A Fast Algorithm for Computing Multidimensional DCT on Certain Small Sizes

Xinjian Chen, Qionghai Dai, and Chunwen Li

***Abstract*— This paper presents a new algorithm for the fast computation of multidimensional (m -D) discrete cosine transform (DCT) with size $N_1 \times N_2 \times \ldots \times N_m$, where N_i is a power of 2 and $N_i \leq$256, by using the tensor product decomposition of the transform matrix. It is shown that the m -D DCT or IDCT on these small sizes can be computed using only 1-D DCT's and additions and shifts. If all the dimensional sizes are the same, the total number of multiplications required for the algorithm is only $1/m$ times of that required for the conventional row-column method. We also introduce approaches for computing scaled DCT's, in which the number of multiplications is considerably reduced.**

***Index Terms*— Discrete cosine transform, fast algorithm, permutation matrix, tensor product.**

I. Introduction

THE discrete cosine transform (DCT) approaches the statistically optimal Karhunen-Loeve transform (KLT) for highly correlated signals, so it is widely used in digital signal processing, especially for speech and image data compression [1], [2]. Most video standards such as HDTV video coding, H.261, JPEG, and MPEG use the two dimensional (2-D) DCT as a standard transform coding scheme. The m-D DCT is also very important for many video processing applications. For example, the three dimensional (3-D) DCT coding is an alternative approach to the motion compensation transform coding technique used in video coding standard [17], and four-dimensional (4-D) DCT is generally required for 3-D motion images.

Many algorithms have been proposed for efficient computation of 1-D DCT [3]-[5]. The conventional approach to compute the 2-D DCT is row-column method, i.e., taking the 1-D DCT along the rows (columns) after along the columns (rows). The algorithms that work directly on 2-D data sets can lead to more efficient computation of 2-D DCT [6]-[13]. The Algorithms in [7]and[8] save 25% multiplications as compared to the conventional row-column method. In [6], Vetterli proposed an indirect polynomial approach using polynomial transform FFT and rotation that has multiplications between 50% and 75% of the conventional row-column method. Furthermore, the direct approach of polynomial transform proposed by Duhamel and Guillemot [9] and the fast algorithm based on trigonometric decomposition presented by Cho and Lee [10] can reduce the number of multiplications to 50% of the conventional row-column method. The algorithm in [12] is based on the decomposition of tensor product of transform matrix and requires the smallest number of operations, but it is only suitable for 8×8 DCT and has irregular architecture. Recently, several fast algorithms have been proposed for multidimensional DCT. In [14], A recursive algorithm for generating higher order m-D DCT by combing the computation of 2^m identical lower order DCT architectures is presented, but it needs too many operations. The algorithm proposed in [15] converts the m-D DCT with size $N \times N \times \ldots \times N$ (N is a power of 2) into 1-D DCT's and greatly reduces the number of operations, but it can not support applications that require different dimensional sizes. Based on the polynomial transform, an algorithm to compute m-D DCT with size $N_1 \times N_2 \times \ldots \times N_m$($N_i$ is a power of 2) proposed by Zeng [16] can reduce the overall computational and structure complexity.

In this paper, a new algorithm for the fast computation of m-D DCT with size $N_1 \times N_2 \times \ldots \times N_m$($N_i$ is a power of 2 and $N_i \leq$256) is presented. It is shown that the m-D data sets on these small sizes can be decomposed into 1-D independent vectors for 1-D DCT's based on the factorization of the tensor product form of DCT. The proposed algorithm requires only additions and shifts in both the pre-processing and post-processing stage. Hence the number of operations is almost comparable to that required in [16]. Both DCT and IDCT can be computed using 1-D DCT's, and this architecture can make the software and hardware implementations more flexible and realizable. We also introduced approaches for computing scaled DCT's, in which the number of multiplications is considerably reduced.

The rest of the paper is organized as follows. In section II, we give the algebraic derivations required for the proposed algorithm. In section III, the methods for computing multidimensional DCT, inverse DCT (IDCT) and scaled DCT's are provided. The comparison of the computation complexity of the proposed algorithm on some small sizes with the conventional row-column method is also given in this section. Finally, in section IV, we give conclusions.

II. Algebraic Derivations

For an input vector $\{x_0, x_1, \ldots, x_{n-1}\}$, the DCT output vector $\{y_0, y_1, \ldots, y_{n-1}\}$ is given by the relation

$$y_k = c_k \sum_{i=0}^{n-1} \cos \frac{2\pi k(2i+1)}{4n} x_k \qquad (1)$$

where $c_0 = 1/\sqrt{n}$, and $c_K = 1/\sqrt{2/n}$ for $1 \leq k \leq n-1$.

For convenience, we will neglect the scale factor c_k, and set $r(k) = \cos(2\pi k/4n)$. Then the n-point DCT matrix can be

Manuscript received June 11, 2001; revised June 6, 2002. The associate editor coordinating the review of this paper and approving it for publication was Prof. Xiang-Gen Xia.

The authors are with the Department of Automation, Tsinghua University, Beijing, China (e-mail: cxjian99@mails.tsinghua.edu.cn).

Publisher Item Identifier.

written as
$T_n =$

$$\begin{pmatrix} r(0) & r(0) & \cdots & r(0) \\ r(1) & r(3) & \cdots & r(2n-1) \\ \vdots & \vdots & \ddots & \vdots \\ r(n-1) & r(3(n-1)) & \cdots & r((n-1)(2n-1)) \end{pmatrix} \tag{2}$$

Assume $k < n$, n is a power of 2. As done in [12], let $P_{(k,1)}$ be the permutation matrix acting on the rows of $k \times k$ matrix reordering them as follows: $0, 2, \cdots, k-2, 1, 3, \cdots, k-1$, and let $P_{(k,2)}$ be the permutation matrix acting on the $k \times k$ matrix by keeping the first $k/2$ columns fixed and reversing the order of the last $k/2$ columns. Let I_k denote the $k \times k$ identity matrix. Define

$$F = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{3}$$

and $B_k =$

$$\begin{pmatrix} r\left(\frac{n}{2k}\right) & r\left(\frac{3n}{2k}\right) & \cdots & r\left(\frac{(2k-1)n}{2k}\right) \\ r\left(\frac{3n}{2k}\right) & r\left(\frac{9n}{2k}\right) & \cdots & r\left(\frac{3(2k-1)n}{2k}\right) \\ \vdots & \vdots & \ddots & \vdots \\ r\left(\frac{(2k-1)n}{2k}\right) & r\left(\frac{(2k-1)3n}{2k}\right) & \cdots & r\left(\frac{(2k-1)(2k-1)n}{2k}\right) \end{pmatrix} \tag{4}$$

According to lemma 2 in [11], we have the following factorization of the n-point DCT matrix.

$$T_n = \widetilde{P}_n \widetilde{K}_n \widetilde{R}_n \tag{5}$$

where

$$\widetilde{K}_n = 1 \oplus \Big(\bigoplus_{i=0}^{\log_2^{n-1}} B_{2^i} \Big) \tag{6}$$

$$\widetilde{P}_n = \prod_{i=1}^{\log_2^n} (P^t_{(n/2^{i-1},1)} \oplus I_{n-n/2^{i-1}}) \tag{7}$$

and

$$\widetilde{R}_n = \prod_{i=1}^{\log_2^n} ((F \otimes I_{2^{i-1}}) P^t_{(2^i,2)} \oplus I_{n-2^i}) \tag{8}$$

where $\oplus$ denotes the matrix direct sum, $\otimes$ denotes the tensor (or Kronecker) product, and P^t denotes the transpose of matrix P.

Lemma 1: For $0 < i < n$, n is a power of 2, $(3^i + 1) \bmod 4n \neq 0$ and $(3^i - 1) \bmod 4n \neq 0$.

Proof:

$$3^i = (4-1)^i = 4^i - i4^{i-1} + \cdots + \frac{i(i-1)}{2} 4^2 (-1)^{i-2} + i4(-1)^{i-1} + (-1)^i$$

If i is odd, then $(3i-1)/2$ and $(3i+1)/4$ are both odd . If i is even, suppose $i = 2^m t$, where $0 < m < \log_2^n$ and t is odd, then we have $(3i+1)/2$ is odd and

$$\frac{3i-1}{4 \times 2^m} = 4^{i-1}/2^m - t4^{i-2} + \cdots + 2t(i-1) - t$$

is also odd. Since n is a power of 2 and $2^m < n$, the theorem is true. ■

Lemma 1 says that 3 can be a generator for the summand which is isomorphic to ring Z_n [11], thus the elements $r(\frac{n}{2k}3^i)$, $i = 0, 1, \cdots, k-1$, are all distinct in absolute value. Define

$$G_k = \begin{pmatrix} r(\frac{n}{2k}3^{k-1}) & r(\frac{n}{2k}3^{k-2}) & \cdots & r(\frac{n}{2k}3^0) \\ -r(\frac{n}{2k}3^0) & r(\frac{n}{2k}3^{k-1}) & \cdots & r(\frac{n}{2k}3^1) \\ \vdots & \vdots & \ddots & \vdots \\ -r(\frac{n}{2k}3^{k-2}) & r(\frac{n}{2k}3^{k-3}) & \cdots & r(\frac{n}{2k}3^{k-1}) \end{pmatrix} \tag{9}$$

Since for $i \geq 0$,

$$r(\frac{n}{2k}3^i) = r(\frac{n}{2k}3^{i+2k}) \tag{10}$$

we should define

$$r(\frac{n}{2k}3^{-i}) = r(\frac{n}{2k}3^{-i+2k}) \tag{11}$$

Let $P_{(k,3)}$ be the signed permutation matrix acting on the column

$$\begin{pmatrix} r(\frac{n}{2k}) & r(\frac{3n}{2k}) & r(\frac{5n}{2k}) & \cdots & r(\frac{(2k-1)n}{2k}) \end{pmatrix}^t$$

is precisely

$$\begin{pmatrix} r(\frac{n}{2k}3^0) & r(\frac{n}{2k}3^1) & r(\frac{n}{2k}3^2) & \cdots & r(\frac{n}{2k}3^{k-1}) \end{pmatrix}^t$$

and $P_{(k,4)}$ be the permutation matrix reversing the order of the columns of $k \times k$ matrix. Then we have

$$G_k = P_{(k,3)} B_k P^t_{(k,3)} P_{(k,4)} \tag{12}$$

From (5) and (12), we obtain

$$T_n = P_n K_n R_n \tag{13}$$

where

$$K_n = 1 \oplus \Big(\bigoplus_{i=0}^{\log_2^{n-1}} G_{2_i} \Big) \tag{14}$$

$$P_n = \widetilde{P}_n (I_2 \oplus \Big(\bigoplus_{i=1}^{\log_2^{n-1}} P^t_{(2^i,3)} \Big)) \tag{15}$$

and

$$R_n = (I_2 \oplus \Big(\bigoplus_{i=1}^{\log_2^{n-1}} P^t_{(2^i,4)} P_{(2^i,3)} \Big)) \widetilde{R}_n \tag{16}$$

Obviously, P_n is a signed permutation matrix and R_n is an integral matrix in which the elements are all whole numbers. Equation (13) is an extension of the conclusion described for the DCT matrix on 8-points in [12].

The minimum number of additions required to compute the product of an arbitrary vector by a integral matrix R will be

denoted by $u(R)$. From (8) and (16), according to the factorization of $\widetilde{R}_n$, we have

$$u(R_n) = 2(1+2+4+\cdots+n/2) = 2(n-1) \tag{17}$$

The companion matrix of a polynomial $p(u) = u^k + 1$ is

$$C_k = \begin{pmatrix} 0 & 0 & \cdots & 0 & -1 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix}_{k\times k} \tag{18}$$

The properties of C_k that will be used in this paper are the following.

$$C_k^k = -I_k \tag{19}$$

$$C_k^i C_k^j = C_k^{i+j} \tag{20}$$

$$C_k^i \otimes I_m = C_{km}^{im} \tag{21}$$

$$(C_k^i)^t = C_k^{-i} = C_k^{2k-i} \tag{22}$$

$$G_k = -\sum_{i=0}^{k-1} r(\frac{n}{2k}3^{i-1})C_k^i \tag{23}$$

$$C_k^i G_k = G_k C_k^i \tag{24}$$

Lemma 2: There exists a signed permutation matrix $P_{c(k,2j+1)}$ such that

$$P_{c(k,2j+1)} C_k^i P_{c(k,2j+1)}^t = C_k^{i(2j+1)} \tag{25}$$

Proof: Let $P_{c(k,2j+1)}$ be the signed permutation matrix acting on the rows of $k\times k$ matrix by making row $i (0 \le i \le k-1)$ to replace row $t = i(2j+1) \bmod 2k$ respectively (if $t \ge k$, then be multiplied by -1 and replace row $t-k$. Obviously, $P_{c(k,2j+1)}^t P_{c(k,2j+1)} = I_k$ and $P_{c(k,2j+1)} C_k P_{c(k,2j+1)}^t = C_k^{2j+1}$. Hence $P_{c(k,2j+1)} C_k^i P_{c(k,2j+1)}^t = C_k^{i(2j+1)}$, and the theorem is true. ∎

For $l \le k$, l is a power of 2, define $lk \times lk$ matrix $R_{(l,k)} =$

$$\begin{pmatrix} I_l & C_l^1 & C_l^2 & \cdots & C_l^{l-1} \\ I_l & C_l^3 & C_l^6 & \cdots & C_l^{3(l-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ I_l & C_l^{2l-1} & C_l^{2(2l-1)} & \cdots & C_l^{(l-1)(2l-1)} \end{pmatrix} \otimes I_{k/l} \tag{26}$$

We can easily verify that

$$R_{(l,k)}^{-1} = \frac{1}{l} R_{(l,k)}^t \tag{27}$$

Applying the properties of C_K (19)-(24), by direct computation, we have

$$\frac{1}{l} R_{(l,k)} (G_l \otimes G_k) R_{(l,k)}^t = \frac{1}{l} R_{(l,k)} \cdot$$

$$\begin{pmatrix} r(\frac{n}{2l}3^{l-1})G_k & r(\frac{n}{2l}3^{l-2})G_k & \cdots & r(\frac{n}{2l}3^{0})G_k \\ -r(\frac{n}{2l}3^{0})G_k & r(\frac{n}{2l}3^{l-1})G_k & \cdots & r(\frac{n}{2l}3^{1})G_k \\ \vdots & \vdots & \ddots & \vdots \\ -r(\frac{n}{2l}3^{l-2})G_k & -r(\frac{n}{2l}3^{l-3})G_k & \cdots & r(\frac{n}{2l}3^{l-1})G_k \end{pmatrix}$$

$$\cdot R_{(l,k)}^t = \bigoplus_{j=0}^{l-1} (G_l^{(2j+1)} \otimes I_{k/l}) G_k \tag{28}$$

where

$$G_l^{(2j+1)} = -\sum_{i=0}^{l-1} r(\frac{n}{2l}3^{i-1}) C_l^{i(2j+1)} \tag{29}$$

The right hand of equation (28) is block diagonal. This conclusion to decompose $G_l \otimes G_k$ into block diagonal matrix is similar but different to that proposed in [12] which uses the "blown-up" construction, and it may lead to further simplification.

Obviously, $u(R_{(2,2)}) = 4$, and $R_{(l,l)}$ can be factorized as follows.

$$\begin{aligned} R_{(l,l)} P_{(l^2,1)}^t = (F \otimes I_{l^2/2})((R_{(l/2,l/2)} \\ \otimes I_2) \oplus ((\bigoplus_{i=0}^{l/2-1} C_l^{2i+1}(R_{l/2,l/2} \otimes I_2))) \end{aligned} \tag{30}$$

where F is defined in (3) and $P_{(l^2,1)}$ is defined as in (7). Therefore

$$u(R_{(l,l)}) = l^2 + 4u(R_{(l/2,l/2)}) \tag{31}$$

Recursively, we have

$$u(R_{(l,l)}) = l^2 \log_2^l \tag{32}$$

$$u(R_{(l,k)}) = kl \log_2^l \tag{33}$$

Lemma 3: Define $D_k = -\sum_{i=0}^{k-1} r(\frac{n}{2k}x_i)C_k^i$, where x_i is a power of 3. If $x_i/x_{i-1} = 3^{2j+1} (1 \le i \le k-1)$, then there exist signed permutation matrices Q_1 and Q_2 such that

$$Q_1 D_k Q_2 = G_k \tag{34}$$

Proof: Suppose $x_0 = 3^s$, then

$$D_k = -\sum_{i=0}^{k-1} r(\frac{n}{2k}3^{s+i(2j+1)}) C_k^i$$

From (23), we obtain

$$\begin{aligned} G_k &= -\sum_{i=0}^{k-1} r(\frac{n}{2k}3^{i-1}) C_k^{i-1} C_k \\ &= -\sum_{i=0}^{k-1} r(\frac{n}{2k}3^{s+i(2j+1)}) C_k^{i(2j+1)} C_k^s C_k \end{aligned}$$

Applying (25), we have

$$P_{c(k,2j+1)} D_k P_{c(k,2j+1)}^t C_k^{s+1} = G_k$$

∎

Lemma 3 plays important roles for the algebraic simplifications of $(G_l^{(2j+1)} \otimes I_{k/l})G_k$. For example, choosing $n = 16$ and $k = 8$, we have

$$(G_2^{(3)} \otimes I_4)G_8 = -\sum_{i=0}^{7} r(3^{2+5i})C_8^i = P_{c(8,5)}^t G_8 C_8^{-3} P_{c(8,5)} \tag{35}$$

and

$$\begin{aligned} G_8^{(5)}G_8(H \otimes I_4) &= -4\sum_{i=0}^{3} r(2 \times 3^{3+3i})C_8^{2i+1} \\ &= -4C_8 \sum_{i=0}^{3} r(2 \times 3^{3+3i})C_4^i \otimes I_2 \\ &= 4C_8(P_{c(4,3)}^t \otimes I_2)(G_4 \otimes I_2)(C_4^{-4}P_{c(4,3)} \otimes I_2) \end{aligned} \tag{36}$$

where

$$H = C_2^0 - C_2^1 = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \tag{37}$$

From (35), we know that there exist signed permutation matrices Q_1 and Q_2 such that

$$(G_2^{(3)} \otimes I_4)G_8 = Q_1 G_8 Q_2 \tag{38}$$

From (36), we also know that there exist signed permutation matrices Q_3 and Q_4 such that

$$G_8^{(5)}G_8(H \otimes I_4) = 4Q_3(G_4 \otimes I_2)Q_4 \tag{39}$$

In order to obtain the rules of the algebraic simplifications of $(G_l^{(2j+1)} \otimes I_{k/l})G_k$, we programmed for computing these expressions and summarized two propositions as follows.

Proposition 1: For $l = 2^s < k \leq 128$, where $s \geq 0$,

1) If $s = 2t$, then there exist signed permutation matrices Q_1 and Q_2 such that

$$(G_l^{(2j+1)} \otimes I_{k/l})G_k = 2^{t-1}Q_1 G_k Q_1(H^t \otimes I_{k/2}) \tag{40}$$

where H is defined in (37).

2) If $s = 2t+1$, then there exist signed permutation matrices Q_3 and Q_4 such that

$$(G_l^{(2j+1)} \otimes I_{k/l})G_k = 2^t Q_3 G_k Q_4 \tag{41}$$

Proposition 2: For $l = 2^s = k \leq 128$, where $s > 0$,

1) If $s = 2t$, then there exist signed permutation matrices $Q_{1(j)}$ and $Q_{2(j)}(0 \leq j \leq k-1)$ such that
$G_k^{(2j+1)}G_k =$

$$\begin{cases} 2^t Q_{1(j)}(G_{k/2} \otimes I_2)Q_{2(j)} & j = u-1 \\ 2^t Q_{1(j)}(G_{k/4} \otimes I_4)Q_{2(j)}(H^t \otimes I_{k/2}) & j = 2u-1 \\ 2^{t+1} Q_{1(j)}(G_{k/8} \otimes I_8)Q_{2(j)} & j = 4u-1 \\ \vdots & \vdots \\ 2^{2t-1} Q_{1(j)}(G_1 \otimes I_k)Q_{2(j)}(H^t \otimes I_{k/2}) & j = ku/2-1 \\ 2^{2t-1} I_k & j = ku-1 \end{cases} \tag{42}$$

where u is odd.

2) If $s = 2t+1$, then there exist signed permutation matrices $Q_{3(j)}$ and $Q_{4(j)}(0 \leq j \leq k-1)$ such that
$G_k^{(2j+1)}G_k =$

$$\begin{cases} 2^t Q_{3(j)}(G_{k/2} \otimes I_2)Q_{4(j)}(H^t \otimes I_{k/2}) & j = u-1 \\ 2^{t+1} Q_{3(j)}(G_{k/4} \otimes I_4)Q_{4(j)} & j = 2u-1 \\ 2^{t+1} Q_{3(j)}(G_{k/8} \otimes I_8)Q_{4(j)}(H^t \otimes I_{k/2}) & j = 4u-1 \\ \vdots & \vdots \\ 2^{2t} Q_{3(j)}(G_1 \otimes I_k)Q_{4(j)}(H^t \otimes I_{k/2}) & j = ku/2-1 \\ 2^{2t} I_k & j = ku-1 \end{cases} \tag{43}$$

where u is odd.

The proof of the propositions above is very trivial, but we can easily verify them by computing $(G_l^{(2j+1)} \otimes I_{k/l})G_k$ for any $l \leq k \leq 128$, l, k are powers of 2. It is also shown that (40)- (43) are not suitable for the case $k \geq 256$ in which the computation results of $(G_l^{(2j+1)} \otimes I_{k/l})G_k$ are difficult to simplify. We can obtain the permutation matrix by applying lemma 3, e.g. (35) and (36).

Corollary 1: For $l = 2^s < k \leq 128$, where $s \geq 0$,

1) If $s = 2t$, then there exist integral matrices $R_{1(l,k)}$ and $R_{2(l,k)}$ such that

$$G_l \otimes G_k = \frac{1}{2^{t+1}} R_{1(l,k)}(G_k \otimes I_l)R_{2(l,k)} \tag{44}$$

2) If $s = 2t+1$, then there exist integral matrices $R_{3(l,k)}$ and $R_{4(l,k)}$ such that

$$G_l \otimes G_k = \frac{1}{2^{t+1}} R_{3(l,k)}(G_k \otimes I_l)R_{4(l,k)} \tag{45}$$

The integral matrices mentioned in corollary 1 can be obtained from (28) and proposition 1. It is clear that

$$u(R_{1(l,k)}) = u(R_{3(l,k)}) = u(R_{4(l,k)}) = lk\log_2^l \tag{46}$$

and

$$u(R_{2(l,k)}) = lk\log_2^l + lk \tag{47}$$

In the product with an arbitrary vector by $G_l \otimes G_k$, $1/2^{t+1}$ means $kl(t+1)$ shifts.

Corollary 2: For $l = 2^s = k \leq 128$, where $s > 0$,

1) If $s = 2t$, then there exist integral matrices $R_{1(k,k)}$ and $R_{2(k,k)}$ such that

$$G_k \otimes G_k = R_{1(k,k)}S_1(K_k \otimes I_k)R_{2(k,k)} \tag{48}$$

where K_k is defined in (14) and $S_1 =$
$(\frac{1}{2}I_k \oplus \frac{1}{2}I_{k+2k} \oplus \frac{1}{4}I_{4k+8k} \oplus \cdots \oplus \frac{1}{2^t}I_{2^{4t-2}+2^{4t-1}})$.

2) If $s = 2t+1$, then there exist integral matrices $R_{3(k,k)}$ and $R_{4(k,k)}$ such that

$$G_k \otimes G_k = R_{3(k,k)}S_2(K_k \otimes I_k)R_{4(k,k)} \tag{49}$$

where$S_2 =$
$(\frac{1}{2}I_k \oplus \frac{1}{2}I_{k+2k} \oplus \frac{1}{4}I_{4k+8k} \oplus \cdots \oplus \frac{1}{2^{t+1}}I_{2^{k^2/2}})$.

The integral matrices mentioned in corollary 2 can be obtained from (28) and proposition 2.

$$u(R_{1(k,k)}) = u(R_{3(k,k)}) = k^2 \log_2^k \tag{50}$$

$$u(R_{2(k,k)}) = k^2 \log_2^k + k(k-1)/3 \tag{51}$$

$$u(R_{4(k,k)}) = k^2 \log_2^k + k(2k-1)/3 \tag{52}$$

In the product with an arbitrary vector by $G_k \otimes G_k$, S_1 means

$$k + k\sum_{i=1}^{t} i(2^{2i-2} + 2^{2i-1}) = \frac{(3t-1)k^2 + 4k}{3} \tag{53}$$

shifts, and S_2 means

$$\frac{k^2}{2}(t+1) + k + k\sum_{i=1}^{t} i(2^{2i-2} + 2^{2i-1}) = \frac{(3t+1)k^2 + 4k}{3} \tag{54}$$

shifts.

Corollary 3: For $256 \geq n_1 > n_2$, there exist diagonal matrix S and integral matrices R_1 and R_2 such that

$$K_{n_1} \otimes G_{n_2} = R_1 S(K_{n_1} \otimes I_{n_2}) R_2 \tag{55}$$

where S only includes shifts and R_1, R_2 only include additions.

Proof: $K_{n_1} = K_{n_2} \oplus G_{n_2} \oplus G_{2n_2} \oplus \cdots \oplus G_{n_1/2}$. Applying corollary 1 and corollary 2, we can easily verify it. ■

Corollary 4: For $256 \geq n_1 \geq n_2$, there exist diagonal matrix S and integral matrices R_1 and R_2 such that

$$K_{n_1} \otimes K_{n_2} = R_1 S(K_{n_1} \otimes I_{n_2}) R_2 \tag{56}$$

where S only includes shifts and R_1, R_2 only include additions.

Proof: $K_{n_2} = 1 \oplus G_1 \oplus G_2 \oplus G_4 \oplus \cdots \oplus G_{n_2/2}$. Applying corollary 3, we can easily verify it. ■

Corollary 5: For $256 \geq n_1 \geq n_2 \geq n_4 \geq \cdots \geq n_m$, there exist diagonal matrix $\widetilde{S}$ and integral matrices $\widetilde{R}_1$ and $\widetilde{R}_2$ such that

$$K_{n_1} \otimes K_{n_2} \otimes \cdots \otimes K_{n_m} = \widetilde{R}_1 \widetilde{S}(K_{n_1} \otimes I_{n_2 n_3 \cdots n_m}) \widetilde{R}_2 \tag{57}$$

where $\widetilde{S}$ only includes shifts and $\widetilde{R}_1$, $\widetilde{R}_2$ only include additions.

Proof: By using the distributive property of tensor product

$$AB \otimes CD = (A \otimes C)(B \otimes D) \tag{58}$$

and mainly applying corollary 4, we can obtain (57). ■

III. DCT AND SCALED DCT

In this section, we will introduce the algorithm for the fast computation of multidimensional DCT and inverse DCT (IDCT). Two methods for computing scaled DCT's (we define them as scaled DCT-I and scaled DCT-II) are also presented in this section.

A. DCT and IDCT

The m-D DCT with size $N_1 \times N_2 \times \cdots \times N_m$ can be represented by the tensor product form

$$T_{n_1,n_2,\cdots,n_m} = T_{n_1} \otimes T_{n_2} \otimes \cdots \otimes T_{n_m} \tag{59}$$

Suppose $256 \geq N_1 \geq N_2 \geq \cdots \geq N_m$, by applying (13) and (57), we have

$$\begin{aligned} T_{n_1,n_2,\cdots,n_m} = {} & (P_{n_1} \otimes P_{n_2} \otimes \cdots \otimes P_{n_m}) \widetilde{R}_1 \widetilde{S} \\ & (K_{n_1} \otimes I_{n_2 n_3 \cdots n_m}) \widetilde{R}_2 (R_{n_1} \otimes R_{n_2} \otimes \cdots \otimes R_{n_m}) \end{aligned} \tag{60}$$

From (13) we also have

$$K_{n_1} = P_{n_1}^{-1} T_{n_1} R_{n_1}^{-1} \tag{61}$$

Equation (60) and (61) say that we can decompose the $N_1 \times N_2 \times \cdots \times N_m$ data sets into $n_2 n_3 \cdots n_m$ vectors for 1-D DCT's on n_1-point with one additional pre-processing stage and post-processing stage which include no multiplications.

For the purpose of reducing the number of additions, we introduce a new method to compute the DCT with 1-D vectors in which the transform matrix is not T_n but K_n.

From (2) and (4), applying the property

$$r(\alpha) + r(\beta) = 2r(\frac{\alpha+\beta}{2}) r(\frac{\alpha-\beta}{2}) \tag{62}$$

We have

$$B_k = L_k T_k M_K \tag{63}$$

where

$$L_k = \begin{pmatrix} 1 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & 1 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 \end{pmatrix} \tag{64}$$

and

$$M_k = \mathtt{diag}\left(\frac{1}{2r\left(\frac{n}{2k}(2i+1)\right)} \right), i = 0, 1, \cdots, k-1. \tag{65}$$

Combing (12) and (63) yields

$$P_{(k,3)} L_k T_k M_k P_{(k,3)}^t P_{(k,4)} = G_k \tag{66}$$

It is assumed that our algorithm uses the most efficient fast algorithm for 1-D k-point DCT [4], which requires $1/2k \log_2^k$ multiplications and $3/2k \log_2^k - k + 1$ additions. Since T_k is DCT transform matrix, from (66) we know that the product with an arbitrary vector by G_k includes $1/2k \log_2^k + k$ multiplications and $3/2k \log_2^k$ additions. So K_k includes $1/2k \log_2^k$ multiplications and $3(1/2k \log_2^k - k + 1)$ additions.

From (60) we know that the number of multiplications required for the computation of m-D DCT is $(1/2n_1 \log_2^{n_1}) n_2 n_3 \cdots n_m$. The formula for the total number of additions required for the algorithm is very complex and may

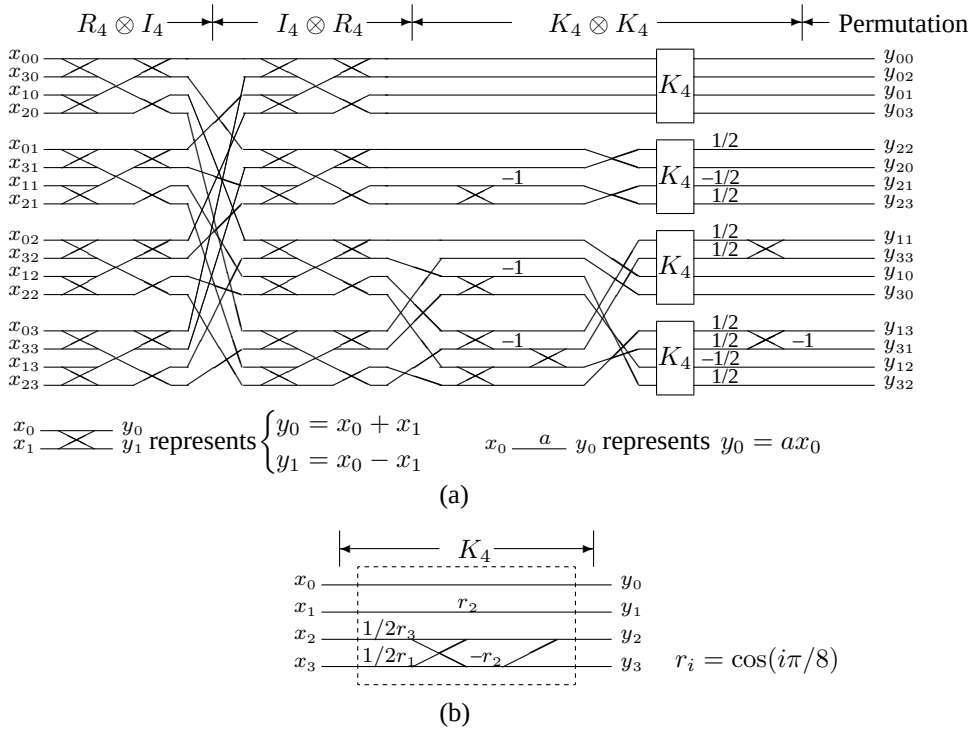

Fig. 1. (a) The signal flow graph for 4×4 DCT. (b) The signal flow graph in block K_4

different with different sizes. Based on the process of decomposition, we can count the number of additions and shifts step by step according to the previous corresponding formulae.

As an example, Fig.1 shows the signal flow graph for 4×4 DCT. In this example,

$$R_4 = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \end{pmatrix} \tag{67}$$

$$P_4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{68}$$

and

$$K_4 = \begin{pmatrix} 1 & & & \\ & G_1 & & \\ & & G_2 & \end{pmatrix} = \begin{pmatrix} 1 & & & \\ & r_2 & & \\ & & r_3 & r_1 \\ & & -r_1 & r_3 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & & & \\ & r_2 & & \\ & & \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & \\ & -r_2 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} \frac{1}{2r_3} & \\ & \frac{1}{2r_1} \end{pmatrix} \end{pmatrix} \tag{69}$$

where $r_i = \cos(i\pi/8)$.

The tensor product form of IDCT is

$$T^t_{n_1,n_2,\cdots,n_m} = (R_{n_1} \otimes R_{n_2} \otimes \cdots \otimes R_{n_m})^t \widetilde{R}^t_2 (K^t_{n_1} \otimes I_{n_2 n_3 \cdots n_m}) \widetilde{S}^t \widetilde{R}^t_1 (P_{n_1} \otimes P_{n_2} \otimes \cdots \otimes P_{n_m})^t \tag{70}$$

Define

$$U_k = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & -1 \\ 0 & 0 & 0 & \cdots & -1 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & -1 & \cdots & 0 & 0 \\ 0 & -1 & 0 & \cdots & 0 & 0 \end{pmatrix} \tag{71}$$

By lemma 2, notice that $U_k = P_{c(k,2k-1)}$ and $G^t_k = G^{(2k-1)}_k$, we have

$$U_k G_k U^t_k = G^t_k \tag{72}$$

therefore

$$V_k K_k V^t_k = K^t_k \tag{73}$$

where

$$V_k = 1 \oplus \Big(\bigoplus_{i=0}^{\log_2^{k-1}} U_{2^i} \Big) \tag{74}$$

Equations (70) and (73) say that IDCT can also use the same computation stage of 1-D DCT's as that of DCT, and only the pre-processing stage and post-processing stage are different.

TABLE I
COMPARISON OF THE NUMBER OF MULTIPLICATIONS

Input Size	Row-Column [4]	This Method (DCT)	This Method (Scaled DCT-I)	This Method (Scaled DCT-II)
4×4	32	16	6	8
8×8	192	96	54	64
16×16	1024	512	342	384
32×32	5120	2560	1878	2048
4×4×4	192	64	28	32
8×8×8	2304	768	476	512

TABLE II
COMPARISON OF THE NUMBER OF ADDITIONS

Input Size	Row-Column [4]	This Method (DCT)	This Method (Scaled DCT-I)	This Method (Scaled DCT-II)
4×4	72	74 (9)	72 (1)	72 (4)
8×8	464	466 (49)	462(6)	472 (36)
16×16	2592	2722 (321)	2584 (21)	2600 (196)
32×32	13376	14082 (1633)	13614 (58)	13896 (1284)
4×4×4	432	448 (72)	432 (9)	432 (32)
8×8×8	5568	5600 (784)	5528 (96)	5696 (576)

[a] Number in the parentheses indicate the shifts required for the computation of DCT.

B. Scaled DCT-I

The approach for computing scaled DCT-I requires the least number of multiplications, but it has irregular architectures that may not suitable for VLSI implementations especially for larger problem sizes.

From (12), (13), (63) and (73), notice that $B_k = B_k^t$, we have

$$G_k = P_{(k,3)} M_k R_k^t V_k K_k V_k^t P_k^t L_k^t P_{(k,3)}^t P_{(k,4)} \tag{75}$$

From (13) and (75), we have

$$T_n = \widehat{P}_n \widehat{M}_n \widehat{R}_{n,1} \widehat{K}_n \widehat{R}_{n,2} \tag{76}$$

where

$$\widehat{K}_n = I_2 \oplus \Big(\bigoplus_{i=1}^{\log_2^{n-1}} K_{2^i} \Big) \tag{77}$$

$$\widehat{P}_n = P_n \Big(1 \oplus \Big(\bigoplus_{i=0}^{\log_2^{n-1}} P_{(2^i,3)} \Big)\Big) \tag{78}$$

$$\widehat{M}_n = 1 \oplus \Big(\bigoplus_{i=0}^{\log_2^{n-1}} M_{2^i} \Big) \tag{79}$$

$$\widehat{R}_{n,1} = 1 \oplus \Big(\bigoplus_{i=0}^{\log_2^{n-1}} R_{2^i}^t V_{2^i} \Big) \tag{80}$$

and

$$\widehat{R}_{n,2} = \Big(1 \oplus \Big(\bigoplus_{i=0}^{\log_2^{n-1}} V_{2^i}^t P_{2^i}^t L_{2^i}^t P_{(2^i,3)}^t P_{(2^i,4)} \Big)\Big) R_n \tag{81}$$

The tensor product form of scaled DCT-I is

$$T_{n_1,n_2,\cdots,n_m} = \Big(\bigotimes_{i=1}^{m} (\widehat{P}_{n_i} \widehat{M}_{n_i})\Big)\Big(\bigotimes_{i=1}^{m} \widehat{R}_{n_i,1}\Big)\Big(\bigotimes_{i=1}^{m} \widehat{K}_{n_i}\Big)\Big(\bigotimes_{i=1}^{m} \widehat{R}_{n_i,2}\Big) \tag{82}$$

Because $\widehat{P}_{n_i}$ is the signed permutation matrix and $\widehat{M}_{n_i}$ is diagonal, so $\bigotimes_{i=1}^{m} (\widehat{P}_{n_i} \widehat{M}_{n_i})$ can be incorporated into scaling and quantization by first computing a scaled DCT. The number of multications lies on $\bigotimes_{i=1}^{m} \widehat{K}_{n_i}$. For $n_i \leq$512, n_i is a power of 2, we can factorize $\bigotimes_{i=1}^{m} \widehat{K}_{n_i}$ according to (57), so the implementation of scaled DCT-I is practical.

C. Scaled DCT-II

The approach for computing scaled DCT-II needs more multiplications than that of scaled DCT-I, but it has regular architectures that is suitable for VLSI implementations. For 512$\geq n_1 \geq n_2 \geq \cdots \geq n_m$, n_i is a power of 2, only the block $G_{n_1/2}$ in $K_{n_i} (1 \leq i \leq m)$ be factorized according to (75). By applying (57), finally we can factorize $T_{n_1,n_2,\cdots,n_m}$ in the form

$$T_{n_1,n_2,\cdots,n_m} = \widehat{P}\widehat{M}\widehat{R}_1 (K_{n_1/2} \otimes I_{2n_2n_3\cdots n_m}) \widehat{R}_2 \tag{83}$$

where $\widehat{P}\widehat{M}$ is incorporated into scaling and quantization as in (82), $\widehat{R}_1$ and $\widehat{R}_2$ and are the matrices which only include additions and shifts.

Equation (83) says that we can decompose the $N_1 \times N_2 \times \cdots \times N_m$ data sets into $2n_2n_3\cdots n_m$ vectors for 1-D DCT's on $n_1/2$-point. Hence the number of multiplications required for the computation of scaled DCT-II is

$(1/2n_1 \log_2^{n_1} - n_1/2)n_2 n_3 \cdots n_m$, which is less than that required for computing DCT by $n_1 n_2 \cdots n_m/2$. We compare the number of multiplications and additions required for this algorithm to that required for the conventional row-column method based on Lee's approach [4]. The results are summarized in Table I and Table II. It is shown that our algorithm can greatly reduced the number of multiplications while the number of additions is almost comparable to that required for the conventional row-column method. In the method for computing Scaled DCT-I, we can incorporate many shifts and G_1's into multiplications, so the number of operations required for this method is slightly different to that required for computing DCT and scaled DCT-II.

IV. Conclusion

In this paper, we have proposed a new algorithm for the fast computation of the m-D DCT with size $N_1 \times N_2 \times \cdots \times N_m$, where N_i is a power of 2 and $N_i \leq$256. Based on the method of factorizing the tensor product form of m-D DCT, the m-D DCT or IDCT on these small sizes can be computed using only 1-D DCT's together with additions and shifts in the pre-processing and post-processing stage. If all the dimensional sizes are the same, the total number of multiplications required for the algorithm is only $1/m$ times of that required for the conventional row-column method. We also introduced two methods for computing scaled DCT's (scaled DCT-I and scaled DCT-II) with size $N_1 \times N_2 \times \cdots \times N_m$, where N_i is a power of 2 and $N_i \leq$512. One requires the least number of multiplications but has irregular architecture, and another (the computation of scaled DCT-II) can both greatly reduce the number of multiplications and has regular computational structure which is suitable for VLSI implementations.

Acknowledgment

The authors would like to thank the Associate Editor Prof. X.-G. Xia and anonymous reviewers for their useful comments.

References

[1] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Trans. Comput.*, vol. C-23, pp. 90–93, Jan. 1974.

[2] R. J. Clark, "Relation between the Karhunen-Loeve and cosine transform," *Proc. Inst. Elec. Engr., pt. F*, vol.128, pp. 359–360, Nov. 1981.

[3] M. J. Narasimha and A. M. Peterson, "On the computation of the discrete cosine transform," *IEEE Trans. Commun.*, vol. COM-26, pp. 934–936, June 1978.

[4] B. G. Lee, "A new algorithm to compute the discrete cosine transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-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. ASSP-35, pp. 1455–1461, Oct. 1987.

[6] M. Vetterli, "Fast 2-D discrete cosine transform," in *Proc. ICASSP'85*, pp. 1538–1541, Mar. 1985.

[7] M. A. Harque, "A two-dimensional fast cosine transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-33, pp. 1532–1539, Dec. 1985.

[8] C. Ma, "A fast recursive two-dimensional cosine transform," *Proc. SPIE Int Soc. Opt. Eng.*, pp. 541–548, 1988.

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

[10] N. I. Cho and S. U. Lee, "Fast algorithm and implementation of 2-D discrete cosine transform," *IEEE Trans. Circuit Syst.*, vol. 38, pp. 297–305, Mar. 1991.

[11] E. Feig and S. Winograd, "On the multiplicative complexity of discrete cosine transform," *IEEE Trans. Inform. Theory*, vol. 38, pp. 1387–1391, July 1992.

[12] E. Feig and S. Winograd, "Fast algorithms for the discrete cosine transform," *IEEE Trans. Signal Process.*, vol. 40, pp. 2174–2193, Sept. 1992.

[13] N. I. Cho and S. U. Lee, "A Fast 4×4 DCT algorithm for the recursive 2-D DCT," *IEEE Trans. Signal Process.*, vol. 40, pp. 2166–2173, Sept, 1992.

[14] A. Elnaggar and H. M. Alnuweiri, "A new multidimensional recursive architecture for computing the discrete cosine transform," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, pp. 113–119, Feb, 2000.

[15] Z. S. Wang, Z. Y. He, C. R. Zou, and J. D. Z. Chen, "A Generalized fast algorithm for -D discrete cosine transform and its application to motion picture coding," *IEEE Trans. Circuits Syst. II*, vol. 46, pp. 617–627, May 1999.

[16] Y. Zeng, G. Bi and A. R. Leyman, "New polynomial transform algorithm for multidimensional DCT," *IEEE Trans. Signal Process.*, vol. 48, pp. 2814–2821, Oct. 2000.

[17] Y. L. Siu and W. C. Siu, "Variable temporal-length 3-D discrete cosine transform coding," *IEEE Trans. Image Processing*, vol. 6, pp. 758–763, May 1997.

Xinjian Chen was born in Hubei, China, in 1976. He received the B. S. degree in electrical engineering from Huazhong University of Science and Technology, Wuhan, China, in 1999. He is currently pursuing the M. S. degree with the Department of Automation, Tsinghua University, Beijing, China. His current research interests include signal processing, video compression, VLSI design and applied mathematics.

Qionghai Dai was born in Shanghai, China, in 1964. He received the B. S. degree in mathematics from Shanxi Normal University, Xi'an, China, in 1987, and the M. S. Degree in computer science in 1994 and the Ph. D. Degree in automation in 1996, both from Northeasten University, Shenyang,China.

Since 1997, he has been with Tsinghua University, Beijing, China, where he is Associate Professor of the Department of Automation, the Director of the Multimedia centre. His main research interests include digital signal processing, broadband networks, very high-speed multimedia communications, and IP-based networks.

Chunwen Li was born in Henan, China, in 1958. He received B. S. degree in 1982, and M. S. and Ph. D degree in 1989, all in automation from Tsinghua University, Beijing, China.

He is currently a Professor of the Department of Automation, Tsinghua University, Beijing, China. His main research interests include nonlinear control, stability of dynamic systems, fast algorithms, quantum control and robot control.