H.263 VIDEO CODEC PERFORMANCE WITH A FAST 8X8 INTEGER IDCT

Joao Tavares, Antonio Silva and Antonio Navarro

Telecommunications Institute - Aveiro University, 3810 Aveiro, Portugal, navarro@av.it.pt

ABSTRACT

Several standardized video coding algorithms use a well known discrete cosine transform (DCT) at the encoder to remove redundancy from video random processes. Its inverse, the IDCT is present at the decoder as well as at the encoder loop. The accuracy of this inverse transform, required to avoid large drift between the encoder and decoder, is defined in an IEEE standard which will be withdrawn from MPEG and ITU standardized codecs. Due to the huge number of computations required to compute the FDCT/IDCT (Forward/Inverse DCT) pair, reduction of their complexity is essential to speed up the video processing. In this paper, we propose a fast integer IDCT calculation method. Additionally, we insert it into the H.263 reference software in order to validate our proposed method. The testing results using two different video sequences, at QCIF and CIF resolutions, show similar PSNR average values between the reference H.263 and the proposed H.263 codec.

1. INTRODUCTION

The H.263 [1] algorithm was designed as a low bitrate solution for video-conferencing and video-telephony applications. It is an hybrid motion compensated algorithm where the DCT is used to remove spatial redundancy. Surely, all DCT based image and video algorithms or standards will benefit from a fast DCT computation. Several floating-point DCT calculation algorithms have been proposed and usually can be classified into two classes: indirect and direct methods. The former computes the DCT through a FFT or other transforms and the latter through matrix factorization or recursive computation.

When direct methods are chosen to calculate (NxN)-point 2-D DCTs, the conventional approach follows the row-column method which requires 2N sets of N-point 1-D DCTs. In [2] and [3], the authors propose two 2-D DCT recursive algorithms based on fast 1-D DCT algorithms of [4] and [5]. However, true 2-D techniques are faster than the conventional row-column approach but demand more hardware resources. A direct 2-D method for the 2-D DCT based on polynomial transform techniques was provided by Duhamel and Guillemot [6]. Feig and Winograd [7] present a matrix factorization algorithm of the 2-D DCT matrix. In [8], Vetterli proposes an indirect method to calculate 2-D DCT by mapping it into a 2-D DFT plus a number of rotations. The 2-D DFT was computed through polynomial transform techniques. In coding applications, the DCT is followed by scaling and quantization, instead of computing the full DCT, we have the advantage in computing a scaled version of it [9]. In this paper, we propose a new H.263 codec where the decoder is multiplierless making it adequate for silicon implementation resulting in power saving and chip area reduction. The integer IDCT proposed in this paper and included in H.263 codec follows the same factorization structure as the DCT proposed in [10] which is based on a floating-point DCT algorithm developed by Feig and Winograd [7] with a complexity close to the theoretical lower bound on the multiplicative complexity presented in [11].

The paper is organized as follows. In Section 2, we briefly present the algorithm described in [7]. In Section 3 we describe our integer IDCT algorithm. Section 4 presents some results in terms of computational complexity and accuracy. Finally conclusion remarks are presented in Section 5.

2. THE FEIG AND WINOGRAD'S 2-D IDCT

As described in the last section, we use one of two algorithms proposed in [7], the scaled version of the IDCT, with a computational complexity of 54 multiplications, 464 additions and 6 shifts.

In [7], Feig and Winograd proposed an algorithm for the factorization of the DCT matrix, which can be represented as a matricial product given by,

$$C_8 = P_8 D_8 R_{81} M_8 R_{82} \tag{1}$$

where D_8 is a diagonal matrix whose diagonal elements are {1; 0.3536; 0.1913; 0.4619; 0.2778; 0.4904; 0.4157; 0.0975}. M_8 is also composed of real values $\gamma(k)=\cos(2\pi k/32)$, k=2,4,6. $R_{82}=B_1B_2B_3$ and P_8 is a permutation matrix.

The computation of the 2-D DCT on 8x8 points involves the product of the matrix,

$$(P_8 D_8 R_{81} M_8 R_{82}) \otimes (P_8 D_8 R_{81} M_8 R_{82}) \tag{2}$$

with a 64-pixel vector X.

Since matrix C is orthogonal its inverse is equal to its transpose. Furthermore, as D is diagonal and M is symmetric, we have,

$$C^{-1} = A_3^T.A_2^T.A_1^T.M.B_2^T.B_1^T.P^T.D \tag{3}$$

The 2-D IDCT can be calculated as

$$\begin{aligned}(C^{-1}\otimes C^{-1})X_{64} &= (A_3^T\otimes A_3^T)(A_2^T\otimes A_2^T)(A_1^T\otimes A_1^T)\\ &(M\otimes M)(B_2^T\otimes B_2^T)(B_1^T\otimes B_1^T)(P^T\otimes P^T)(D\otimes D)X_{64}\end{aligned} \tag{4}$$

and since P is a permutation matrix, (4) can be transformed into

$$\begin{aligned}(C^{-1}\otimes C^{-1})X_{64} &= (A_3^T\otimes A_3^T)(A_2^T\otimes A_2^T)(A_1^T\otimes A_1^T)\\ &(M\otimes M)(B_2^T\otimes B_2^T)(B_1^TP^T\otimes B_1^TP^T)(D\otimes D)X_{64}\end{aligned} \tag{5}$$

where X_{64} is the 64-point vector with DCT coefficients.

The above equation yields an algorithm for the inverse scaled-DCT computation. Multiplication by $D\otimes D$ is a simple pointwise multiplication (which can be incorporated in pre-processing stages), $P^T\otimes P^T$ is a matrix permutation and multiplication of $B_1^T\otimes B_1^T$, $B_2^T\otimes B_2^T$, $A_1^T\otimes A_1^T$, $A_2^T\otimes A_2^T$, $A_3^T\otimes A_3^T$ by X_{64} involves only additions, altogether requires 416 additions. Multiplication by $M\otimes M$ demands for 54 multiplications, 6 shifts and 46 additions. In total, the algorithm requires 54 multiplications, 462 additions and 6 shifts. A more detailed explanation about the computation of the IDCT can be found in [7] and [10].

3. INTEGER 2-D IDCT

Efficient implementation of the IDCT, in terms of computational complexity and power consumption, requires fixed-point (integer) implementations. However, in fixed-point implementation, there is an inherent accuracy problem due to finite word length. Therefore, as the elements of matrix M are real numbers an approximation error will occur.

The multiplications between $M\otimes M$ and $(B_2^T\otimes B_2^T)(B_1^TP^T\otimes B_1^TP^T)X_{64}$ are replaced by a sequence of sums and shifts given by,

$$\begin{aligned}\gamma_2 &= 1-2^{-4}-2^{-6}+2^{-9}+2^{-14}\\ \gamma_4 &= 1-2^{-2}-2^{-5}-2^{-7}-2^{-8}+2^{-14}\\ \gamma_6 &= 2^{-2}+2^{-3}+2^{-7}-2^{-13}\\ \gamma_2+\gamma_6 &= 1+2^{-2}+2^{-4}-2^{-8}+2^{-9}\\ \gamma_2-\gamma_6 &= 2^{-1}+2^{-5}+2^{-7}+2^{-9}+2^{-13}+2^{-14}\\ \gamma_4/2 &= 2^{-2}+2^{-3}-2^{-6}-2^{-8}-2^{-9}+2^{-13}\\ \gamma_4\gamma_6 &= 2^{-2}+2^{-6}+2^{-8}+2^{-10}+2^{-14}\end{aligned} \tag{6}$$

The precision of the approximations (6) satisfy the conditions imposed in [12]. To preserve accurate calculations, we have to perform a scaling operation (multiplication by 2^{15}) to increase the internal precision. According to (5), $D\otimes D$ is the first operation in the calculation chain and our approach is to combine both operations, scaling and multiplication by $D\otimes D$, into one operation denoted as $SD\otimes SD$. Furthermore $SD\otimes SD$ was also approximated by a sequence of additions and shifts. Since multiplication of a 64-point vector by $SD\otimes SD$ is simply a pointwise multiplication, (5) can be transformed into,

$$\begin{aligned}(C^{-1}\otimes C^{-1})X_{64} &= (A_3^T\otimes A_3^T)(A_2^T\otimes A_2^T)(A_1^T\otimes A_1^T)\\ &(M\otimes M)(B_2^T\otimes B_2^T)(B_1^TP^T\otimes B_1^TP^T)\,\|\,X_{8x8}.*SD\,\|\end{aligned} \tag{7}$$

where .* and || denote, respectively, a pointwise multiplication and interlacing operation whereby the columns of the 8x8 matrix are converted into a 64-point vector.

Our algorithm is therefore as follows. Firstly, input data go through a scaling operation which performs the multiplication by SD increasing the internal precision up to 27 bits. Once this operation is performed, data is sent to the pre-addition stage which deals with the multiplication of the 64-point scaled input vector by matrices $(B_2^T\otimes B_2^T)$ and $(B_1^TP^T\otimes B_1^TP^T)$. Then, the following operation is the IDCT core which performs the multiplication of a 64-point vector by matrix $M\otimes M$. The final stage is the post-addition, performing the multiplication by $(A_1^T\otimes A_1^T)$, $(A_2^T\otimes A_2^T)$ and $(A_3^T\otimes A_3^T)$ as well as the de-scaling operation that manages the rounding of the resulting numbers in order to match with the output dynamic range (9 bits). The de-scaling operation consists of a right shift of 15 bits, together with a rounding operation, as follows,

$$descaling(x) = (x+16384) >> 15 \tag{8}$$

The matrix SD is given by,

$$SD = \begin{bmatrix} sd_0 & sd_1 & sd_2 & sd_3 & sd_0 & sd_4 & sd_5 & sd_6 \\ sd_1 & sd_7 & sd_8 & sd_9 & sd_1 & sd_0 & sd_{11} & sd_{12} \\ sd_2 & sd_8 & sd_{13} & sd_{14} & sd_2 & sd_{15} & sd_{16} & sd_{17} \\ sd_3 & sd_9 & sd_{14} & sd_{18} & sd_3 & sd_{19} & sd_{20} & sd_{21} \\ sd_0 & sd_1 & sd_2 & sd_3 & sd_0 & sd_4 & sd_5 & sd_6 \\ sd_4 & sd_{10} & sd_{15} & sd_{19} & sd_4 & sd_{22} & sd_{23} & sd_{24} \\ sd_5 & sd_{11} & sd_{16} & sd_{20} & sd_5 & sd_{23} & sd_{25} & sd_{26} \\ sd_6 & sd_{12} & sd_{17} & sd_{21} & sd_6 & sd_{24} & sd_{26} & sd_{27} \end{bmatrix} \quad (9)$$

with the elements in matrix SD are approximated following the same procedure as in (6), and are given by,

$$\begin{aligned}
&sd_0=2^{12}+1;\\
&sd_1=2^{11}+2^{10}-2^7+2^3+1;\\
&sd_2=2^{11}+2^{10}+2^6-1;\\
&sd_3=2^{12}-2^9-2^7+2^5-2^2;\\
&sd_4=2^{12}+2^{10}+2^7-2^5-2^1;\\
&sd_5=2^{13}-2^9-2^7+2^4+2^1;\\
&sd_6=2^{14}-2^{10}-2^9;\\
&sd_7=2^{11}+2^6+2^4+1;\\
&sd_8=2^{11}+2^7+2^6+2^4+2^2;\\
&sd_9=2^{11}+2^9-2^6+2^4;\\
&sd_{10}=2^{12}-2^8-2^6-2^4-1;\\
&sd_{11}=2^{12}+2^{10}+2^8+2^6+2^4+1;\\
&sd_{12}=2^{13}+2^{11}+2^9-2^6+2^4;\\
&sd_{13}=2^{11}+2^8+2^6+2^5-1;\\
&sd_{14}=2^{11}+2^9+2^6+2^5+2^3+2^1; \qquad (10)\\
&sd_{15}=2^{12}-2^7+2^4+2^2+2^1;\\
&sd_{16}=2^{12}+2^{11}-2^8-2^6-2^5+1;\\
&sd_{17}=2^{13}+2^{11}+2^{10}+2^6+2^5+2^1+1;\\
&sd_{18}=2^{11}+2^{10}-2^7+2^4+2^1+1;\\
&sd_{19}=2^{12}+2^8+2^6+2^4+2^1;\\
&sd_{20}=2^{12}+2^{11}+2^8+2^5+2^2+1;\\
&sd_{21}=2^{13}+2^{12}+2^8+2^6+2^4+2^1+1;\\
&sd_{22}=2^{12}+2^{11}+2^9-2^5+2^3+2^2;\\
&sd_{23}=2^{13}+2^{10}+2^8+2^7+2^5+1;\\
&sd_{24}=2^{14}+2^{11}+2^9-2^6+2^4;\\
&sd_{25}=2^{14}-2^{11}-2^8-2^6-2^5+1;\\
&sd_{26}=2^{15}-2^{12}-2^{10}-2^8+2^5+2^3+1;\\
&sd_{27}=2^{15}+2^{14}+2^{12}+2^9+2^5+2^4+2^2+1.
\end{aligned}$$

All operations are done with 32-bit integer arithmetic.

4. TESTING RESULTS

The computational complexity of the algorithm described in Section 2 is remarkable low, 54 multiplications, 464 additions and 6 shifts making it competitive with other fast IDCTs available in the literature like [13] and [14] with 192 multiplications + 512 additions and 176 multiplications + 464 additions, respectively.

Table 1. Precision results for the proposed IDCT.

Interval	Error type	Precision	
		Sign=+1	Sign=-1
[-5 ,+5]	ppe	1	1
	pme(0.015)	7.00e-004	5.00e-004
	omse(0.02)	4.17e-004	4.06e-004
	pmse(0.06)	9.00e-004	1.00e-003
	me(0.0015)	7.81e-006	2.50e-005
[-256, +255]	ppe	1	1
	pme(0.015)	4.70e-003	3.80e-003
	omse(0.02)	1.73e-002	1.73e-002
	pmse(0.06)	2.27e-002	2.27e-002
	me(0.0015)	2.05e-004	1.67e-004
[-300, +300]	ppe	1	1
	pme(0.015)	4.80e-003	5.20e-003
	omse(0.02)	1.70e-002	1.69e-002
	pmse(0.06)	2.16e-002	2.17e-002
	me(0.0015)	1.75e-004	2.23e-004
[-384, +383]	ppe	1	1
	pme(0.015)	3.70e-003	3.60e-003
	omse(0.02)	1.62e-002	1.62e-002
	pmse(0.06)	2.17e-002	2.18e-002
	me(0.0015)	2.19e-005	7.81e-005
[-512, +511]	ppe	1	1
	pme(0.015)	2.50e-003	2.60e-003
	omse(0.02)	1.63e-002	1.62e-002
	pmse(0.06)	2.21e-002	2.20e-002
	me(0.0015)	2.34e-005	5.00e-005

The main purpose is actually to convert those multiplications into additions as described in Section 3. Thus, concerning our proposed multiplierless algorithm, the accuracy parameters are shown in Table 1, where ppe, pme, omse, pmse and me are mean peak error, peak mean error, overall mean square error, peak mean square error and mean error, respectively. The proposed integer IDCT fulfills thus all precisions defined in IEEE 1180 [12] and in the Call for Proposals recently issued by MPEG [15].

The following step is the replacement of IDCT at the H.263 encoder and decoder. Four different combinations of float and integer IDCT were tested as shown in Tables 2-5. In our setup, the H.263 encoder does not use any optional annexes neither B frames. The simulations were conducted using two test video sequences of 300 frames each, Foreman and Silent, at QCIF and CIF resolutions. The bitrates used were 32, 64, 128 and 256 kbps for QCIF and 128, 256, 512 and 1000 kbps for CIF. Only the first frame was INTRA coded in order to observe the drifting effect. However, it is not recommended to use long refresh periods especially in error prone environments. At the decoder, the mean PSNR for all three color components was measured. For instance, the first row in Table 2 shows the PSNR (dB) results of reference FDCT and IDCT implementations with float values. The second table row presents the PSNR when the decoder reference IDCT is replaced by our IDCT whereas the third line presents the same metric but now when the encoder reference IDCT is replaced by the proposed integer IDCT. Finally, the table bottom row shows the PSNR for the proposed integer implementation of the IDCT at the encoder and at the decoder.

Table 2. Average PSNR obtained for all combinations. Video sequence Foreman QCIF.

	32 kbps	64 kbps	128 kbps	256 kbps
All Float	31.0068	33.0917	35.2959	37.5070
Dec Int_IDCT	31.0064	33.0915	35.2943	37.5041
Enc Int_IDCT	30.9768	33.0751	35.3025	37.5113
Enc+Dec Int_IDCT	30.9771	33.0753	35.3034	37.5133

Table 3. Average PSNR obtained for all combinations. Video sequence Silent QCIF.

	32 kbps	64 kbps	128 kbps	256 kbps
All Float	34.0589	36.4157	39.4628	43.5960
Dec Int_IDCT	34.0558	36.4130	39.4536	43.5655
Enc Int_IDCT	33.9323	36.3541	39.4626	43.1094
Enc+Dec Int_IDCT	33.9314	36.3522	39.4705	43.1369

Table 4. Average PSNR obtained for all combinations. Video sequence Foreman CIF.

	128 kbps	256 kbps	512 kbps	1000 kbps
All Float	32.1251	34.7484	36.9485	39.1405
Dec Int_IDCT	32.1244	34.7466	36.9406	39.1255
Enc Int_IDCT	32.1411	34.7386	36.9646	39.0531
Enc+Dec Int_IDCT	32.1407	34.7381	36.9643	39.0547

Table 5. Average PSNR obtained for all combinations. Video sequence Silent CIF.

	128 kbps	256 kbps	512 kbps	1000 kbps
All Float	35.8330	38.0305	40.7261	44.1824
Dec Int_IDCT	35.8307	38.0249	40.7148	44.1587
Enc Int_IDCT	35.7831	38.0021	40.7657	44.1648
Enc+Dec Int_IDCT	35.7828	38.0013	40.7723	44.1852

By observing Tables 2-5, we realized that the value differences are very small. Therefore, the performance of our modified H.263 is comparable to the original float IDCT based H.263. We also calculated the average PSNR per row using all four tables and the results are respectively, 37.0106, 37.0031, 36.9586 and 36.9625 dB. The lowest PSNR occurred when our proposed algorithm is only placed at the encoder. The maximum drift is 1.3 dB and occurred for Foreman QCIF when the quantization parameter, QP=1.

5. CONCLUSIONS

This paper proposes a fast integer 2D-IDCT algorithm. The computational complexity is remarkable low, 979 additions and 674 shifts making it a very competitive integer IDCT satisfying the IEEE 1180 standard requirements. The PSNR results in H.263 codec are quite comparable in every situation when the IDCT is either the reference or the proposed integer one. The drifting effect is very small on average, 0.05 dB using video sequences with 300 frames. We should mention that the standard forces each macroblock to be coded in INTRA mode at least once every 132 times. The results were obtained by using two video sequences, Foreman and Silent, both at QCIF and CIF resolutions. All decoding operations are calculated in the integer domain. The next step in our research will be the inclusion of an integer version of the FDCT and find out a tradeoff between the FDCT/IDCT precision and the reconstructed video quality.

6. REFERENCES

[1] ITU Telecom. Standardization Sector of ITU, "Video coding for low bitrate communication," *ITU-T Recommendation H.263*, Jan. 2005.

[2] M. A. Haque, "A two dimensional fast cosine transform," IEEE Trans. Acoust., Speech, and Signal Processing, vol. 33, no. 6, pp. 1532-1539, Dec. 1985.

[3] S. C. Chan and K. L. Ho, "A new two-dimensional fast cosine transform algorithm," IEEE Trans. Signal Processing, vol.39, no. 2, pp. 481-485, Feb. 1991.

[4] B. G. Lee, "A new algorithm to compute the discrete cosine transform", IEEE Trans. Acoust. Speech, Signal Processing, vol. 32, pp 1243-1245, Dec. 1984.

[5] H. S. Hou, "A fast recursive algorithm for computing the discrete cosine transform," IEEE Trans. Acoust. Speech. Signal Processing, vol. 35, pp. 1455 -1461, Oct. 1987.

[6] P. Duhamel and C. Guillemot, "Polynomial transform computation of 2-D DCT," Proc. ICASSP'90, pp. 1515 – 1518, 1990.

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

[8] M. Vetterli, "Fast 2-D discrete cosine transform," Proc. IEEE ICASSP, pp. 1538–1541, 1985.

[9] M. Vetterli, P. Duhamel and C. Guillemot, "Trade-offs in the computation of mono- and multi-dimensional DCTs", IEEE Trans. Acoust. Speech, Signal Processing, vol. 2, pp 999-1002, Dec. 1989.

[10] A. Silva and A. Navarro, "Fast 8X8 DCT Pruning Algorithm," Proc. IEEE ICIP, pp. 317–320, Sept. 2005.

[11] E. Feig and S. Winograd, "On the multiplicative complexity of discrete cosine transform," IEEE Trans. Information Theory, vol.38, no. 4, pp. 1387-1391, Jul. 1992.

[12] IEEE, Standard Specifications for the implementations of 8x8 Inverse Discrete Cosine Transform. *IEEE Std* 1180-1990.

[13] C. Loeffler, A. Ligtenberg, and G. S. Moschytz, "Practical fast 1-D DCT algorithms with 11 multiplications", Proc. IEEE Intl. Conf on Acoust., Speech, and Signal Proc. (ICASSP), vol. 2, pp. 988-991, Feb. 1989.

[14] Chen-Wang, "Inverse Two-dimensional DCT algorithm," IEEE ASSP-32, pp. 803-816, Aug. 1984.

[15] ISO/IEC JTC1/SC29/WG11 N7335, Call for Proposals on Fixed-Point 8x8 IDCT and DCT Standard, Poznan, Poland, July 2005.