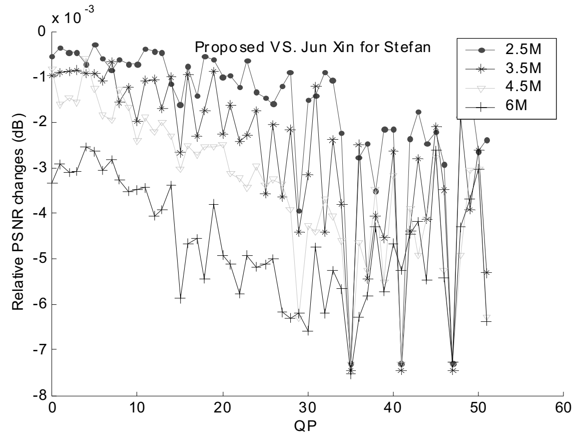


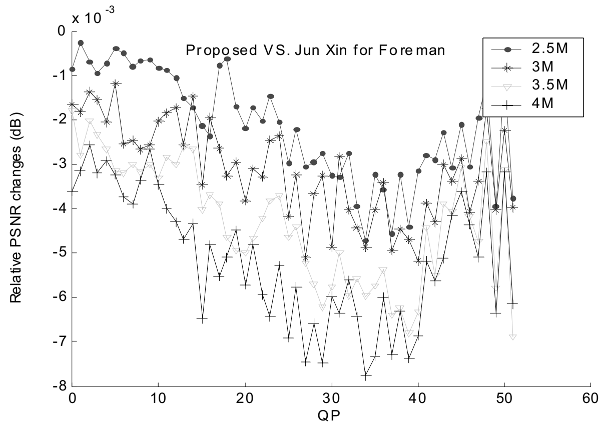
Figure 2. Simulation setting

Extensive simulations and performance comparison have been done with different motion characteristic sequences, but we only give the comparison results of STEFAN and FOREMAN which are shown in Fig. 3 and the similar results of other sequences are omitted in this paper due to the limit of page. The actual runtime requirements for both methods are to be presented elsewhere in a separate paper [13] also for the same reason.

Though our proposed method is not approximation of Jun Xin's method and is equivalent in terms of functionality, it still causes a little quality degradation. This is because that there exits the rounding error in the implementing of absorbing the diagonal matrix D to MPEG-2 de-quantization process. However, the degradation is very small with the maximum value lower than 8×10^{-3}dB as shown in Fig. 3, which is completely negligible in practice.



(a) STEFAN



(b) FOREMAN

Figure 3. Relatively average PSNR difference of our proposed method vs. Jun Xin's method for STEFAN (a) and FOREMAN (b) by changing the re-quantization parameter form 0 to 51 with four different input bit rates

5. CONCLUSION

We have proposed a novel DCT-to-IT transform kernel matrix S^d based on the factorization of DCT and furthermore a fast computing process for the multiplication of S^d exploited the sparseness and symmetry of S^d is presented. Relative to Jun Xin's method, our proposed method saved 16 operations for each 8×8 DCT blocks, while achieved almost the same video quality. In order to further reduce the computation of DCT-to-IT transform, the integer form of S^d can be easily derived using the similar method described in [7], and the multiplications to compute S^d also can be replaced by just using add operations and shift operations just like doing in [8].

6. ACKNOWLEDGEMENT

This work is supported by National Nature Science Foundation of China (60302028) and Key Project of the International Technical Cooperation (2005DFA11060).

7. REFERENCES

[1] Thomas Wiegand, Gary Sullivan. "Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITUT Rec. H.264 | ISO/IEC 14496-10 AVC)," JVT-G050, Pattaya, Thailand, Mar. 2003.
[2] ISO/IEC JTC11/SC29/WG11, "Generic Coding of Moving Pictures and Associated Audio Information: Video", ISO/IEC 13818-2. May 1994.A. N. Netravali and B. G. Haskell, *Digital Pictures,* 2nd ed., Plenum Press: New York, 1995, pp. 613-651.
[3] A. Vetro, C. Christopoulos, and H.Sun. "Video Transcoding Architectures and Techniques: An Overiew". IEEE Signal Processing Magazine, Vol 20, no.2, pp.18-29, March. 2003.
[4] Hari Kalva. "Issues in H.264/MPEG-2 Video Transcoding". CCNC 2004. First IEEE, 5-8 Jan. 2004.
[5] Hari Kalva, Branko Petljanski, and Borko Furht. "Complexity Reduction Tools for MPEG-2 to H.264 Video Transcoding." WSEAS Transactions on Information Science & Applications, Vol. 2, Issues, Marc. 2005, pp. 295-300.
[6] Chen Chen, Ping-Hao Wu, and Chen, H, "Transform-domain intra prediction for H.264," Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on 23-26 May 2005 Page(s):1497-1500 Vol 2
[7] J. Xin, A. Vetro and H.Sun, "Converting DCT coefficients to H.264/AVC transform coefficients," IEEE Pacific-Rim Conference on Multimedia (PCM), Lecture Notes in Computer Science, ISSN: 0302-9743, November 2004, Vol.3332/2004 pp. 939.
[8] Bo Shen, "From 8-tap DCT to 4-tap integer-transform for MPEG to H.264 transcoding". Proc IEEE International Conference on Image Processing, ICIP '04, Oct 2004, Vol 1, Page(s): 115 – 118.
[9] W.B.Pennebaker and J.L.Mitchell, "JPEG Still Image Data Compression Standard," Van Nostrand Reinhold, 1993.
[10]Y. Arai, T.Agui and M. Nakajima, " A Fast DCT-SQ scheme for Images," Trans. Of the IEICE, E 71(11): 1095, November 1988.
[11] MPEG-2 video decodec v12, available online at http://www.mpeg.org/MPEG/MSSG.
[12] H.264/AVC reference software JM8.2, available online at http://bs.hhi.de/~suehring/tml/download.
[13] Gao Chen, shouxun Lin, and Yongdong Zhang, "A Fast Scheme for Converting DCT Coefficients to H.264/AVC Integer Transform Coefficients", submitted to ICIAR2006.