A Fast Scheme for Converting DCT Coefficients to H.264/AVC Integer Transform Coefficients

Gao Chen, Shouxun Lin, and Yongdong Zhang

Institute of Computing Technology, Chinese Academy of Sciences
Beijing, 100080, P.R. China
{chengao, sxlin, zhyd}@ict.ac.cn

Abstract. Converting MPEG-2 8-tap discrete cosine transform (DCT) coefficients to H.264/AVC 4-tap integer transform (IT) coefficients is one of the indispensable operations for MPEG-2 to H.264 transform domain transcoding. In this paper, we propose a novel transform kernel matrix based on the factorization of DCT to perform this operation directly in transform domain. Furthermore, additional reduction in computation can be obtained by taking advantage of the fact that most of the 8×8 DCT blocks in real video sequences only have nonzero coefficients in the upper left 4×4 quadrant. Relative to Jun Xin's method, the proposed method saves 16 operations in the worst case and 440 operations in the best case for each 8×8 DCT block. Experimental results also show that the proposed method efficiently reduces the computational cost up to 42.9% with negligible degradation in video quality. Hence, the proposed method can be efficiently used in the real-time and high capacity MPEG-2 to H.264 transform domain 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, can offer 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 over the next several years, but the complete migration to H.264 will take several years given the fact that MPEG-2 has been widely used in many applications today, including DVD and digital TV. With the deployment of H.264, e.g., for mobile devices, there is a need to develop transcoding technologies to convert videos in the MPEG-2 format into the videos in the H.264 format and vice versa [3]. Furthermore, there is a clear industry interest in technologies facilitating the migration from MPEG-2 to H.264 [4].

In the implementing of MPEG-2 to H.264 transcoder, reducing the computational complexity is a major issue, especially for real-time implementing [3]. It is well known that transform domain transcoding techniques may be simpler than the conventional pixel domain transcoding techniques, since the former avoid the complete decoding and re-encoding which are computationally expensive [3]. 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 transform domain. Unlike previous transform

A. Campilho and M. Kamel (Eds.): ICIAR 2006, LNCS 4141, pp. 559 – 570, 2006.

domain transcoding, such as H.263 to MPEG-2 transcoding, the decoded DCT coefficients in the MPEG-2 to H.264 transcoder can not be reused directly and have to be converted to IT coefficients. This is because that MPEG-2 uses 8-tap DCT to produce the transform coefficients, while H.264 uses a 4-tap IT to do so. So, one of the indispensable steps in the transform domain MPEG-2 to H.264 transcoding is to convert MPEG-2 DCT coefficients to IT coefficients i.e., DCT-to-IT coefficients conversion.

Based on the fact that the DCT transform matrix is orthonormal and separable, Jun Xin [5] and Bo Shen [6] have derived two different transform kernel matrices and have shown their transform domain conversion methods outperform the pixel domain approach. Although Jun Xin's method needs less 64 shift operations and only two calculating stages compared with Bo Shen's method, there are 19 non-trivial elements in Jun Xin's transform kernel matrix, which cause one matrix multiplication (8×8 with 8×8) to have a total of 352 operations. The computational complexity of Jun Xin's method can be reduced through the factorization of the transform kernel matrix. Furthermore, both of the two existing methods do not utilize the property that a large percentage of 8×8 DCT blocks in real video sequences, we refer as low pass block thereinafter, only have nonzero DCT coefficients in the upper left 4×4 quadrant.

The role of DCT-to-IT coefficients conversion 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. 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 operation of DCT-to-IT conversion. In this work, we propose a fast scheme to speed up this operation. The rest of the paper is organized as follows. Section 2 describes our proposed fast scheme for DCT-to-IT coefficients conversion. A performance evaluation of the proposed algorithm in terms of computational complexity and distortion result is carried out in Section 3. Finally, Section 4 concludes this paper.

2 Fast Scheme for DCT-to-IT Coefficients Conversion

Since the DCT-to-IT coefficients conversion can be represented as multiplication by a fixed matrix, fast multiplication by this fixed matrix is possible if it can be factorized into a product of sparse matrices whose entries are mostly 0, 1 and –1. We will demonstrate that this can be done efficiently by the factorizing of Jun Xin's transform kernel matrix. Furthermore, since the energy distribution of DCT blocks obtains from the real MPEG-2 video sequences mainly concentrate on the low frequency region, it is beneficial to exploit this property to speed up the DCT-to-IT coefficients conversion in the case of transcoding. The detail process of our proposed fast DCT-to-IT coefficients conversion scheme is described in the following.

2.1 Factorization of the Transform Kernel Matrix

Let T_8 be the transform kernel matrix of DCT, H be the IT transform matrix: $H = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{pmatrix}$, $K = \begin{pmatrix} H & 0 \\ 0 & H \end{pmatrix}$. Jun Xin's transform kernel matrix

(we refer as S or S-transform thereinafter), which is shown in (1) and (2), is first adopted as our initial transform kernel matrix, since it requires less 64 operations and only two calculating stages compared with Bo Shen's one.

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

Where the superscript T denotes matrix transposition.

$$S = \begin{pmatrix} a & b & 0 & -c & a & d & 0 & -e \\ 0 & f & g & h & 0 & -i & -j & k \\ 0 & -l & 0 & m & a & n & 0 & -o \\ 0 & p & j & -q & 0 & r & g & s \\ a & -b & 0 & c & 0 & -d & 0 & e \\ 0 & f & -g & h & 0 & -i & j & k \\ 0 & l & 0 & -m & a & -n & 0 & o \\ 0 & p & -j & -p & 0 & r & -g & s \end{pmatrix} \tag{2}$$

Where the values a…s are (rounded off to four decimal places): a= 1.4142, b= 1.2815, c=0.45, d= 0.3007, e= 0.2549, f= 0.9236, g= 2.2304, h=1.7799, i=0.8638, j=0.1585, k=0.4824, l=0.1056, m=0.7259, n=1.0864, o=0.5308, p=0.1169, q=0.0922, r=1.0379, s=1.975. For the proof and more details of the S, please refer to [5].

Then, we shall focus on efficient factorization of the S. A factorization of DCT that is the fastest existing algorithm for 8-point DCT due to Arai, Agui, and Nakajima [7] [8], is exploited to perform the factorizations of the transform kernel matrix S. According to this factorization, T_8 is represented as $T_8 = DPB_1B_2MA_1A_2A_3$, which we state without proof.

Thus, we have

$$S = K \times T_8{}^T = \overbrace{K \times A_3{}^T A_2{}^T A_1{}^T M^T} B_2{}^T B_1{}^T P^T D^T \tag{3}$$

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

$$S = \overbrace{K \times A_3{}^T A_2{}^T A_1{}^T M^T} \times (B_1B_2)^T \times P \times D \tag{4}$$

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 defining: $S^d = K \times A_3{}^T A_2{}^T A_1{}^T M^T$, we have

$$S = S^d \times (B_1B_2)^T \times P \times D \tag{5}$$

The DCT-to-IT coefficients conversion now can be carried out multiplication by $(B_1B_2)^T$ and S^d in turn (for simplicity, we refer as $\mathbf{S}^d$ or S^d-transform thereinafter). The multiplication with D can be ignored since it can be absorbed in MPEG-2 inverse quantization matrix without any change in arithmetic complexity of the dequantizer.

The matrix P causes only changes in the order of the components it can be ignored as well. The matrix of (B_1B_2) only contains 0, 1, and -1 and only S^d contains non-trivial elements as shown in the following.

$$B_1B_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 \\ 0 & 0 & 0 & 0 & -1 & -1 & 0 & 1 \end{pmatrix} \tag{6}$$

$$S^d = \begin{pmatrix} 4 & 0 & 0 & 0 & a & b & c & 1 \\ 0 & 0 & d & 4 & -e & 0 & f & 2 \\ 0 & 4 & 0 & 0 & 0 & -b & 0 & 1 \\ 0 & 0 & -b & 2 & g & 0 & -h & 1 \\ 4 & 0 & 0 & 0 & -a & -b & -c & -1 \\ 0 & 0 & -d & -4 & -e & 0 & f & 2 \\ 0 & 4 & 0 & 0 & 0 & b & 0 & -1 \\ 0 & 0 & b & -2 & g & 0 & -h & 1 \end{pmatrix} \tag{7}$$

Where the 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.

Let us count the number of operations that are required for performing the above calculating process. Just like the 2D S-transform, the 2D S^d-transform is also separable. For the sake of simplicity, let us confine attention first to the one-dimensional case. The two-dimensional case will be a repeated application for very row and then for very column of each 8×8 DCT block. Let z be an 8-dimensional column vector, and a vector Z be the 1D transform of z. The sparseness and symmetry of S^d can be exploited to perform fast computation of the S^d-transform. The following steps describe the calculating process, which is 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 to get the transform results:

$m_1 = 4 \times x[1]$
$m_2 = a \times x[5] + b \times x[6] + c \times x[7] + x[8]$

$$m_3 = f\times x[7] + 2\times x[8] - e\times x[5]$$
$$m_4 = d\times x[3] + 4\times x[4]$$
$$m_5 = 4\times x[2]$$
$$m_6 = x[8] - b\times x[6]$$
$$m_7 = g\times x[5] - h\times x[7] + x[8]$$
$$m_8 = b\times x[3] - 2\times x[4]$$

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

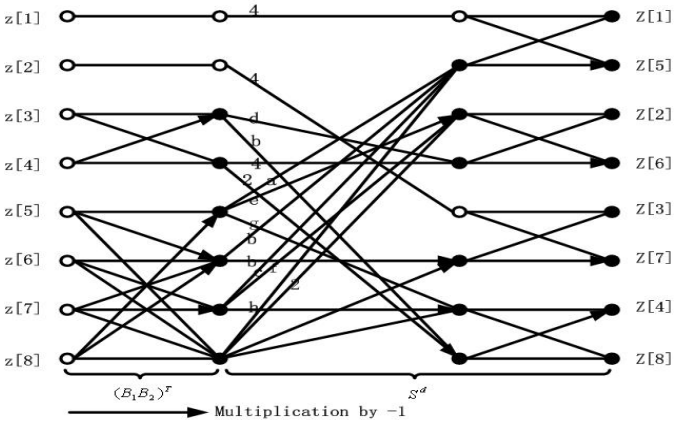

Fig. 1. Fast implementing for S^d-transform

The overall computational requirement of the one dimension calculating process is 15 multiplications and 28 additions. It follows that 2D $(B_1B_2)^T$ needs 160 (=16×10) additions and 2D S^d needs 240 (=16×15) multiplications and 288 (=16×18) additions, for total of 688 operations. Thus, when compared with the S-transform, the S^d-transform saves 16 operations, where it saves 112 multiplications but increases 96 additions. In addition, our proposed S^d-transform requires three calculating stages, where the $(B_1B_2)^T$ needs one and the S^d needs two. S^d can be used as a novel transform kernel matrix to perform DCT-to-IT coefficients conversion.

2.2 Simplified S-Transform for the Low Pass Block

The key to reducing the video transcoding complexity is to reuse the information gathered during MPEG-2 decoding stage [3]. Either the S-transform or the S^d-transform can be explicitly used for a low pass block. However, this is not efficient since both of they do not utilize the coefficients distribution property of a low pass

block. In the following, we will demonstrate that a significant saving of computation can be got by taking advantage of the fact that only the low frequency coefficients are nonzero.

We first select to adopt the S-transform as the foundations to simplify the calculating process. Note that one can argues that why not use the S^d-transform to replace the S-transform in processing the low pass block, since the former requires less operations in computation. The reason is that before multiplications by S^d, the input low pass block first needs to be multiplied by $(B_1B_2)^T$ which resulting the intermediate resulting block is no longer a low pass block. So, adopting the S^d-transform cannot exploit the low pass assumption for the computation of the matrix multiplication.

For the same reasons mentioned above, we still first discuss the one dimension case. Let z be an 8-dimensional column vector, and a vector Z be the 1D transform of z. Considering that elements z[5]–z[8] equal to zero for a low pass column vector, the calculating steps described in [5] can be simplified as:

$$
\begin{aligned}
m_1 &= a \times z[1] \\
m_2 &= b \times z[2] - c \times z[4] \\
m_3 &= g \times z[3] \\
m_4 &= f \times z[2] + h \times z[4] \\
m_6 &= m \times z[4] - l \times z[2] \\
m_7 &= j \times z[3] \\
m_8 &= p \times z[2] - q \times z[4]
\end{aligned}
$$

$$
\begin{aligned}
Z[1] &= m_1 + m_2 \\
Z[5] &= m_1 - m_2 \\
Z[2] &= m_3 + m_4 \\
Z[6] &= m_4 - m_3 \\
Z[3] &= m_6 \\
Z[7] &= -m_6 \\
Z[4] &= m_7 - m_8 \\
Z[8] &= m_7 + m_8
\end{aligned}
$$

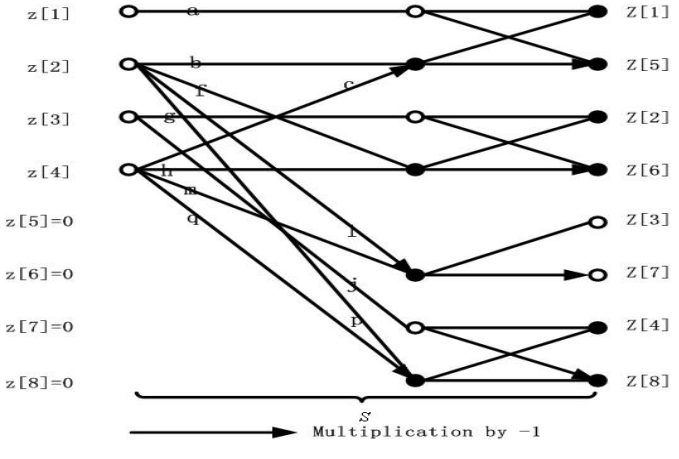

Fig. 2. Simplified S transform for low pass block

Where the const values a…. q have the same value in equation (2) and the zero variables are eliminated form the computing steps. The corresponding flow-chart is depicted in Fig. 2. This simplified calculating process is referred as the simplified S-transform hereafter.

The simplified S-transform totally needs only 11 multiplications and 11 additions. Furthermore, considering that the 2D transform of a low pass 8×8 DCT block needs only four 1D column transforms and eight 1D row transforms due to that the last 4 column vectors in a low pass block are all zero vector. So, It follows that 2D transform needs 132 (=12×11) and 132 (=12×11) additions, for total of 264 operations. Thus, the simplified S-transform saves 62.5% of operations (i.e., 440 operations) than the S-transform for a low pass 8×8 DCT block.

2.3 Summary of the Proposed Scheme

The operations and calculating stages needed in different approaches for DCT-to-IT coefficients conversion are tabulated in Table 1. Note that the simplified S-transform is only used for the low pass block.

Table 1. The number of operations and calculating stages for DCT-to-IT coefficients conversion

Method	Pixel domain	Bo Shen	S-transform	S^d-transform	Simplified-S
Add	672	352	352	448	132
Mul	256	352	352	240	132
Shift	64	64	0	0	0
Sum of operations	992	768	704	688	264
Calculating Stages	6	4	2	3	2

In order to reduce the computation complexity as much as possible, we propose a fast scheme that operates in two steps.

Step 1, the input MPEG-2 8×8 DCT blocks are classified as two types, that are the low pass blocks and the non-low pass blocks.

Usually, the MPEG-2 to H.264 transcoder need to perform an entropy decoding to extract the motion vectors, quantized DCT coefficients form the input MPEG-2 video bit stream. At the same time, the raster scan order of each nonzero DCT coefficients within an 8×8 DCT block is also available to the transcoder. Exploiting such information, if no one of the nonzero DCT coefficients of a 8×8 DCT block is located in the upper left 4×4 quadrant, this 8×8 DCT block is classified as a low pass block, otherwise, it is classified as a non-low pass block. It is well known that only the nonzero coefficients are encoded in MPEG-2 bit-stream. The computation for judging the type of block is very little compared with the computation of DCT-to-IT coefficients conversion and can be ignored.

Step 2, We adopt the optimum algorithm to perform DCT-to-IT coefficients conversion according to the type of DCT block, that is using the S^d-transform for the non-low pass blocks and using the simplified S-transform for the low pass block to perform DCT-to-IT coefficients conversion respectively.

The block diagram of our proposed fast DCT-to-IT coefficients conversion scheme is depicted in Fig. 3.

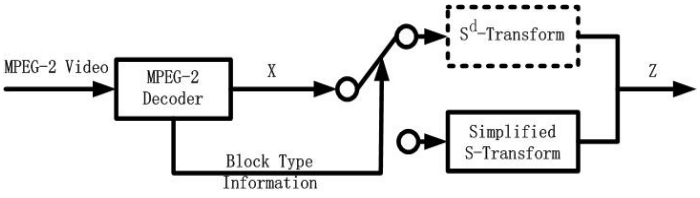

Fig. 3. The proposed fast scheme for DCT-to-IT coefficients conversion

In the worst case, where all input 8×8 DCT blocks are non-low pass blocks, our proposed scheme can save 16 operations for each 8×8 DCT block, and in the best case, where all input 8×8 DCT blocks are low pass blocks, our proposed scheme can save 440 operations for each 8×8 DCT block compared with Jun Xin's method. The practical reduction of complexity depends on the percentage of low pass blocks in video sequences. The distortion and speed performance of the scheme are quantitatively evaluated in following section.

3 Experimental Results

In order to evaluate our proposed method, we adopt MPEG Software Simulation Group (MSSG) MPEG-2 software decoder [9] to decode the input MPEG-2 test videos. A flag, i.e., low pass block flag, which denote whether an 8×8 DCT block is a low pass block or not, is adopted in simulations. The decoded 8×8 DCT block (X) and the associated low pass block flag are sent to three processing systems, which adopting the pixel domain method, the Jun Xin's method, and our proposed method respectively. Each of the three processing systems converts X to the IT coefficients. In order to avoid the influences of other 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 transform process, which are the same processes as in the reference software H.264/AVC JM8.2 [10], to get the reconstructed images. The H.264 re-quantization QP factors ranging from 0 to 51, corresponding to the full H.264 QP range. The block diagram of simulation setting is shown in Fig. 4.

The first 90 frames of each of test sequences in CIF resolution (352×288) are precoded to four different MPEG-2 test videos at 2.5Mbps/s, 3Mbps/s, 3.5Mbps/s and 4Mbps/s respectively. These MPEG-2 test videos are all intra-coded at a frame rate of 30 fps. All simulations are performed using Windows XP, Intel P4 2.8GHz CPU and 1GB memories and Visual C++ 6.0 Compiler. The performance is measured using mean Peak Signal to Noise Ratio (PSNR) between the reconstructed frame sequence and the corresponding original uncompressed frame sequence. The computational complexity is measured using the mean runtime of the all DCT block in one frame needed to convert to IT coefficients. Furthermore, the mean runtime is obtained from the average value of three repeated simulations.

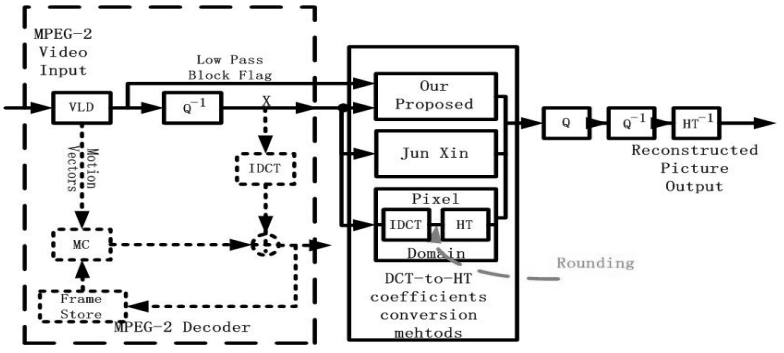

Fig. 4. Simulation setting

A series of experiments processing a great deal of video sequences containing different amounts of motion and spatial details have been made. The results of different sequences are similar except that the percentages of low pass block are different, but due to the limit of page, we only show the results of two sequences: STEFAN and TEMPLATE.

3.1 Our Proposed Method vs. Pixel Domain Method

Fig. 5 shows the PSNR difference between our proposed method and pixel domain method. For all of the four bit-rate, the relative PSNR difference is very small, with the maximum gain around 14×10^{-3}dB and maximum loss around 5×10^{-3}dB. The on the whole superior quality of our proposed method is due to the rounding operation is taken place for the result IT coefficients. On the contrary, for pixel domain method, the saturation and mismatch control (the standard MPEG-2 decoding steps following inverse DCT to reconstruct pixel data), which act as the rounding operation, are taken place for the intermediate results.

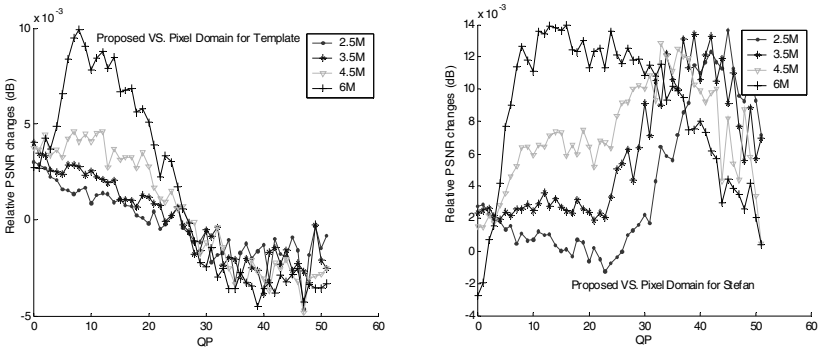

Fig. 5. Relative PSNR difference of our proposed method vs. pixel domain method for (a) TEMPLATE and (b) STEFAN

3.2 Our Proposed Method vs. Jun Xin's Method

Fig. 6 shows the PSNR difference between our proposed method and Jun Xin's method. Though our proposed method is equivalent to Jun Xin's method in terms of mathematics, we can see that our proposed approach produces a little quality degradation compared with Jun Xin's method. The factor is that there exits the rounding error in the implementing of absorbing the diagonal matrix D to MPEG-2 de-quantization process. However, the minus PSNR gain is very small, with the maximum value lower than 6×10^{-3}dB, which is negligible in practice, especially for practical transcoding application, where the bit-rate reduction and spatial/temporal resolution downscale transcoding are usually considered as the goal. So we can say that our method products almost identical results as Jun Xin's.

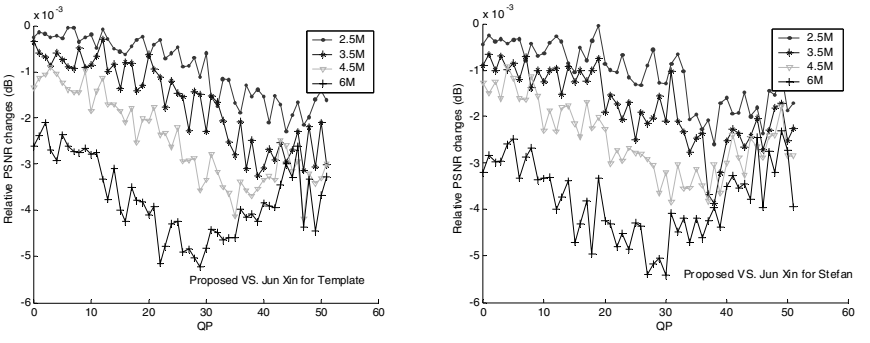

Fig. 6. Relative PSNR difference of our proposed method vs. Jun Xin's method for (a) TEMPLATE and (b) STEFAN

3.3 Complexity Comparison

The overall percentages of low pass blocks for each sequence at different bit-rate are shown in Table 2. As expected, the lower the bit-rate is, the more of the low pass blocks are in video sequences. This is due to that in order to meet the requirement of target bit-rate, the video encoder has to adopt high quantization parameter, which may cause more DCT coefficients to be quantized to zero. The actual runtime requirements (in terms of microseconds used) for each method at different bit-rates are also shown in Table 2. The relative improvements of our proposed method compared with Jun Xin's approach are shown inside the parenthesis. Our proposed method achieves the lowest runtime compared with the other methods and saves about 34.3%–42.9% computation compared with Jun Xin's method. The complexity reduction is proportional to the percentage of the low pass blocks in one real video sequence.

It is interesting to note that the reduction of computational complexity of Jun Xin's method compared with the pixel domain method is not obvious. This is because that Jun Xin's method increases 96 multiplication operations compared with the pixel domain method as shown in Table 1 and the real-arithmetic multiplication operation is usually three to four times higher time consuming than the real-arithmetic addition operation in our simulation setting.

Table 2. Runtime for DCT-to-IT coefficients conversion

Test Sequence	Bit-Rate (Mb/s)	Low pass block percentage	Runtime for DCT-to-IT coefficients conversion (ms)		
			Pixel Domain	Jun Xin	Proposed
TEMPLATE	2.5	60.6%	12.82	12.25	7.29 (40.5%)
	3.5	44.2%	13.03	12.38	7.47 (39.7%)
	4.5	36.0%	12.86	12.22	7.60 (37.8%)
	6	27.1%	13.05	12.12	7.84 (35.3%)
STEFAN	2.5	59.7%	13.21	12.18	6.95 (42.9%)
	3.5	49.4%	13.20	12.39	7.16 (42.2%)
	4.5	41.6%	13.11	12.33	7.50 (39.2%)
	6	28.2%	13.24	12.26	8.06 (34.3%)

4 Conclusion

In this paper, we derive a fast scheme to reduce the computational complexity of the DCT-to-IT coefficients conversion. We analyze the operations involved in our proposed fast scheme. The efficiency of our proposed fast scheme is also quantitatively evaluated. Our simulation results show that the proposed method leads to about 34.3%–42.9% saves in computational complexity, with negligible impact in PSNR of less 6×10^{-3}dB. In our simulations, only the intra-coded MB is used to test. Considering that the DCT coefficients decoded from inter-coded MBs of MPEG-2 video are computed from the motion compensated residual, there is more low pass block in inter-coded frames than in intra-coded frames. Our proposed method can be more efficiently used in inter-coded frames transcoding. Due to its efficiency, the proposed approach has been applied to our on-going real-time MPEG-2 to H.264 transcoder.

Acknowledgement

This work is supported by the National Nature Science Foundation of China (60302028), by the Key Project of International Science and Technology Cooperation (2005DFA11060), by the Key Project of Beijing Natural Science Foundation (4051004).

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 iformation: vdeo”, 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 tanscoding achitectures and tehniques: 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. 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, Vol. 3332/2004, pp. 939, November 2004
6. Bo Shen, "From 8-tap DCT to 4-tap integer-transform for MPEG to H.264 transcoding," Proc. IEEE International Conf. on Image Processing (ICIP04), Oct. 2004
7. Y. Arai, T. Agui and M. Nakajima, “ A fast DCT-SQ scheme for images,” Trans. Of the IEICE, E 71(11): 1095, November 1988
8. W. B. Pennebaker and J. L. Mitchell, JPEG still image data compression standard, Van Nostrand Reinhold, 1993, pp. 53-57
9. MPEG-2 video decodec v12, available online at http://www.mpeg.org/MPEG/MSSG
10. 10.H.264/AVC reference software JM8.2, available online at http://bs.hhi.de/~suehring/tml/download