FAST ALGORITHMS FOR THE RECURSIVE COMPUTATION OF TWO-DIMENSIONAL DISCRETE COSINE TRANSFORM *

Wen–Hsien Fang, Neng–Chung Hu and Shih–Kuo Shih
Dept. of Electronic Engineering, National Taiwan University of Science and Technology,
Taipei, Taiwan, R.O.C.
Tel: 886-2-27376412, Fax: 886-2-27376424
e-mail: whf@et.ntust.edu.tw

ABSTRACT

This paper presents an efficient algorithm for computing the two-dimensional discrete cosine transform (2-D DCT) of size $p^r \times p^r$, where p is a prime. The algorithm decomposes the 2-D DCT outputs into three parts: the first part contains outputs whose indices are both multiples of p and forms a 2-D DCT of size $p^{r-1} \times p^{r-1}$, whereas the remaining outputs are further decomposed into two parts, depending on the summation of their indices. The latter two parts can be reformulated as a set of circular correlation (CC) or skew-circular correlation (SCC) matrix-vector products. Such a decomposition procedure can be repetitively carried out, resulting in a sequence of CC and SCC matrix-vector products. Employing fast algorithms for these CC/SCC operations, we can thus obtain algorithms with minimum multiplicative complexity.

1 INTRODUCTION

The DCT has found applications in various facets of signal processing [1]. The 2-D DCT has been, in particular, widely regarded to be the most effective scheme for transform coding. To facilitate real-time implementations, the development of efficient algorithms has been of great interest over the past few decades.

The conventional approach for computing the 2-D DCT is based on the row-column decomposition (RCD) method. More efficient algorithms can in general be attained by working directly on the 2-D data by fully exploiting the 2-D characteristics of the kernel functions. For example, Cho and Lee addressed a fast algorithm which decomposes an $N \times N$ DCT into N 1-D N-point DCT's [2]. Duhamel and Guillemot considered an efficient polynomial transform-based approach [3, 4]. Despite their efficiency, these fast algorithms, however, are only applicable to the 2-D DCT of size $2^r \times 2^r$.

In this paper, we address a new fast algorithm for the recursive computation of the 2-D DCT of size $p^r \times p^r$, where p is a prime. Based on the 2-D to 1-D index mapping scheme addressed in [5], the proposed algorithm decomposes the 2-D DCT output components into three parts. The first part contains outputs whose indices are both multiples of p and forms a $p^{r-1} \times p^{r-1}$ DCT. The remaining output components are further decomposed into two parts, depending on the summation of their indices. The latter two parts can be reformulated as a set of CC or SCC matrix-vector products by utilizing the recently proposed maximum coset decomposition [6]. Such a decomposition procedure can be repetitively carried out for the 2-D DCT of the first part by following the same steps. As a consequence, the computation of the 2-D DCT is converted into a sequence of CC and SCC matrix-vector products of various sizes. Employing fast algorithms for the computation of these CC/SCC operations. we can thus substantially reduce the numbers of multiplication required. In the special case when $p = 2$, the proposed algorithm can be further simplified, calling for computations comparable to those of previous works.

*This research is supported by National Science Council of R.O.C. under contract NSC87-2213-E-011-014.

2 PROPOSED FAST ALGORITHMS

For a set of 2-D data $\{x(k_1, k_2); 0 \le k_1 \le N-1, 0 \le k_2 \le N-1\}$, the 2-D DCT is defined as

$$X(n_1, n_2) = \sum_{k_1=0}^{N-1} \sum_{k_2=0}^{N-1} x(k_1, k_2) \cos(\frac{2\pi(2k_1+1)n_1}{4N}) \cdot \cos(\frac{2\pi(2k_2+1)n_2}{4N}), \quad 0 \le n_1, n_2, \le N-1 \quad (1)$$

where $N = p^r$ and p is a prime. Invoking the trigonometric identity of $\cos(\alpha)\cos(\beta) = \frac{1}{2}[\cos(\alpha+\beta)+\cos(\alpha-\beta)]$, we can rewrite (1) as

$$X(n_1, n_2) = \frac{1}{2}\left[X^+(n_1, n_2) + X^-(n_1, n_2)\right] \quad (2)$$

where

$$X^{\pm}(n_1, n_2) = \sum_{k_1=0}^{N-1} \sum_{k_2=0}^{N-1} x(k_1, k_2) \cdot \cos(\frac{2\pi[(2k_1+1)n_1 + (2k_2+1)n_2]}{4N}) \quad (3)$$

Using the 2-D to 1-D index mapping equations of [5]:

$$(2k_1+1)n_1 \pm (2k_2+1)n_2 \equiv (2n+1)2k \mod 4N$$
$$\text{if } n_1+n_2 \text{ is even} \quad (4)$$
$$(2k_1+1)n_1 \pm (2k_2+1)n_2 \equiv (2n+1)(2k+1) \mod 4N$$
$$\text{otherwise} \quad (5)$$

$X^{\pm}(n_1,n_2)$ can then be evaluated via the following 1-D expressions

$X^{\pm}(n_1,n_2) \triangleq Y^{\pm}(n) =$

$$\begin{cases} \sum_{k=0}^{\frac{N+1}{2}-1} y^{\pm}(k)\cos(\frac{2\pi(2n+1)2k}{4N}) \\ \qquad \text{if } n_1+n_2 \text{ is even} \\ \sum_{k=0}^{\frac{N-1}{2}-1} y^{\pm}(k)\cos(\frac{2\pi(2n+1)(2k+1)}{4N}) \\ \qquad \text{otherwise} \end{cases} \quad (6)$$

where $y^{\pm}(k) = \sum_{k_1}\sum_{k_2} x(k_1,k_2)$ in which (k_1,k_2) satisfies the index mapping equations of (4) or (5), depending on whether n_1+n_2 is even or odd.

Thereby, $X(n_1,n_2)$ can now be rewritten as

$$X(n_1,n_2) \triangleq Y(n) = \begin{cases} \sum_{k=0}^{\frac{N+1}{2}-1} y(k)\cos(\frac{2\pi(2n+1)2k}{4N}) \\ \qquad \text{if } n_1+n_2 \text{ is even} \quad (7a) \\ \sum_{k=0}^{\frac{N}{2}-1} y(k)\cos(\frac{2\pi(2n+1)(2k+1)}{4N}) \\ \qquad \text{otherwise} \quad (7b) \end{cases}$$

where $Y(n) \triangleq \frac{1}{2}\big[Y^{+}(n)+Y^{-}(n)\big]$ and $y(k) \triangleq \frac{1}{2}\big[y^{+}(k)+y^{-}(k)\big]$. Next, we address the issue of solving the 2-D to 1-D index mapping equations of (4) and (5). To solve these, n_1, n_2 and n must be decided first. In other words, the index (n_1,n_2) of the 2-D output component is first mapped to a predetermined 1-D output component with an index n, denoted by $(n_1,n_2) \to n$. According to this $(n_1,n_2) \to n$, a solution set of (k_1,k_2) on the left-hand side of (4) and (5) can thus be obtained for every specific k on the right-hand sides of (4) and (5). $y(k)$ can then be determined by a summation of $x(k_1,k_2)$ of these corresponding (k_1,k_2)'s.

To mitigate the overhead in the computation of $y(k)$, we consider those $\{X(n_1,n_2)\}$ which share the same set of $\{y(k)\}$ together. Since all of the corresponding (n_1,n_2)'s satisfy the congruence equations of (4) and (5), they are related to each other by some factors. Therefore, if one of these indices, say (n_1^0,n_2^0), $\to 0$, then the other indices (n_1,n_2)'s inside this set can be determined as a function of (n_1^0,n_2^0) and n. Such a particular (n_1^0,n_2^0) is thus referred to as a dominant point for the corresponding set of $\{X(n_1,n_2)\}$. It can be readily shown that the rest of $\{X(n_1,n_2)\}$, which share the same set of $\{y(k)\}$, are related to $Y(n)$ by [7]

$$Y(n) = X(<(2n+1)n_1^0>, <(2n+1)n_2^0>) \quad (8)$$

where $<\cdot>$ stands for either mod $4N$ or mod $2N$, depending on whether (4) or (5) is employed.

Some output components, however, appear repetitively for different dominant points if $(2n+1)$ is a multiple of p [7]. To avoid this,n that satisfies $(2n+1)=mp$ will be treated separately. Based on this fact along with the index mapping equations of (4) and (5), we can classify the computation of the 2-D DCT into the following three categories:

Part I: $\{X(n_1,n_2) : n_1 \bmod p = n_2 \bmod p = 0\}$
Part II: $\{X(n_1,n_2) : n_1 \bmod p \neq 0$ or $n_2 \bmod p \neq 0$, n_1+n_2 is even $\}$
Part III: $\{X(n_1,n_2) : n_1 \bmod p \neq 0$ or $n_2 \bmod p \neq 0$, n_1+n_2 is odd $\}$

For the output components of Part II, by using the maximum coset decomposition as addressed in [6], the corresponding $2n+1$ ($2n+1 \neq mp$) can be obtained by a generator which is an odd prime. More specifically, such a set can be generated by a power of g mod $2N$ as $G_1 = \{g^0, g^1, \cdots, g^{\lambda-1}\}$ mod $2N$, where g is the coset generator with order λ which satisfies $\cos(\frac{2\pi}{4N}) = \pm\cos(\frac{2\pi g^r}{4N})$. Therefore, for every dominant point, there are only λ (nonredundant) corresponding $Y(n)$'s (or, equivalently, $X(n_1,n_2)$'s), where $\lambda = p^{r-1}\frac{(p-1)}{2}$ if $N=p^r$. The $\frac{N+1}{2}$ input points in (7a) can be divided into the following cosets: G_1, $G_2 = pG_1 = \{p, pg, \cdots, pg^{\lambda/p-1}\}$ mod $2N$, $\cdots$, $G_r = p^{r-1}G_1$ mod $2N$, and $G_{r+1} = \{N\}$. The corresponding orders of these cosets are λ, $\frac{\lambda}{p}$, $\cdots$, and 1, respectively.

By replacing $(2n+1)$ and k with g^m and $p^i g^l$, respectively, in (7a), we can rewrite (7a) with $\frac{N+1}{2}$ inputs and λ outputs as

$$Y(<\frac{g^m-1}{2}>) = \sum_{i=0}^{r}\sum_{l=0}^{\lambda/p^i-1} y(<p^i g^l>) \cdot \cos(\frac{2\pi p^i g^{l+m}}{2N}) \quad (9)$$

where $m = 0, 1, \cdots, \lambda-1$ and $<\cdot>$ denotes mod $2N$ operation. Expanding (9) yields the following more illustrative matrix notation:

$$\begin{bmatrix} Y(<\frac{g^0-1}{2}>) \\ Y(<\frac{g^1-1}{2}>) \\ Y(<\frac{g^2-1}{2}>) \\ \vdots \\ Y(<\frac{g^{\lambda-1}-1}{2}>) \end{bmatrix} = \sum_{i=0}^{r} \begin{bmatrix} C_{p^i g^0} & \cdots & C_{p^i g^{\lambda/p^i-1}} \\ C_{p^i g^1} & \cdots & C_{p^i g^0} \\ \vdots & \ddots & \vdots \\ C_{p^i g^{\lambda-1}} & \cdots & C_{p^i g^{\lambda-2}} \end{bmatrix} \cdot \begin{bmatrix} y(<p^i g^0>) \\ y(<p^i g^1>) \\ y(<p^i g^2>) \\ \vdots \\ y(<p^i g^{\lambda-1}>) \end{bmatrix} \quad (10)$$

where $C_k \triangleq \cos(\frac{2\pi k}{2N})$ and we have used the fact that $C_{p^i g^{\lambda/p^i+m}} = C_{p^i g^m}$, $0 \le i \le r$, $0 \le m \le \lambda-1$. The

computations of the 2-D DCT output components in Part II can be thus implemented by a sequence of CC matrix-vector products of sizes $\frac{\lambda}{p^i}$, $i = 0, \cdots, r$.

Along the same line, the computation of the output components of PART III can be simplified by following the above steps, except that the operation of mod $2N$ is replaced with mod $4N$. By replacing $(2n+1)$ and $(2k+1)$ with g^m and $p^i g^l$, respectively, in (7b), we can rewrite (7b) with $\frac{N-1}{2}$ inputs and λ outputs as

$$Y(< \frac{g^m - 1}{2} >) = \sum_{i=0}^{r-1} \sum_{k=0}^{\frac{\lambda}{p^i}-1} y(\frac{< p^i g^k > -1}{2}) \cdot \cos(\frac{2\pi p^i g^{l+m}}{4N}) \quad (11)$$

where $m = 0, \cdots, \lambda - 1$ and $< \cdot >$ denotes mod $4N$ operation. As the above, we can also express (11) as the following matrix notation:

$$\begin{bmatrix} Y(< \frac{g^0-1}{2} >) \\ Y(< \frac{g^1-1}{2} >) \\ Y(< \frac{g^2-1}{2} >) \\ \vdots \\ Y(< \frac{g^{\lambda-1}-1}{2} >) \end{bmatrix} = \sum_{i=0}^{r-1} \begin{bmatrix} \bar{C}_{p^i g^0} & \cdots & \bar{C}_{p^i g^{\lambda/p^i-1}} \\ \bar{C}_{p^i g^1} & \cdots & -\bar{C}_{p^i g^0} \\ \vdots & \ddots & \vdots \\ \bar{C}_{p^i g^{\lambda-1}} & \cdots & -\bar{C}_{p^i g^{\lambda-2}} \end{bmatrix} \cdot \begin{bmatrix} y(\frac{<p^i g^0>-1}{2}) \\ y(\frac{<p^i g^1>-1}{2}) \\ y(\frac{<p^i g^2>-1}{2}) \\ \vdots \\ y(< \frac{p^i g^{\lambda-1}>-1}{2}) \end{bmatrix} \quad (12)$$

where $\bar{C}_k \triangleq \cos(\frac{2\pi k}{4N})$ and we have used the fact that $\bar{C}_{p^i g^{\lambda/p^i+m}} = -\bar{C}_{p^i g^m}$, $\leq 0i \leq r$, $0 \leq m \leq \lambda - 1$. The computations of the 2-D DCT outputs in Part III can be thus implemented by a sequence of SCC matrix-vector products of sizes $\frac{\lambda}{p^i}$, $i = 0, \cdots, r$.

The above decomposition procedure can be recursively carried out for the 2-D DCT of PART I. The overall steps can be summarized into the following algorithm (with initial value $i = 1$):

Step 1. Divide the output components into three parts. If both of their indices are multiples of p (Part I), these $\{X(n_1, n_2)\}$ form a $p^{r-i} \times p^{r-i}$ 2-D DCT. The other output components are further classified into two parts, Parts II and III, depending on the summation of $\frac{n_1}{p^{i-1}} + \frac{n_2}{p^{i-1}}$ is even or odd.

Step 2. The $\{Y(n)\}$ in Parts II and III are divided into groups with different choices of dominant points. Every group of $\{Y(n)\}$ in Parts II and III can be implemented via the CC operation of (10) and the SCC operation of (12), respectively. The resulting $\{X(n_1, n_2)\}$ can be obtained by the index mapping scheme of (8). The $p^{r-i} \times p^{r-i}$ DCT of Part I can be further decomposed along the same steps by going back to Step 1 with $i = i + 1$ until $i = r$.

As a special case when $p = 2$, (7a) and (7b) can be simplified as

$$X(n_1, n_2) = \begin{cases} \sum_{k=0}^{\frac{N+1}{2}-1} y(k) \cos(\frac{2\pi(2n+1)2k}{4N}) \\ \quad \text{if } n_1 + n_2 \text{ is even} \\ \sum_{k=0}^{\frac{N}{2}-1} y(k) \cos(\frac{2\pi(2n+1)(2k+1)}{4N}) \\ \quad \text{otherwise} \end{cases} \quad (13)$$

Employing the maximum coset decomposition as discussed above, we can obtain

$$Y(< \frac{g^m - 1}{2} >) = \sum_{i=0}^{r} \sum_{l=0}^{\lambda/2^i-1} y(< 2^i g^m >) \cdot \cos(\frac{2\pi 2^i g^{m+l}}{2N}) \quad \text{if } n_1 + n_2 = \text{even} \quad (14)$$

$$\text{and} \quad Y(< \frac{g^m - 1}{2} >) = \sum_{l=0}^{\lambda-1} y(< p^i g^m >) \cdot \cos(\frac{2\pi p^i g^{m+l}}{4N}) \quad \text{otherwise} \quad (15)$$

where $m = 0, \cdots, \lambda - 1$, and $< \cdot >$ denotes mod $2N$ and $4N$ in (14) and (15), respectively. Like the above, (14) and (15) can be implemented via a sequence of SCC matrix-vector products. These SCC operations can be implemented via the efficient algorithm of [6] with $y(k)$ as the input and determine the odd-indexed output components.

3 COMPARISON AND DISCUSSION

In this section, we compare the computational complexity of the proposed fast algorithm with that of existing ones. Because the algorithm recursively computes the 2-D DCT of size $N \times N$ ($N = p^r$), it is straightforward to justify that the numbers of multiplication MUL(N) and addition ADD(N) required are [7]

$$\text{MUL}(N) = \sum_{i=0}^{r-1} \frac{N^2(1 - \frac{1}{p^2})}{2\lambda p^i} \cdot \sum_{j=0}^{r-i-1} [M_{cc}(\frac{\lambda}{p^{r-j-1}}) + M_{scc}(\frac{\lambda}{p^{r-j-1}})] \quad (16)$$

and

$$\text{ADD}(N) = \sum_{i=0}^{r-1} \frac{N^2(1 - \frac{1}{p^2})}{2\lambda p^i} \{ \sum_{j=0}^{r-i-1} [A_{CC}(\frac{\lambda}{p^{r-j-1}}) + A_{SCC}(\frac{\lambda}{p^{r-j-1}}) + \frac{2\lambda(1 + \frac{1}{p} - \frac{1}{p^2})}{p^{r-j-1}}] - \frac{2\lambda(\frac{1}{2} + \frac{1}{p} - \frac{1}{p^2})}{p^{r-1}} \} + (\frac{\lambda}{\frac{N}{p}} - 1) + N^3(1 + \frac{1}{p}) - N^2(\frac{3}{2} + \frac{5}{2p}) + N(3 + \frac{1}{p}) - 2 \quad (17)$$

where $M_{cc}(m)(A_{cc}(m))$, and $M_{scc}(m)(A_{scc}(m))$ stand for the numbers of multiplication (addition) required for the CC and SCC matrix-vector products of size m, respectively.

In the special case when $p = 2$, the recursive equations for computing the required numbers of multiplication can be simplified as

$$\mathrm{MUL}(N) = (\frac{N^2}{2}) \log_2 N \tag{18}$$

and the simplified expression for the numbers of additions can be found in [7]. Table 1 provides the numbers of multiplication and addition required for different $p^r \times p^r$ (p is an odd prime) DCT's by using the proposed fast algorithm and the RCD approach, while other existing fast algorithms are only applicable to $p = 2$. The 1-D DCT algorithms used in the RCD approach is based on the fast algorithms of [8] (for $p = 2$) and [9] (for p being an odd prime), whereas the CC/SCC matrix-vector products employ the algorithms of [10]. In Table 2, we also compare the computational complexity with other existing fast algorithms when $p = 2$.

From Table 2, we can note that the proposed fast algorithm calls for the same numbers of multiplication (and roughly the same numbers of addition) as those of [2, 4], which are known by far to be the most parsimonious approaches. The new algorithm, however, is more general than these algorithms in that it is also applicable to the 2-D DCT of size $p^r \times p^r$, where p is any prime. In such a case, as we can observe from Table 1, the proposed algorithm requires fewer numbers of multiplication (at the price of more additions) as compared with those of the RCD approach. Most importantly, the proposed algorithm, unlike the latter, does not require any transposition manipulation and the outputs corresponding to every dominant point can be computed independently, thus allowing for efficient parallel/pipelined implementations.

4 CONCLUSION

This paper describes a new fast algorithm for the recursive computation of $p^r \times p^r$ DCT, where p is a prime. The algorithm recursively decomposes the output components to form a set of CC and SCC matrix-vector products of various sizes. Utilizing existing fast algorithms for the computations of these CC and SCC matrix-vector products, we can thus obtain an algorithm with minimum number of multiplications.

References

[1] K.R. Rao and R. Yip, *Discrete Cosine Transform,* Academic Press, San Diego, CA, 1990.

[2] N.-I. Cho and S.-U. Lee, "Fast algorithm and implementation of 2-D discrete cosine transform," *IEEE Trans. Circuits Syst.*, vol. 38, no.3, pp. 297-305, Mar. 1991.

[3] P. Duhamel and C. Guillemot, "Polynomial transform computation of the 2-D DCT," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing,* vol. 3, pp. 1515-1518, 1990.

[4] P. Duhamel and C. Guillemot, "A polynomial-transform based computation of the 2-D DCT with minimum multiplicative complexity," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing,* vol. 3, pp. 1347-1350, Atlanta, 1996.

[5] N.-C. Hu and F.-F. Lu, " Fast computation of the two-dimensional generalized Hartley transforms," *IEE. Proc.-Vis. Image Signal Process.*, vol. 142, no. 1, pp. 35-39, Feb. 1995.

[6] N.-C. Hu and K.-C. Lin, "Skew-circular/circular correlation decomposition of prime-factor DCT," *IEE. Proc.-Vis. Image Signal Process.*, vol. 142, no. 4, pp. 241-246, Aug. 1995.

[7] W.-H. Fang, N.-C. Hu and S.-K. Shih, "Fast algorithms for the recursive computation of two-dimensional discrete cosine transform," submitted to *IEE Proc.*

[8] H. S. Hou, "A fast recursive algorithm for computing the discrete cosine transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 35, no. 10, pp. 1455-1461, Oct. 1987.

[9] M.T. Heideman, "Computation of an Odd-Length DCT from a Real-Valued DFT of the same length," *IEEE Trans. Signal Processing*, vol. 40, no. 1, pp. 54-61, Jan. 1992.

[10] H. H. Nussbaumer, *Fast Fourier Transform and Convolution algorithms,* New York : Spring-Verlag, 1982, Chapter 3, pp. 66-79.

Algorithms		3×3	5×5	7×7	9×9
row-column	Mult.	18	50	112	180
	Add.	24	130	420	612
proposed	Mult.	8	30	64	80
	Add.	27	169	523	1047

Table 1: Comparison of the RCD approach with the proposed for computing various 2-D DCT, where $N = p^r$ with p being an odd prime.

N	RCD	[3]	[2]	new
4	32(74)	16(68)	16(74)	16(74)
8	194(464)	96(484)	96(466)	96(466)
16	1024(2592)	512(2531)	512(2530)	512(2530)

Table 2: Comparison of the numbers of multiplication(addition) using the existing fast algorithms for the $N \times N$ DCT, where $N = 2^r$.