A Low Complexity Integer Transform and Quantization for Video Coding

ZHOU JINGLI, XIANG DONG, CHEN JIAZHONG, YU SHENGSHENG，XU JUN
Department of Computer Science and Engineering
Huazhong University of Science & Technology
Wuhan, Hubei, 430074
P. R. China

Abstract: - This paper presents a new low complexity integer 4x4 transform and quantization. The new 4x4 transform is more approximate to discrete cosine transform (DCT) matrix than the transform in H.264. Moreover, the transform can be calculated without multiplication, just additions and shifts, in 16-bit arithmetic, thus minimizing computational complexity. The quantization can match the distribution of coefficients better than that in H.264. Simulation results reveal a performance increasing up to 1.87% bit rate savings and 0.1 dB in peak signal-to-noise ratio.

Key-Words: - Integer Transforms, Quantization, Video Coding, H.264

1 Introduction

The DCT is widely applied in block-based hybrid video coding standards. This increases the possibility of drift (mismatch between the decoded data in the encoder and decoder) [1]. The state-of-the-art H.264 video coding standard adopts integer 4×4 transforms, which can be computed exactly in integer arithmetic, thus avoiding inverse transform mismatch problems. In addition the H.264 architecture has many innovations such as variable block-size motion compensation with small block sizes, quarter-sample accurate motion compensation, multiple reference pictures motion compensation, in-the-loop de-blocking filtering, and context-adaptive entropy coding. H.264 has achieved a significant improvement in rate-distortion (RD) efficiency relative to existing standards [2].

In this paper, a new integer 4×4 DCT transform and quantization are presented. The transform can be calculated without multiplication, just additions and shifts, in 16-bit arithmetic, thus minimizing computational complexity. The quantization can match the distribution of coefficients better than that in H.264. Experimental results show that this method has better RD performance than that in H.264.

2 Integer Transform Design

The DCT is commonly used in block transform coding of images and video, e.g., JPEG and MPEG, because it is a close approximation to the statistically optimal K-L transform, for a wide class of signals [3]. A 2-dimension 4×4 forward DCT transform can be written as follows:

$$Y = AXA^T = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix} X \begin{bmatrix} a & b & a & c \\ a & c & -a & -b \\ a & -c & -a & b \\ a & -b & a & -c \end{bmatrix} \quad (1)$$

Where X is the 4×4 input signals, A is the transform matrix whose entries are:

$$a = 0.5, b = \sqrt{0.5} \times \cos(\pi/8), c = \sqrt{0.5} \times \cos(3\pi/8) \quad (2)$$

The above equation can be written as:

$$Y = \left(CXC^T\right) \otimes E_f = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ d & 1 & -1 & -d \\ 1 & -1 & -1 & 1 \\ 1 & -d & d & -1 \end{bmatrix} X \begin{bmatrix} 1 & d & 1 & 1 \\ 1 & 1 & -1 & -d \\ 1 & -1 & -1 & d \\ 1 & -d & 1 & -1 \end{bmatrix} \right) \otimes \begin{bmatrix} a^2 & ac & a^2 & ac \\ ac & c^2 & ac & c^2 \\ a^2 & ac & a^2 & ac \\ ac & c^2 & ac & c^2 \end{bmatrix} \quad (3)$$

Where $d = b/c$, $\otimes$ denotes element-by-element multiplication instead of normal matrix multiplication. Since multiplying with elements of matrix E_f consists of simply scaling the output with constants, this scaling can be combined with the quantization tables at the encoder, and with de-

quantization tables at the decoder. The inverse transform for the previously presented DCT can be calculated accordingly as:

$$X = C^T (Y \otimes E_i) C =$$

$$\begin{bmatrix} 1 & d & 1 & 1 \\ 1 & 1 & -1 & -d \\ 1 & -1 & -1 & d \\ 1 & -d & 1 & -1 \end{bmatrix} \left(Y \otimes \begin{bmatrix} a^2 & ac & a^2 & ac \\ ac & c^2 & ac & c^2 \\ a^2 & ac & a^2 & ac \\ ac & c^2 & ac & c^2 \end{bmatrix} \right) \begin{bmatrix} 1 & 1 & 1 & 1 \\ d & 1 & -1 & -d \\ 1 & -1 & -1 & 1 \\ 1 & -d & d & -1 \end{bmatrix} \quad (4)$$

The coefficient $d = 1/(\sqrt{2} - 1) = 2.4142...$ in the DCT matrix is not a rational number, and therefore cannot be implemented exactly using binary arithmetic. This problem can be solved by integer transform.

In order to develop integer transforms approximating to the DCT matrix, the parameter d (b/c) should be around 2.4 and the sign of (b, c) should be positive. In addition the matrix C should be orthogonal. However, the matrix C is orthogonal whatever the parameter d equals to.

In the transform coding of pictures and video, transform are used to convert highly correlated signals into coefficients of low correlation. Such decorrelation ability may be measured by the transform efficiency η [4]. A large η implies high decorrelation ability.

$$\eta = \frac{\sum_{i=1}^{n} |s_{ii}|}{\sum_{p=1}^{n} \sum_{q=1}^{n} |s_{pq}|} \quad (5)$$

A computer search has been performed to find the sets of (b, c) that give the highest transform efficiency when c is less than 7 (3 bits) and d is less than 5.The DCT, which is widely accepted as the best suboptimal transform, is also compared. Table 1 show the best transforms that have the highest transform efficiencies for ρ equals to 0.9. From the table we find that when d is around 2.4 the transform can get desirable decorrelation performance. To decide which set is best we should consider the following two aspects [5]:

1) For the implementation, we want the parameters to be as small as possible. So the number of binary-addition operations and binary-shifting operations will be less.
2) For the performance, we want the ratios of the entries for each row of the integer transform matrix to be approximated to the DCT matrix.

The transform (17, 7), which was originally adopted by H.264, was replaced by the transform (2, 1) because the former requires 32-bit arithmetic. However, the transform (2, 1) doesn't perfectly approximate to DCT matrix. We select the transform (5, 2) as the best one for better approximation and lower complexity computation. This transform can be written as:

Table 1. The integer transforms that have the best transform efficiencies for ρ equal to 0.9 and c equal to 1, 2…7

Transform efficiency η (%)	Transform (b, c)
95.24	(2,1)
95.62	(5,2)
95.89	(7,3)
95.76	(9,4)
95.78	(12,5)
95.59	(13,6)
95.82	(16,7)
95.73	(17,7)
95.75	DCT

$$Y = \left(C_f X C_f^T \right) \otimes E_f =$$

$$\left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 5 & 2 & -2 & -5 \\ 1 & -1 & -1 & 1 \\ 2 & -5 & 5 & -2 \end{bmatrix} X \begin{bmatrix} 1 & 5 & 1 & 2 \\ 1 & 2 & -1 & -5 \\ 1 & -2 & -1 & 5 \\ 1 & -5 & 1 & -2 \end{bmatrix} \right) \otimes \begin{bmatrix} a^2 & \frac{ac}{2} & a^2 & \frac{ac}{2} \\ \frac{ac}{2} & \frac{c^2}{4} & \frac{ac}{2} & \frac{c^2}{4} \\ a^2 & \frac{ac}{2} & a^2 & \frac{ac}{2} \\ \frac{ac}{2} & \frac{c^2}{4} & \frac{ac}{2} & \frac{c^2}{4} \end{bmatrix} \quad (6)$$

For 8 bits pixel data, the prediction errors need 9 bits. The maximum value for the sum of the absolute values of any row is 14. The output coefficients after the 2-D transform above will span $\log_2 14 \times 14 \cong 7.6$ more bits than the input. Therefore, the transform coefficients have a dynamic range of 17 bits. Since many processors cannot perform 32-bit multiplications in a single cycle, we should keep the dynamic range within 16 bits. We scale the second and fourth column of the matrix C_f^T by 1/2.The forward transform finally becomes:

$$Y = \left(C_f X C_f^T \right) \otimes E_f =$$

$$\left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 5 & 2 & -2 & -5 \\ 1 & -1 & -1 & 1 \\ 2 & -5 & 5 & -2 \end{bmatrix} X \begin{bmatrix} 1 & \frac{5}{2} & 1 & 1 \\ 1 & 1 & -1 & -\frac{5}{2} \\ 1 & -1 & -1 & \frac{5}{2} \\ 1 & -\frac{5}{2} & 1 & -1 \end{bmatrix} \right) \otimes \begin{bmatrix} a^2 & ac & a^2 & ac \\ \frac{ac}{2} & \frac{c^2}{2} & \frac{ac}{2} & \frac{c^2}{2} \\ a^2 & ac & a^2 & ac \\ \frac{ac}{2} & \frac{c^2}{2} & \frac{ac}{2} & \frac{c^2}{2} \end{bmatrix} \quad (7)$$

In above equation the output coefficients after the 2-D transform will need no more than $9 + \log_2 14 \times 7 < 16$ bits. We implement scale operation by 1/2 as right shifts. This will introduce

truncation error. However, these errors are completely masked by quantization errors [6]. The inverse transform accordingly is:

$$X = C_i^T (Y \otimes E_i) C_i =$$

$$\begin{bmatrix} 1 & \frac{5}{2} & 1 & 1 \\ 1 & 1 & -1 & -\frac{5}{2} \\ 1 & -1 & -1 & \frac{5}{2} \\ 1 & -\frac{5}{2} & 1 & -1 \end{bmatrix} \left(Y \otimes \begin{bmatrix} a^2 & ac & a^2 & ac \\ ac & c^2 & ac & c^2 \\ a^2 & ac & a^2 & ac \\ ac & c^2 & ac & c^2 \end{bmatrix} \right) \begin{bmatrix} 1 & 1 & 1 & 1 \\ \frac{5}{2} & 1 & -1 & -\frac{5}{2} \\ 1 & -1 & -1 & 1 \\ 1 & -\frac{5}{2} & \frac{5}{2} & -1 \end{bmatrix} \quad (8)$$

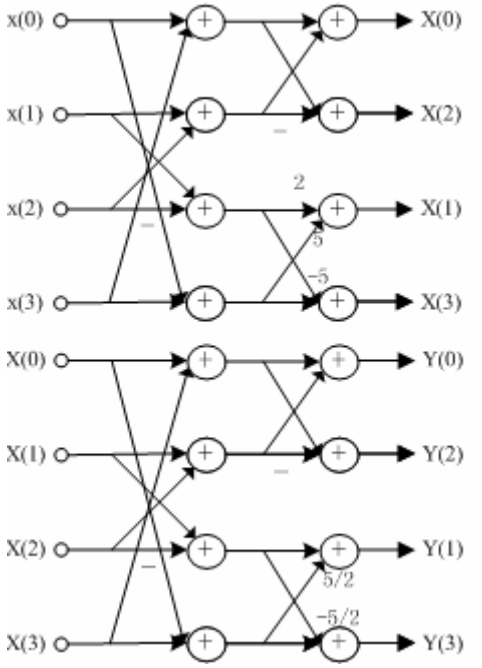

Fig.1 the 1-D row transform (top) and 1-D column transform (bottom) of 2-D forward transform

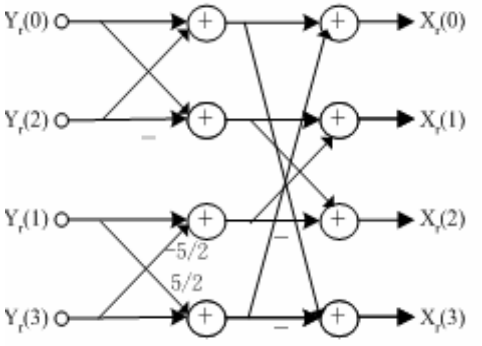

Fig. 2 the 1-D row transform and 1-D column transform of 2-D inverse transform

The 2-D forward transform can be computed as a 1-D row transform followed by a 1-D column transform. The 1-D row transform (top) and 1-D column transform (bottom) are shown in Fig. 1. Fig. 2 shows flow graphs of the inverse transform, which is applied to rows and columns of each 4x4 block. We replace multiplications by addition and left shift operations. For example multiplied by 5 can be expressed as left shift by 2 and one addition. Thus the transform (5, 2) can be calculated without multiplication, just additions and shifts. Relative to the transform (2, 1) in H.264, the transform need two more addition and shift operations for row transform or column transform.

3 Quantization and De-quantization

In lossy compression, quantization is the step that introduces signal loss. The quantization and de-quantization procedure of transform (5, 2), which uses a scalar quantizer without division and float operation, are similar to H.264. The quantization step size roughly doubles for every increase of six in quantization parameter (QP). For a given QP, the forward quantization operation can be written as following [1]:

$$Y_q(i,j) = \text{Sign}(Y(i,j))[(|Y(i,j)| \cdot A(Q_M, i, j) + f \cdot 2^{P+Q_E}) >> (P+Q_E)] \quad (9)$$

Where Q_M=QP%6, Q_E=QP/6, (i, j) are the row and column indices. The parameter P should be as large as possible to reduce rounding errors as long as the elements of A are in the 16-bit range. In our quantization P is set to 17. The corresponding de-quantization procedure is:

$$Y_r(i,j) = Y_q(i,j) B(Q_M, i.j) >> (8 - Q_E) \quad (10)$$

The parameter f in quantization formula, which is typically in the range 0 to 1/2, controls the dead-zone width. In H.264 reference software, the parameter f for INTER coefficients is fixed at 1/6. This may not be a better solution for different motion estimation block sizes. Fig.3 shows the transformed coefficients distribution of 16x16 and 4x4 blocks. There are more coefficients concentrating about zero in 4x4 blocks than in 16x16 blocks. So it is better to choose different f for different block size to achieve optimum image quality. However, this would improve computational complexity. For simplicity we set the parameter f to 1/5 when block sizes are larger than 8x8 and 1/6 otherwise.

The quantization and de-quantization factor matrix A and B are given by $A(Q_M, i, j) = M(Q_M, r)$ and $B(Q_M, i, j) = S(Q_M, t)$,where r=0 and t=0 for (i, j)= {(0,0),(0,2),(2,0),(2,2)}, r=1 and t=1 for (i, j)= {(1,1),(1,3),(3,1),(3,3)}, r=2 and t=2 for (i, j)= {(0,1),(0,3),(2,1),(2,3)} ,r=3 and t=2 otherwise, with

$$\mathbf{M} = \begin{bmatrix} 52429 & 7231 & 27537 & 13768 \\ 47663 & 6574 & 25304 & 12517 \\ 40330 & 5563 & 21182 & 10591 \\ 37449 & 5165 & 19669 & 9835 \\ 32768 & 4520 & 17211 & 8605 \\ 29127 & 4018 & 15298 & 7649 \end{bmatrix}, \mathbf{S} = \begin{bmatrix} 5120 & 1412 & 2689 \\ 5632 & 1554 & 2958 \\ 6656 & 1836 & 3496 \\ 7168 & 1977 & 3765 \\ 8192 & 2260 & 4303 \\ 9216 & 2542 & 4840 \end{bmatrix}$$

4 Experimental Results

We have modified the JM 7.3 software to implement the transform and quantization as specified above. Several CIF sequences were processed with QP varied on 24, 28, 32 and 36. The Sequences are encoded in an IPPP structure at 30 Hz using two reference frames and a search range of 32 pixels. Fig.4 is the RD curve comparison between the presented transform and the transform in H.264. The PSNR gain is listed in Table 2.

Fig.4 shows that the RD curve of the presented transform is above the transform in H.264. The average PSNR gain is +0.046dB, indicating that our method produces a better average PSNR. Therefore, we can state that our presented transform and quantization achieve better RD performance than that in H.264.

5 Conclusions

In this paper a new 4x4 transform and quantization are presented. The new 4x4 transform is a close approximation to DCT and can be calculated without multiplication, just additions and shifts, in 16-bit arithmetic. The quantization can match the distribution of coefficient well. Experimental results show that our method has better RD performance than that in H.264.

References:

[1] Henrique S. Malvar, Antti Hallapuro, et al, Low-complexity Transform and Quantization in H.264/AVC, *IEEE Transactions on Circuits and Systems for Video Technology*, VOL. 13, NO.7, 2003, pp.598-603.

[2] T. Wiegand, G. J. Sullivan, et al, Overview of the H.264 / AVC Video Coding Standard, *IEEE Transactions on Circuits and Systems for Video Technology*, VOL. 13, NO.7, 2003, pp. 1-19.

[3] K. R. Rao and P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications,* Academic Press, 1990.

[4] W. Cham, Development of Integer Cosine Transforms by the Principle of Dyadic Symmetry, *IEE Proc.,* Part 1, vol. 136, 1989, pp. 276–282.

[5] Soo-Chang Pei and Jian-Jiun Ding, The Integer Transforms Analogous to Discrete Trigonometric Transforms, *IEEE Trans. on Signal Processing*, Vol. 48, No.11, 2000, pp.3345-3364.

[6] F. S. Wu and W. K. Cham, A Comparison of Error Behaviour in the Implementation of the DCT and the ICT, *IEEE Region 10 Conference on Computer and Communication Systems*, Vol.2，1990, pp.450-453.

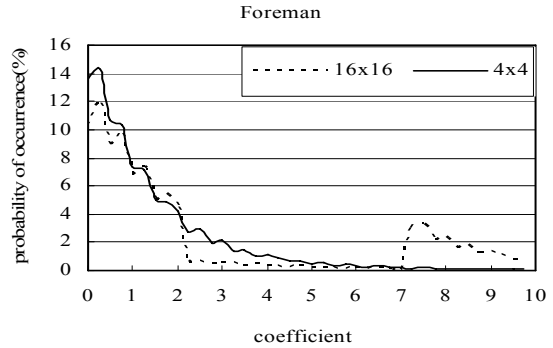

Fig.3 the distribution of INTER coefficients for block 16x16 and 4x4

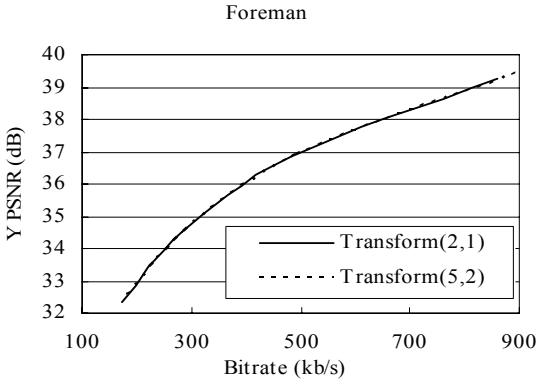

Fig.4 the RD curves comparison of transform (2, 1) and transform (5, 2)

Table 2. The RD performance comparison between the presented transform and transform in H.264

Sequence	Average PSNR gain(dB)	Average bit rate saving (%)
News	+0.1	+1.87
Mobile	+0.009	+0.22
Foreman	+0.07	+1.68
Tempete	+0.038	+0.82
Flowergarden	+0.03	+0.56
Container	+0.03	+0.82
Average	+0.046	+1.00