A FAST DEGRADATION-FREE ALGORITHM FOR DCT BLOCK EXTRACTION IN THE COMPRESSED DOMAIN

Yoshiaki Shibata

Platform Software Development Center,
Sony Corporation,
6-7-35, Kitashinagawa, Shinagawa-ku,
Tokyo, 141-8680, JAPAN
yshibata@sslab.sony.co.jp

Zhigang Chen and Roy H. Campbell

Department of Computer Science,
University of Illinois at Urbana-Champaign,
1304 W.Springfield Av, Urbana, IL, 61801, USA
{*zchen,roy*}*@cs.uiuc.edu*

ABSTRACT

A fast, degradation-free solution for the DCT block extraction problem is proposed. The problem is defined as extracting a DCT block from a DCT compressed frame composed of DCT blocks. This problem is encountered in both video/image manipulations in the compressed domain and transcodecs, for example, converting from MPEG to Motion JPEG. Traditionally, solutions involve using the pixel domain manipulation or Chang's algorithm with approximations. The new solution expands Chang's algorithms, takes full advantage of a fast DCT algorithm, and exploits characteristics of the input DCT blocks without any approximation. The new DCT block extraction achieves 70% performance improvement without any degradation of image quality compared with the conventional solutions.

1. INTRODUCTION

Image and video processing in the discrete cosine transformed (DCT) compressed domain allows direct manipulation of the data without decompression using the inverse discrete cosine transform (IDCT) [1]. Before Chang, *et al.* proposed the technique, image manipulation requires a) decompression of image/video data, b) image/video processing in the pixel domain (IPPD), and c) recompression of the resulting image/video data. Since image/video data is usually both stored on media and transmitted in a compressed form, image/video processing in the compressed domain (IPCD) can improve performance. For example, IPCD shows significant performance improvement for the block based linear pixel manipulation [2, 3] and the inner-block pixel operations [4]. However, total operations for IPCD can overwhelm ones for IPPD if processing needs pixel manipulation over original block boundaries. Specifically, this performance issues constitute a *DCT block extraction problem*:

> Given a 16×16 pixel sized DCT block region built from four DCT blocks $\overline{P}$, $\overline{Q}$, $\overline{R}$, and $\overline{S}$ as shown in Fig. 1, efficiently determine $\overline{H}(v,u)$ (a DCT block arbitrarily located within the region) as a function of the four DCT input blocks and the displacement (v,u).

The solution of this problem is a key procedure for IPCD. Typical situations where the problem is encountered include *compressed domain inverse motion compensation* [3, 5] and *transcoding*, *e.g.*, a transcodec from MPEG to Motion JPEG [6, 7]. While the problem has the traditional IPPD solution, Chang, *et al.* proposed another scheme based on IPCD [1] that operates directly on the input DCT blocks without decompression/recompression. However, as we demonstrate in the next section, Chang's algorithm is computationally more expensive than the IPPD solution with a fast DCT algorithm [8]-[16]. Although other researchers have proposed various approximation techniques for Chang's algorithm to reduce the computational cost [5, 6, 7], these approximations naturally induce degradation of image quality, and are not appropriate especially for high-end use.

In this paper, we propose a new solution named FADEP (Fast Algorithm for Dct block Extraction Problem) for the DCT block extraction problem. Although FADEP is also based on Chang's algorithm, it takes full advantage of a fast DCT algorithm, and exploits characteristics of the input DCT blocks without any approximation, resulting in less complexity than the IPPD solution. In the inverse motion compensation equivalent computation, FADEP achieves more than 70% performance improvement compared with the IPPD solution without any degradation of image quality.

This paper consists of six sections. In Sec. 2, the high complexity of conventional solutions are revealed. In Sec. 3, FADEP is developed with a fast DCT algorithm selection discussed in Sec. 4. The quantitative improvement of FADEP is then demonstrated in Sec. 5, followed by conclusion in Sec. 6

2. CONVENTIONAL SOLUTIONS

The IPPD solution of the problem takes three steps: a) obtain original pixel blocks, P, Q, R and S by decompression, b) extract $H(v,u)$ (pixel block), and c) convert $H(v,u)$ into a corresponding DCT block, $\overline{H}(v,u)$. The complexity for this solution is analyzed as follows: Let T be the DCT operation matrix whose elements, $t_{k,i}$, are given by

$$t_{k,i} = \frac{c(k)}{2}\cos\frac{(2i+1)k\pi}{16} \quad \text{for } k,i = 0,\cdots,7 \qquad (1)$$

where $c(k) = 1/\sqrt{2}$ for $k = 0$ and 1 otherwise. The 2-dimensional (2-D) DCT of H can be written as

$$\overline{H} = \text{DCT}(H) = THT^t, \qquad (2)$$

where T^t denotes a transposed matrix of T. Note that the unitary property, $TT^t = T^tT = I$ where I is an identity matrix, holds for the DCT operation matrix given by Eq. (1). Now, let the computational complexity for 8×8 matrix multiplication be ξ_{MM}, then the complexity for Eq. (2) becomes $2\xi_{MM}$. Since the complexities of the respective IDCTs for $\overline{P}, \cdots, \overline{S}$ are also given by $2\xi_{MM}$, the total complexity of this solution is estimated as $10\xi_{MM}$.

On the other hand, Chang,*et al.* proposed a solution of the problem based on IPCP [1]. According to their algorithm, $\overline{H}(v,u)$ is given by

$$\begin{aligned}\overline{H}(v,u) &= \overline{W}_1^t(v)\overline{P}\,\overline{W}_1(u) + \overline{W}_1^t(v)\overline{Q}\,\overline{W}_0(u) + \\ &\quad \overline{W}_0^t(v)\overline{R}\,\overline{W}_1(u) + \overline{W}_0^t(v)\overline{S}\,\overline{W}_0(u) \qquad (3)\end{aligned}$$

where

$$\begin{cases} \overline{W}_0(u) = TW_0(u)T^t \\ \overline{W}_1(u) = TW_1(u)T^t \end{cases} \qquad (4)$$

and

$$W_0(u) = \begin{pmatrix} 0 & I_u \\ 0 & 0 \end{pmatrix}, \quad W_1(u) = \begin{pmatrix} 0 & 0 \\ I_{8-u} & 0 \end{pmatrix}. \qquad (5)$$

Although Equation (3) has the complexity of $8\xi_{MM}$, decomposing Eq. (3) into

$$\begin{aligned}\overline{F}_0 &= \overline{P}\,\overline{W}_1(u) + \overline{Q}\,\overline{W}_0(u), && (6)\\ \overline{F}_1 &= \overline{R}\,\overline{W}_1(u) + \overline{S}\,\overline{W}_0(u), \text{ and} && (7)\\ \overline{H}(v,u) &= \overline{W}_1^t(v)\overline{F}_0 + \overline{W}_0^t(v)\overline{F}_1 && (8)\end{aligned}$$

can save two matrix multiplications, resulting in $6\xi_{MM}$ as its complexity.

According to the above analysis, the Chang's algorithm's solution can reduce the computational complexity to 60% of the one based on IPPD. However, the IPPD solution performs more efficiently if a fast DCT algorithm is introduced. If TH or HT^t in Eq. (2) is computed using matrix multiplication, the multiplicative complexity is calculated as $64 \times 8 = 512$. On the other hand, when Chen's fast DCT algorithm is applied to the computation for example, the complexity becomes $16 \times 8 = 128$, because their algorithm requires only 16 multiplications for a 1-D pixel vector [9]. Thus, letting the complexity of TH or HT^t using a fast DCT algorithm be ξ_{DCT}, we obtain the approximation $\xi_{DCT} \simeq \frac{1}{4}\xi_{MM}$ which gives the complexity of the IPPD solution as $10\xi_{DCT} \simeq 2.5\xi_{MM}$. This suggests that the IPPD solution performs more than twice as efficiently as that based on Chang's algorithm.

3. NEW SOLUTION (FADEP)

2-D DCT blocks have a separable property that allows independent handling of their row and column vectors. The row-column approach together with a sub-block extraction needs less IDCT operation than the IPPD solution. This is the basic idea of FADEP.

According to Eqs. (4), Equation (6) can be expanded to

$$\overline{F}_0 = \overline{P}\,TW_1(u)T^t + \overline{Q}\,TW_0(u)T^t = \tilde{F}_0T^t \qquad (9)$$

where

$$\tilde{F}_0 = \overline{P}\,TW_1(u) + \overline{Q}\,TW_0(u). \qquad (10)$$

Similarly, we can obtain from Eq. (7)

$$\overline{F}_1 = \tilde{F}_1T^t, \qquad (11)$$

where

$$\tilde{F}_1 = \overline{R}\,TW_1(u) + \overline{S}\,TW_0(u). \qquad (12)$$

Substituting Eqs. (9) and (11) into Eq. (8) after the decomposition of $\overline{W}_0^t(v)$ and $\overline{W}_1^t(v)$ gives

$$\overline{H}(v,u) = TW_1^t(v)T^t\overline{F}_0 + TW_0^t(v)T^t\overline{F}_1 = TGT^t \qquad (13)$$

where

$$G = W_1^t(v)T^t\tilde{F}_0 + W_0^t(v)T^t\tilde{F}_1. \qquad (14)$$

Based on these modifications of Chang's algorithm, FADEP is derived as Eqs. (10), (12), (13), and (14). The complexity of FADEP is less than the IPPD solution: since the complexity of each of the equation is $2\xi_{DCT}$, FADEP's complexity is their sum $8\xi_{DCT}$, $2\xi_{DCT}$ less than that of the IPPD solution.

Although FADEP performs more efficiently than the IPPD solutions, the improvement is not significant. However two techniques, *pruned IDCT* and *scaled DCT*, can further enhance FADEP performance. The former is applied to the partial IDCT component or Eqs. (10), (12), and (14), while the latter to the backward DCT component or Eq. (13). The pruned IDCT considers only a limited number of low frequency elements as input, and is based on *DCT pruning* [16]. Since a number of input elements should be specified in the pruned IDCT, additional quantities, rowL and clmL, are introduced in the DCT block data structure. They represent the lengths of row and column vectors in the block, respectively, and are given by the maximum index of non-zero elements in the vectors.

4. SELECTION OF FAST DCT ALGORITHMS

Several existing fast DCT algorithms [9]-[15] can implement the scaled DCT and pruned IDCT required for FADEP's backward DCT and partial IDCT components. However, a flowgraph analysis reveals that particular algorithms are better than others.

The flowgraphs of the algorithms consist of three parts: a) a 1st stage butterfly, b) 2nd and further stages for even DCT coefficients (Even part),and c) 2nd and further stages for odd DCT coefficients (Odd part). If μ and α are the numbers of multiplications and additions, respectively, the complexity of the algorithms when they are applied to the scaled DCT is shown in Table 1. This table suggests that Arai's algorithm has the least complexity and is suitable for the backward DCT component of FADEP

The flowgraph of the IDCT can be obtained by reversing the direction of the DCT flowgraph. The pruned IDCT can thus be obtained by pruning higher frequency input in the flowgraphs. The number of operations (μ, α) is summarized in Table 2 as a function of a number of frequency input (arguments) for each IDCT algorithm. This table clearly indicates that among the pruned IDCT algorithms investigated, the pruned IDCT algorithm based on Hou's DCT algorithm [11] gives the least number of operations for all numbers of input. Therefore, their algorithm is suitable for the partial IDCT component of FADEP.

Combining the two fast DCT algorithms for FADEP, the resulting flowgraph for a 1-D input vector is illustrated in Fig. 2. In this figure, solid lines denote positive flows while dotted lines represent flows negatively added (multiplied by -1), and values are multiplied by the weight coefficients $(C_i, \cdots, E_i)$ at the dotted points (●). Note that FADEP consists of three components: the partial IDCT, the backward DCT, and the final scaling. The final scaling component includes all of the scaling operations used in FADEP, and can be combined with the following quantizing stage.

5. DEMONSTRATIONS

In order to demonstrate the quantitative improvement of FADEP over the IPPD solution, we implemented the DCT block extraction module based on FADEP as well as the IPPD solution. The performance is evaluated by measuring the CPU time of the DCT block extraction module on a Sun Ultra SPARC machine running SunOS 5.6.

One of the important characteristics of FADEP is that it works more efficiently when the input DCT blocks are sparse. In order to demonstrate this characteristic, the performance of the module for artificially designed input DCT blocks was evaluated. Using a test DCT block full of non-zero elements, the evaluation used modified DCT blocks that are created by zeroing all of the elements whose indices in the zigzag sequence is larger than a specified value. This value, *zigzag length*, characterizes the sparseness of the input DCT block. When the zigzag length is small, modified DCT blocks have non-zero elements at their upper-left corner as shown by the inner figure of Fig. 3. The figure shows the performance of FADEP for DCT blocks as a function of the zigzag length. The performance of the IPPD solution is independent of the sparseness of the input DCT blocks. The FADEP performance is thus provided in the form of the ratio of the CPU time for the IPPD solution to that of FADEP. As shown in Fig. 3, FADEP performance improves as the zigzag length decreases. FADEP also shows 18% performance improvement even when all the elements in the input DCT block are non-zero (the zigzag length at 64). This is explained by the complexity analysis of Sec. 3. Figure 4 shows the performance of FADEP for the input DCT blocks composed of a certain number of row or column vectors (see inner figure). Because the FADEP module has symmetric structure in terms of the row and column vectors, the figure shows no performance discrepancy between the number of vectors.

Finally, we evaluated FADEP with actual image data. The original images of various image qualities were provided in JPEG format. The image data was Huffman/Run-length decoded, reordered from the zigzag sequence to an 8×8 size DCT block, and dequantized to form a *DCT block frame*. The evaluation used only the luminance (Y) component for the DCT block frames. Then, a DCT block arbitrarily located within the DCT block frames was extracted using both the FADEP and the IPPD solution based modules. (Note that this procedure is regarded as a simplified form of the inverse motion compensation.) Table 3 summarizes the experimental results as well as some statistical properties of the input DCT block frames. As shown in Table 3, the performance of the FADEP based module is improved more than 1.7 times over the IPPD based module for example images. It is also seen from Table 3 that even for a image of very high quality, where most of DCT elements are non-zero, a 15% performance improvement is obtained. The FADEP extracted DCT block is equivalent to the one extracted based on the IPPD solution, *i.e.*, the FADEP based module induces no degradation of the image quality. Moreover, FADEP is also applicable to constant bit rate applications in which the bit rate of the output would be controlled at the expense of degradation. As shown in Fig. 2, the final scaling stage is separable from the total solution, and can be merged with the following quantization stage that dynamically controls the bit rate of the resulting video stream.

6. CONCLUSIONS

In this paper, we propose a new solution (FADEP) for the DCT block extraction problem. Although FADEP is developed based on Chang's algorithms, it not only takes full advantage of a fast DCT algorithm but also exploits characteristics of the input DCT blocks without any approximation. The DCT block extraction module based on FADEP achieves more than 70% performance improvement without any degradation of image quality for the motion compensation equivalent computation.

7. REFERENCES

[1] S. -F. Chang, W. -L. Chen and D. G. Messerschmitt, "Video Compositing in the DCT Domain", IEEE Workshop on Visual Signal Process. and Commun., Rayleigh, North Carolina, pp.(044)1-6, 1992

[2] B. C. Smith and L. A. Rowe, "Algorithms for Manipulating Compressed Images", IEEE Computer Graphics and Appli, Vol.13, pp.34-42, 1993

[3] S. -F. Chang and D. G. Messerschmitt, "Manipulation and Compositing of MC-DCT Compressed Video", IEEE J. on Selected Areas in Commun., Vol.13, pp.1-11, 1995

[4] B. Shen and I. K. Sethi, "Inner-Block Operations On Compressed Images", Proc. of the 3rd ACM Int'l Conf. on Multimedia, pp.489-498, 1995

[5] N. Merhav and V. Bhaskaran, "Fast Inverse Motion Compensation Algorithms for MPEG and for Partial DCT Information", J. Visual Commun. and Image Represent., Vol.7, pp.395-410, 1996

[6] P. A. A. Assuncao and M. Ghanbari, "Transcoding of MPEG-2 Video in the Frequency Domain", Proc. ICASSP'97, pp.2633-2636, 1997

[7] S. Acharya and B. C. Smith, "Compressed Domain Transcoding of MPEG", Proc. IEEE Multimedia Systems '98, pp.295-304, 1998

[8] Y. Arai, T. Agui and M. Nakajima, "A Fast DCT-SQ Scheme for Images", Trans. IEICE, vol.E 71, pp.1095-1097, 1988

[9] W. -H. Chen, C. H. Smith and S. C. Fralick "A Fast Computational Algorithm for the Discrete Cosine Transform", IEEE Trans. Commun., vol.COM-25, pp.1004-1009, 1977

[10] E. Feig and S. Winograd, "Fast Algorithms for the Discrete Cosine Transform", IEEE Trans. Signal Process., vol.40, pp.2174-2193, 1992

[11] H. S. Hou, "A Fast Recursive Algorithm For Computing the Discrete Cosine Transform", IEEE Trans. ASSP, vol.ASSP-35, pp.1455-1461, 1987

[12] B. G. Lee, "A New Algorithm to Compute the Discrete Cosine Transform", IEEE Trans. ASSP, vol.ASSP-32, pp.1243-1245, 1984

[13] C. Loeffler, A. Ligtenberg and G. S. Moschytz, "Practical Fast 1-D DCT Algorithms with 11 Multiplications", Proc. ICASSP'89, pp.988-991, 1989

[14] N. Suehiro and M. Hatori, "Fast Algorithms for the DFT and Other Sinusoidal Transforms", IEEE Trans. ASSP, vol.ASSP-34, pp.642-644, 1986

[15] M. Vetterli and A. Ligtenberg, "A Discrete Fourier-Cosine Transform Chip", IEEE J. on Selected Areas in Commun., vol.SAC-4, pp.49-61, 1986

[16] Z. Wang, "Pruning the Fast Discrete Cosine Transform", IEEE Trans. Commun., vol.39, pp.640-643, 1991

	Even part		Odd part		Total*	
Algorithm	μ	α	μ	α	μ	α
Arai [8]	1	9	4	12	5	29
Chen [9]	2	9	6	12	8	29
Feig [10]	2	9	5	12	7	29
Hou [11]	3	9	8	12	11	29
Lee [12]	2	9	8	12	10	29
Loeffler [13]	2	9	6	12	8	29
Suehiro [14]	2	9	6	12	8	29
Vetterli [15]	2	9	6	12	8	29
Wang [16]	3	9	8	12	11	29

Table 1: Comparison of the complexity of scaled DCT for several fast DCT algorithms. μ and α are numbers of multiplications and additions used in the algorithm, respectively. Note that the total number of additions $(*)$ includes a further eight additions from the first stage.

	Arg 8		Arg 7		Arg 6		Arg 5		Arg 4	
Algorithm	μ	α	μ	α	μ	α	μ	α	μ	α
Arai	13	29	12	27	11	24	10	22	9	21
Chen	13	29	12	26	11	23	10	20	9	18
Feig	14	29	13	27	12	24	11	22	10	20
Hou	13	29	12	25	11	22	10	19	9	17
Lee	13	29	13	28	12	25	12	23	11	21
Loeffler	13	29	13	27	12	24	11	22	10	20
Suehiro	13	29	12	26	11	23	10	20	9	18
Vetterli	13	29	12	26	11	23	10	20	9	18
Wang	13	29	12	26	11	23	10	20	9	18

Table 2: Comparison of the complexity of pruned IDCT for several fast DCT algorithms. "Arg i" represents the pruned IDCT with the lowest i DCT coefficients as input. Note that all algorithms have the same complexity for $i < 4$.

file(.jpg)	CmpRatio	$\langle zzL \rangle$	$\langle rowL \rangle$	$\langle clmL \rangle$	perform
test0	19.2	16.6	1.0	1.1	1.76
test1	11.3	17.2	1.5	1.5	1.63
test2	6.4	24.9	2.3	2.3	1.44
test3	5.4	45.1	4.7	4.7	1.15

Table 3: Relative performance of FADEP together with statistical properties of input image data. "CmpRatio " denotes the compression ratio of original JPEG file. $\langle zzL \rangle$, $\langle RowL \rangle$, and $\langle ClmL \rangle$ represent the average values of the zigzag length, rowL, and clmL of the DCT block over the entire frame, respectively.

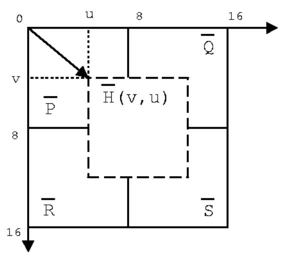
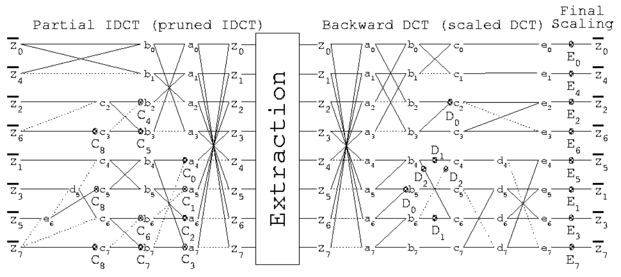


Figure 1: DCT block extraction in the compressed domain.



$C_0 = 0.5\sin(\pi/16)$, $C_1 = 0.5\sin(3\pi/16)$, $C_2 = 0.5\cos(3\pi/16)$,
$C_3 = 0.5\cos(\pi/16)$, $C_4 = 0.5\sin(\pi/8)$, $C_5 = 0.5\cos(\pi/8)$,
$C_6 = 2\sin(\pi/8)$, $C_7 = 2\cos(\pi/8)$, $C_8 = 2\sin(\pi/4)$
$D_0 = \cos(\pi/4)$, $D_1 = \cos(\pi/8)$, $D_2 = \cos(3\pi/8)$
$E_0 = 1/8$, $E_i = \sqrt{2}/16\cos(i\pi/16)$ for $i = 1, \cdots, 7$

Figure 2: Flowgraph of FADEP for a 1-D input vector.

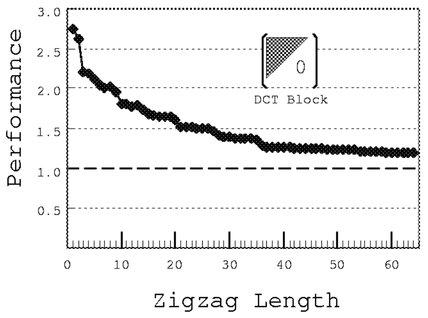


Figure 3: Relative performance of FADEP as a function of the zigzag length in the input DCT blocks.

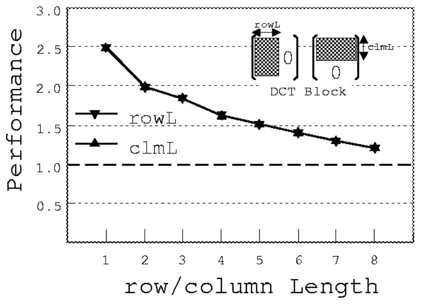


Figure 4: Relative performance of FADEP as a function of the number of row/column vectors in the input DCT blocks.