ADAPTIVE PRUNED 4×4 DCT ALGORITHM FOR H.264

Dongming Zhang
Institute of Computing Technology, Chinese Academy of Science
dmzhang@ict.ac.cn

Shouxun Lin
Institute of Computing Technology, Chinese Academy of Science
sxlin@ict.ac.cn

Yongdong Zhang
Institute of Computing Technology, Chinese Academy of Science
zhyd@ict.ac.cn

Abstract

A new integer 4×4 transform is adopted in the new video coding standard H.264/AVC. To reduce the computation load of the new transform, the SSAVT model for 8×8 transform is applied into H.264 *4×4 transform with some modifications. Then, three classes of pruned block are defined and the corresponding simplified butterfly computation processes are designed. Furthermore, a fast DCT algorithm is proposed, with which we can identify the class of pruned block, so as to adopt simplified computation, with the help of modified model for 4×4 transform. Simulation results show that the proposed DCT algorithm saves about average 55% computation of DCT compared with full computation.*

1. Introduction

The new video standard, H.264 can obtain better video quality with less bitrate compared with the previous standards. But the cost is enormous number of computations required. This limits the rapid application of H.264. Many researchers pay much attention to the complexity reduction of motion estimation since it consumes above 85% computation of the total encoding in joint model (JM). However, when the motion estimation (ME) is improved, the computation of DCT becomes noticeable. Table 1 lists the relative time analysis of one optimized real-time encoder for standard definition sequence which is designed by us on TI DM642 DSP. It illustrates the increasing importance of DCT. This paper aims to build an efficient approximation algorithm for DCT to reduce the complexity of DCT with negligible video quality degradation in H.264 encoder.

Many literatures contribute to the optimization of the computing structure of DCT. These algorithms buffer the intermediate computation results to eliminate the repeated computation. Among these algorithms, the algorithm presented in literature[1] reaches the lowest complexity level.

Table 1. Relative time complexity analysis

Module	Time percentage
Integer ME	26%
Sub pel ME	35%
DCT+Q+IQ+IDCT	17%
Entropy coding	10%
Deblocking	7%
Intra prediction	3%
The rest	2%

Approximation computing brings new chances to accelerate DCT. Two kinds of approximation algorithm are proposed to further accelerate DCT computation. One way is to replace the original float transform matrix with low accuracy transform[4], this way can obtain variable complexity DCT through a series of different accuracy transform matrix. The other is pruned algorithm[2][3]. Pruned algorithm for DCT can reduce the computation of DCT via pruning the quantized DCT coefficients very likely to be zero. In pruned algorithm, a specified subset of block coefficients should be pre-defined according to computation resource, this must lead to some encoding performance loss. To avoid this, Pao and Sun propose an model, statistical of sum of absolute value test (SSAVT), to decision which subset is chosen to be computed depends on the characteristics of the block[5]. The latest contribution to DCT approximation computation comes from literature[6]. It analyzes the pros and cons of [4] and [5] and proposes

0-7695-2650-0/06/$20.00 (c) IEEE

an integrated algorithm which works well from low bitrate to high-medium bitrate.

But these DCT approximation computation algorithms can't apply to the new video coding standard H.264 directly because H.264 uses one 4×4 integer transform as the basic transform instead of float 8×8 DCT. Especially, algorithm in [4] can't apply to H.264 at all since it depends on different approximation of float transform matrix.

In this paper, we firstly introduce the SSAVT model for 8×8 DCT, and we make some necessary modifications on it to apply it into 4×4 transform of H.264. Then we define three class pruned blocks so as to adopt simple computation. At last, an adaptive pruned DCT algorithm is proposed to speed up H.264 4×4 transform. Besides, when DCT coefficients of one block are pruned, the related processes including Q, IDCT, IQ, zig-zag scan, can be simplified as well. So the propose algorithm can reduce the complexity of DCT, Q, IQ and IDCT together.

2. New model of 4×4 DCT coefficients in H.264

SSAVT is built on the two assumptions, i.e. DCT coefficient in a 2-D blocks is an independent random variable with Laplacian distribution[7] and the distribution of the predication residue can be modeled as Laplacian[5].

2.1. Introduction of SSAVT

SSAVT builds a condition quantizing any DCT coefficient in one block to zero with 99% probability. The condition [5] is defined by

$$SAD_N < \frac{QP \cdot q(u,v) \cdot N^2}{3\sqrt{2\Gamma_N(u,v)}} \quad (1)$$

Where, N is transform size, u and v are horizontal and vertical displacements respectively, q is quantization matrix, QP is quantization parameter, $QP \cdot q(u,v)$ indicates the quantization step of the (u,v)-th coefficient, SAD_N is the sum of absolute difference of one $N \times N$ block and can be obtained in the encoding process, $\Gamma_N(u,v)$ is calculated according to the following equation:

$$\Gamma_N(u,v) = [D_N R_N D_N^T]_{(u,u)} [D_N R_N D_N^T]_{(v,v)} \quad (2)$$

Where, D_N is transform matrix of DCT, the superscript T indicates the matrix transpose and R_N is spatial correlation matrix of one block. R_N is defined by equation(3), in which ρ is the one-dimensional correlation coefficient[8].

$$R_N = \begin{bmatrix} 1 & \rho & \rho^2 & \cdots & \rho^{N-1} \\ \rho & 1 & \rho & \cdots & \rho^{N-2} \\ \rho^2 & \rho & 1 & \cdots & \rho^{N-3} \\ \vdots & \vdots & & \ddots & \vdots \\ \rho^{N-1} & \rho^{N-2} & \rho^{N-3} & \cdots & 1 \end{bmatrix} \quad (3)$$

According to inequality(1), SSAVT is capable of selecting the coefficients to be computed adaptively.

2.2 Modified SSAVT for 4×4 DCT

SSAVT is proposed for 8×8 float transform initially. Since the differences exist between 8×8 float transform and 4×4 integer transform, SSAVT can not be applied into 4×4 integer transform directly.

Let X be the residual matrix and Y be the transformed matrix, the actual 4×4 DCT should be:

$$Y = A \cdot X \cdot A^T$$

$$A = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix}$$

Where, $a = 1/2$, $b = \sqrt{1/2}\cos(\pi/8)$, and $c = \sqrt{1/2}\cos(3\pi/8)$.

H.264 chooses one approximation of the actual 4×4 DCT as the basic transform. The approximated transform is described as follow:

$$Y = H \cdot X \cdot H^T \otimes S \quad (4)$$

In equation(4), ⊗ is Kronecker operator and

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}$$

$$S = \begin{bmatrix} a^2 & ab/2 & a^2 & ab/2 \\ ab/2 & b^2/4 & ab/2 & b^2/4 \\ a^2 & ab/2 & a^2 & ab/2 \\ ab/2 & b^2/4 & ab/2 & b^2/4 \end{bmatrix}$$

Where, a=1/2, $b = \sqrt{2/5}$.

Referred to equation(2), $\Gamma_4(u,v)$ should be calculated according to the below equation:

$$\Gamma_4(u,v) = [HR_4H^T \otimes S]_{(u,u)} [HR_4H^T \otimes S]_{(v,v)} \quad (5)$$

On the other hand, according to the quantization scheme of H.264, when scalar matrix S is not integrated into quantization, $QP \cdot q(u,v)$ is independent of the displacement (u,v) and can be deduced to $q(QP)$ for all coefficients in one 4×4 block. Therefore, the condition to check any coefficient quantizing to be zero with 99% probability in H.264 is written as follow:

$$SAD_4 < \frac{q(QP) \cdot 4^2}{3\sqrt{2\Gamma_4(u,v)}} \tag{6}$$

The right-hand-side term of inequality(6) represents the minimum SAD of one 4×4 block.

3. Adaptive pruned DCT algorithm

Theoretically, we can decision any coefficient in one block whether to be zero according to inequality(6), however, the process will consume many compare operations and it is unnecessary since the distribution of DCT coefficients basically obeys to the rule that once some a coefficient is quantized to zero, the coefficients after it will be regarded as all zero. At the mean time, considering the butterfly process is applied, it is not meaningful for reducing DCT computation to check all simplified block. In the following, we will define three pruned blocks.

3.1. Definition of pruned block

In our fast DCT algorithm, we define three classes of block for pruned algorithm, i.e. A, B, and C. Class A block is all-zero block, class B block is block with only DC coefficient and class C block is block with three low frequency coefficients. Block with low frequency 2×2 coefficients in literature[5] is not defined because its four coefficients(1,2,3, and 5) are not continuous in the zig-zag scan order illustrated by Fig.1, while computing the low frequency four neighboring coefficients(1,2,3, and 4) is inconvenient for the implementation of butterfly algorithm[10]. So we define class C block as the block has three low frequency coefficients (1,2, and 3), and the rest coefficients of which are all zero.

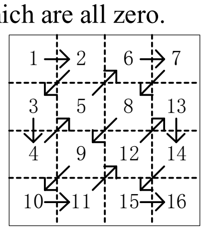

Fig. 1. Zig-zag scan order of 4×4 block

On the other hand, DCT computation of block with more than three coefficients is very closed to full DCT in H.264 considering the implementation of butterfly DCT algorithm, so more block class definitions are not deserved.

3.2. Proposed algorithm description

According to the above block definitions, all blocks can be classified into one of class A, B, C or full block. Inequality (6) is used to adaptively decision the class of one block. We define T_A, T_B, T_C to identify A, B, and C block respectively. They are three thresholds and calculated by the following equations:

$$T_A = \frac{q(QP) \cdot 16}{3\sqrt{2\Gamma_4(0,0)}} = 1.488 \cdot q(QP)$$

$$T_B = \frac{q(QP) \cdot 16}{3\sqrt{2\Gamma_4(0,1)}} = 2.265 \cdot q(QP)$$

$$T_C = \frac{q(QP) \cdot 16}{3\sqrt{2\Gamma_4(2,0)}} = 3.794 \cdot q(QP)$$

In the above equations, $\Gamma_4(u,v)$ is calculated according to equation(5) and experiential ρ is set to 0.65. T_A, T_B, T_C are calculated before encoding and do not lead to any overhead computation.

So the adaptive pruned DCT algorithm is described as follow:

```
1. calculate T_A、T_B、T_C according to initial QP
2. encoding one frame:
for all blocks in the current encoding frame
{
  if(SAD<T_A)
    skip DCT
  else if (SAD>T_C)
    compute full DCT
  else if (SAD<T_B)
    compute only DC coefficient
  else
    compute low frequency 3coefficients
}
```

In the described process, we first detect A class block and full block in order to reduce the number of overhead CMP operations because statistical data show the two class blocks accounts for the most of blocks. So the numbers of CMP operation for class A, B, C and full block are 1, 3, 3 and 2.

3.3. Complexity analysis

DCT can be divided into two processes: horizontal transform and vertical transform, and each process can be regarded as four one-dimensional transforms. Therefore, one full 4×4 DCT contains eight one-dimensional transforms. While the butterfly computation process of one-dimensional transform contains 8 add operations and 2 shift operations. Herein, the computation of one full 4×4 DCT includes

0-7695-2650-0/06/$20.00 (c) IEEE

64 add operations and 16 shift operations and we denote the computation as DCT_{org}.

Table 2. Operations of DCT for four class blocks

Block class	Processing	Operations		
		Add	Shift	CMP
A	Transform	0	0	0
	Overhead	0	0	1
B	Transform	15	0	0
	Overhead	0	0	3
C	Transform	25	5	0
	Overhead	0	0	3
Full	Transform	64	16	0
	Overhead	0	0	2

In the following, we analyze the necessary DCT operations for different class block. Table 2 list the necessary operations of DCT for four class block. To simplify the analysis, we assume that three kind operations: add operation, shift operation and compare operation consumes the same instruction cycles.

For A class block, no operation is used for transform and overhead is one compare operation. The computation is denoted by DCT_A and we obtain the following relation:

$$DCT_A \approx 1.25\% \cdot DCT_{org} \tag{7}$$

When only DC coefficient is nonzero, the transform is simplified into adding 16 residual coefficients and the computation denoted by DCT_B is 15 add operations and 3 compare operations. So we get the following equation.

$$DCT_B \approx 22.5\% \cdot DCT_{org} \tag{8}$$

For class C block, only 2 vectors transform is needed in the horizontal transform process, and in vertical transform process, the butterfly computation can be further simplified into Fig. 2. In Fig. 2, the computation represented by dashed line is not necessary for the second row vector transform. So the computation of class C block denoted by DCT_C is 25 add operations, 5 shift operations and 3 compare operations. So

$$DCT_C \approx 41.25\% \cdot DCT_{org} \tag{9}$$

For full DCT block, 2 compare operations are needed besides full block transform operations since fast algorithm is applied. Its computation load is denoted by DCT_{full} and the relation between it and DCT_{org} is as follows:

$$DCT_{full} \approx 102.5\% \cdot DCT_{org} \tag{10}$$

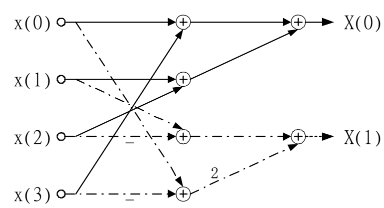

Fig. 2. simplified butterfly computation for C kind block

We define C_{DCT} to represent the efficient of the proposed pruned algorithm on the frame level:

$$C_{DCT} = \frac{CL_{real}}{CL_{total}} \cdot 100\% \tag{11}$$

Where, CL_{total} is the computation load of encoding one frame with original computation of DCT.

$$CL_{total} = DCT_{org} \cdot a \tag{12}$$

CL_{real} is the real computation load after using pruned algorithm. Let λ_A, λ_B, and λ_C represent the percentage of class A, B, and C block among all 4×4 blocks, which can be obtained after encoding one frame, then CL_{real} can be calculated as follow:

$$CL_{real} = DCT_{full} \cdot (1 - \lambda_A - \lambda_B - \lambda_C) \cdot a + DCT_A \cdot \lambda_A \cdot a + DCT_B \cdot \lambda_B \cdot a + DCT_C \cdot \lambda_C \cdot a \tag{13}$$

In equation(12) and (13), a is the number of 4×4 block. Then, (7)-(13) finally yield

$$C_{DCT} = 1.025 - 1.0125\lambda_A - 0.8\lambda_B - 0.6125\lambda_C \tag{14}$$

4. Simulation results

In this section, we evaluate the encoding performance of the proposed algorithm. We make the simulations on JM7.3[11]. Because there is no rate control in JM7.3, we choose QP= 28, 32, 36 and 40 as the comparison point.

Simulation results show that the R-D curves (Fig. 3 and Fig. 4) of JM7.3 with the proposed algorithm and original JM7.3. Stefan is difficult to be encoded and has very high bitrate, so the RD curve for stefan is plotted separately. The RD curves for the five sequences are too closed to be distinguished. This indicates that the proposed pruned DCT algorithm does not lead to encoding performance loss.

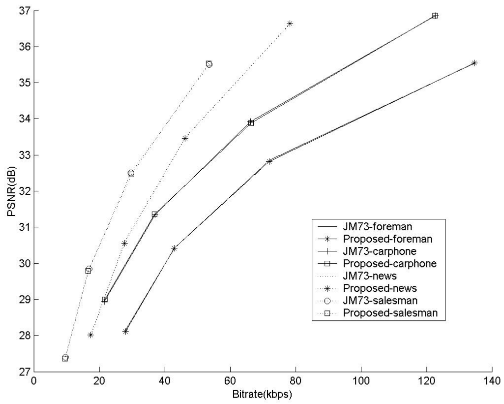

Fig. 3. R-D curves change after the proposed algorithm is applied into JM7.3 (foreman, carphone, news, salesman)

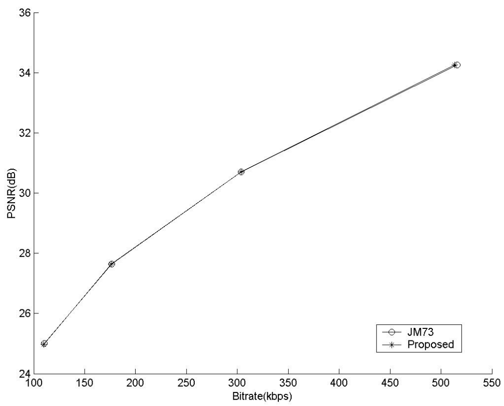

Fig. 4. R-D curves change after the proposed algorithm is applied into JM7.3 (stefan)

In the simulations, λ_A, λ_B and λ_C are recorded and the corresponding C_{DCT} is computed according to equation(14). Table 3 lists the average values of 200 frames for five sequences after the proposed algorithm is applied into JM73. The simulation results show that the proposed adaptive pruned DCT algorithm saves about 55% (1-42.2%) computation compared with original JM73.

We also applied the adaptive algorithm for DCT into the real time encoder on TI DM642 DSP, and relative time analysis shows that the time percentage of DCT, Q, IQ and IDCT is reduced to below 10% from 17%.

5. Conclusion

We propose an adaptive pruned DCT algorithm to reduce the computation of 4×4 transform in H.264. Compared with the original JM, It can save about 55% computation of the transform with negligible encoding performance loss.

Before the proposed pruned algorithm is not used, the computation of DCT, Q, IQ and IDCT accounts for 17% of the total encoding on our H.264 encoder platform based on TI DM642. After the proposed algorithm is applied, the percentage is reduced to below 10%. So the algorithm can effectively reduce the computation of 4×4 transform in H.264.

Table 3. C_{DCT} of JM7.3 with the proposed algorithm (%)

QP		foreman	carphone	stefan	news	salesman
28	λ_A	11.2	26.4	16.3	29.5	8.7
	λ_B	14.7	15.0	18.7	14.1	10.4
	λ_C	29.8	25.1	13.7	22.0	30.9
	C_{DCT}	58.2	45.7	68.0	45.3	63.5
32	λ_A	22.3	34.2	21.6	34.6	13.8
	λ_B	17.5	15.8	7.9	15.0	13.1
	λ_C	28.6	24.9	15.5	23.1	35.2
	C_{DCT}	45.7	37.4	62.4	38.8	53.5
36	λ_A	34.0	44.8	26.9	44.2	21.8
	λ_B	18.0	15.9	8.1	14.1	18.0
	λ_C	28.6	23.1	19.0	22.1	38.1
	C_{DCT}	33.5	27.8	54.7	30.5	39.8
40	λ_A	45.5	54.6	31.6	53.3	31.7
	λ_B	19.6	16.9	9.7	15.0	25.9
	λ_C	24.1	19.1	24.4	18.7	31.2
	C_{DCT}	23.5	19.7	45.3	22.8	27.8
Average		40.225	32.65	57.6	34.35	46.15
						42.2

Acknowledgement

This work is supported by National Nature Science Foundation of China under grant number 60302028.

References

[1] E. Feig, S. Winograd, "Fast algorithms for the discrete cosine transform," IEEE Trans. On Spatial Processing, vol.40, no. 9, pp. 2174-2193Sep. 1992.

[2] Z. Wang, "Pruning the fast discrete cosine transform," IEEE Trans. Commun., vol. 39, no. 5, pp.640-643, May 1991.

[3] A. Skodras, "Fast discrete cosine transform pruning," IEEE Trans. Signal Processing., vol. 43, no. 2, pp. 197-205, May 1995.

0-7695-2650-0/06/$20.00 (c) IEEE

[4] K. Lengwehasatit and A. Ortega, “DCT computation based on variable complexity fast approximations,” ICIP98, Chicago, IL, Oct. 1998.

[5] I.-M. Pao and M.-T. Sun, “Modeling DCT coefficients for fast video encoding,” IEEE Trans. Circuits Syst. Video Technol., vol. 9, pp. 608-616, June 1999.

[6] K. Lengwehasatit and A. Ortega, “Scalable variable complexity approximate forward DCT,” IEEE Trans. Circuits Syst. Video Technol., vol. 14, pp.1236-1248, Nov. 2004.

[7] M.J.Gormish, “Source coding with channel, distortion and complexity constraints,” Ph.D. dissertation, Stanford Univ., CA, 1994.

[8] A.K.Jain, Fundamentals of Digital Image Processing. Englewood Cliffs, NJ: Prentice-Hall,1989.

[9] I. E. G. Richardson, “H.264/MPEG-4 Part 10 white paper: Transform and quantization”,http://www.vcodex.com

[10] A. Hallapuro, M. Karczewicz and H. S. Malvar, “Low-Complexity Transform and Quantization in H.264/AVC,” in Joint Video Team(JVT) of ISO/IEC MPEG and ITU-T VCEG, Jan. 2002 Docs. JVT-B038 and JVT-B039.

[11] JVT reference software JM7.3

0-7695-2650-0/06/$20.00 (c) IEEE