MATRIX FACTORIZATION FOR FAST DCT ALGORITHMS

Wenjia Yuan[a] *Pengwei Hao*[a,b] *Chao Xu*[a]

[a]Center for Information Science, Peking University, Beijing, 100871, China
[b]Department of Computer Science, Queen Mary, University of London, E1 4NS, UK

ABSTRACT

Two principles to produce new possibilities for the radix-2 Discrete Cosine Transform (DCT) have been presented in this paper. One is to employ matrix factorization through revealing the intrinsic relationship among several existing famous algorithms, which is regarded as an effective guide for exploring new algorithms. The other is to make use of the orthogonal property of the DCT matrix. As long as the recursive kernel of an algorithm is orthogonal, there must be a twin fast DCT algorithm of it. Matrix factorization is applied through the research and can be used to show how data flows and compute the computational complexity easily. At the end of this paper, we also present a new fast algorithm for DCT. It enjoys the parallel structure which is simpler for programming and hardware implementation and keeps the same numbers of the additions and multiplications as the fastest algorithms.

1. INTRODUCTION

DCT has widely been applied in speech and image processing [1-3] and fast DCT algorithms, which simplify the computational structure and lower the computational complexity, have been adopted in most image and video coding standards and processing systems.

There are two categories for computing the fast DCT, indirect and direct implementations.

For indirect computation, DCT is decomposed into other fast algorithms with smaller scales, such as DFT, FHT [4-5]. These algorithms make use of the mature model in programming and hardware implementation, but the arithmetic complexity is increased.

Among the direct implementations, Wang [6] and Chen [7] firstly put forward the fast DCT algorithms whose computational complexity is the same. Lee proposed a fast algorithm of IDCT, but the inversion and division of the cosine values give rise to the instability problems. The papers [1,8-10] presented the fast algorithms of DCT later, which are regarded as the fastest ones commonly applied. Their computational complexity is the same, with $\frac{N}{2}\log_2 N$ additions and $\frac{3}{2}N\log_2 N-N+1$ multiplications.

In this paper, we reveal the regularity of the existing algorithms of fast DCT using matrix factorization, which is also a good tool to find the computational complexity. By comparison, we learn that the existing algorithms are constructed by following the same idea, with which we can contrive as many fast DCT algorithms as possible. At the same time, the orthogonal property [11-12] can also produce the twin fast algorithms, as long as the recursive kernel is orthogonal. At last, we present a fast DCT algorithm which is different from the algorithms we have seen in the literature. Its parallel structure makes it simpler for programming and hardware implementation.

2. THE INTRINSIC REGULARITY

We denote the input sequence of length N is $x(n)$, $n\in[0,\mathrm{N}-1]$ and the output sequence is $X(k)$, $k\in[0,\mathrm{N}-1]$. Thus, the Discrete Cosine Transform (DCT) can be defined as:

$$X(k)=\frac{2}{N}\varepsilon(k)\sum_{n=0}^{N-1}x(n)\cos(\pi(2n+1)k/2N) \quad (1)$$

where $\varepsilon(k)=1/\sqrt{2}$ for k=0, and $\varepsilon(k)=1$ for $k\in[1,N-1]$.

For most of the existing fast algorithms, the normalized form is used for the fast algorithm derivation:

$$X(k)=\sum_{n=0}^{N-1}x(n)\cos(\pi(2n+1)k/2N) \quad (2)$$

The fast algorithms are all formulated in the recursive form to reduce the repetitious operations. We expect to look insight into the intrinsic regularity of the existing fast DCT methods. Five main algorithms [1-2, 8-10] are listed in Table 1 and the matrix factorization not only helps us learn the relationship among them, but also shows the computational complexity.

2.1. The General Formula

From recursive kernels of the five algorithms in Table 1, we learn that the algorithms of the length of N are composed of 4 types of basic matrices: permutation matrices, integer coefficient matrices, trigonometric coefficient matrices and the block diagonal matrices for the $N/2$ recursive kernels.

This work was supported by FANEDD of China (200038) and NKBRPC (2004CB318005).

In Equation (8), we take matrix factorization of Lee's method with the size 4 by 4 for example.

Table 1 Fast DCT algorithms in matrix form

Algorithm	Recursive Regularity
[1] Hou	$\begin{bmatrix} I & 0 \\ 0 & K_H \end{bmatrix} \cdot \begin{bmatrix} H_{\frac{N}{2}} & 0 \\ 0 & H_{\frac{N}{2}} \end{bmatrix} \cdot \begin{bmatrix} I & 0 \\ 0 & Q_H \end{bmatrix} \cdot \begin{bmatrix} I & I \\ I & -I \end{bmatrix}$ (3)
[2] Lee	$[P_{L1}] \cdot \begin{bmatrix} I & 0 \\ 0 & Q_L \end{bmatrix} \cdot \begin{bmatrix} L_{\frac{N}{2}} & 0 \\ 0 & L_{\frac{N}{2}} \end{bmatrix} \cdot [P_{L2}]$ (4)
[8] Kok	$[P_{K1}] \cdot \begin{bmatrix} I & 0 \\ 0 & L_K \end{bmatrix} \cdot \begin{bmatrix} K_{\frac{N}{2}} & 0 \\ 0 & K_{\frac{N}{2}} \end{bmatrix} \cdot \begin{bmatrix} I & 0 \\ 0 & Q_K \end{bmatrix} \cdot [P_{K2}]$ (5)
[9] L&H	$\begin{bmatrix} I & 0 \\ 0 & L \end{bmatrix} \cdot \begin{bmatrix} P & 0 \\ 0 & P \end{bmatrix} \cdot \begin{bmatrix} T_{\frac{N}{2}} & 0 \\ 0 & T_{\frac{N}{2}} \end{bmatrix} \cdot \begin{bmatrix} I & 0 \\ 0 & Q \end{bmatrix} \cdot \begin{bmatrix} I & I \\ I & -I \end{bmatrix}$ (6)
[10] C&P	$\begin{bmatrix} I & 0 \\ 0 & L_C \end{bmatrix} \cdot \begin{bmatrix} C_{\frac{N}{2}} & 0 \\ 0 & C_{\frac{N}{2}} \end{bmatrix} \cdot \begin{bmatrix} I & 0 \\ 0 & Q_C \end{bmatrix} \cdot \begin{bmatrix} I & I \\ I & -I \end{bmatrix}$ (7)

$$\hat{IDCT} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{2}Sec(\frac{\pi}{8}) & 0 \\ 0 & 0 & 0 & \frac{1}{2}Sec(\frac{3\pi}{8}) \end{bmatrix} \cdot \begin{bmatrix} 1 & 1/\sqrt{2} & 0 & 0 \\ 1 & -1/\sqrt{2} & 0 & 0 \\ 0 & 0 & 1 & 1/\sqrt{2} \\ 0 & 0 & 1 & -1/\sqrt{2} \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (8)$$

The four matrices above from left to right are respectively integer coefficient matrix, trigonometric coefficient matrix, the block diagonal matrix made of two recursive kernel matrices for the length of $N/2$ and permutation matrix.

With the tool of matrix factorization, not only is the structure of each fast algorithm clearly shown, but also we can deduce one from another, which is critical for us to explore new possibilities. And we will deduce Hou's algorithm [1] from Kok's [8] to illustrate how the general formula is produced.

For Hou's algorithm, $\hat{DCT}_N = [P_{H1}] \cdot [H_N] \cdot [P_{H2}]$ and $\hat{DCT}_N$ is the normalized form of DCT matrix with size N by N.

$[H_N]$ is the recursive kernel (3). $[P_{H1}]$ is a bit reversal matrix and $[P_{H2}]$ is to split the input data sequence into 2 parts: the numbers in the even and ordinal positions of the original sequence and those in the odd and reverse ordinal positions. (Set the first position as 0.)

The recursive kernel of Kok's algorithm is $\hat{DCT}_N$ itself and we can rewrite Equation (5) as Equation (9) and (10):

$$\hat{DCT}_N = [P_{H1}] \cdot [P_{H1}]^{-1} \cdot [K_N] \cdot [P_{H2}]^{-1} \cdot [P_{H2}] \quad (9)$$

$$\hat{DCT}_N = [P_{K1}] \cdot \begin{bmatrix} I & 0 \\ 0 & L_K \end{bmatrix} \cdot \begin{bmatrix} P_{H1} & 0 \\ 0 & P_{H1} \end{bmatrix} \cdot \begin{bmatrix} P_{H1}^{-1} \cdot K_{\frac{N}{2}} \cdot P_{H2}^{-1} & 0 \\ 0 & P_{H1}^{-1} \cdot K_{\frac{N}{2}} \cdot P_{H2}^{-1} \end{bmatrix} \cdot \begin{bmatrix} P_{H2} & 0 \\ 0 & P_{H2} \end{bmatrix} \cdot \begin{bmatrix} I & 0 \\ 0 & Q_K \end{bmatrix} \cdot [P_{K2}] \quad (10)$$

Define $[H_N] = [P_{H1}^{-1} \cdot K_N \cdot P_{H2}^{-1}]$. (11)

Then we have

$$[H_N] = [P_{H1}^{-1}] \cdot [P_{K1}] \cdot \begin{bmatrix} I & 0 \\ 0 & 2L_K \end{bmatrix} \cdot \begin{bmatrix} P_{H1} & 0 \\ 0 & P_{H1} \end{bmatrix} \cdot \begin{bmatrix} H_{\frac{N}{2}} & 0 \\ 0 & H_{\frac{N}{2}} \end{bmatrix} \cdot \begin{bmatrix} P_{H2} & 0 \\ 0 & P_{H2} \end{bmatrix} \cdot \begin{bmatrix} I & 0 \\ 0 & \frac{1}{2}Q_K \end{bmatrix} \cdot [P_{K2}] \cdot [P_{H2}^{-1}] \quad (12)$$

We can demonstrate that:

$$[P_{H1}^{-1}] \cdot [P_{K1}] \cdot \begin{bmatrix} I & 0 \\ 0 & 2L_K \end{bmatrix} \cdot \begin{bmatrix} P_{H1} & 0 \\ 0 & P_{H1} \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & K_H \end{bmatrix} \quad (13)$$

$$\begin{bmatrix} P_{H2} & 0 \\ 0 & P_{H2} \end{bmatrix} \cdot \begin{bmatrix} I & 0 \\ 0 & \frac{1}{2}Q_K \end{bmatrix} \cdot [P_{K2}] \cdot [P_{H2}^{-1}] = \begin{bmatrix} I & 0 \\ 0 & Q_H \end{bmatrix} \cdot \begin{bmatrix} I & I \\ I & -I \end{bmatrix} \quad (14)$$

Hence $[H_N] = \begin{bmatrix} I & 0 \\ 0 & K_H \end{bmatrix} \cdot \begin{bmatrix} H_{\frac{N}{2}} & 0 \\ 0 & H_{\frac{N}{2}} \end{bmatrix} \cdot \begin{bmatrix} I & 0 \\ 0 & Q_H \end{bmatrix} \cdot \begin{bmatrix} I & I \\ I & -I \end{bmatrix}$ (15)

From the example above, we can summarize the general formula for constructing new fast DCT algorithms below.

The recursive kernel is: $[DCT_N] = [A] \cdot \begin{bmatrix} DCT_{\frac{N}{2}} & 0 \\ 0 & DCT_{\frac{N}{2}} \end{bmatrix} \cdot [B]$ (16)

Define the new recursive kernel $[New_N]$ with the length of N:

$$[New_N] = [S^{-1}] \cdot [DCT_N] \cdot [T^{-1}]$$

Multiply the matrix $[S^{-1}]_N$ on the left of $[\mathrm{DCT}_N]$ and $[T^{-1}]_N$ on the right, then we have:

$$[New_N] = [S^{-1}] \cdot [A] \cdot \begin{bmatrix} S & 0 \\ 0 & S \end{bmatrix} \cdot \begin{bmatrix} New_{\frac{N}{2}} & 0 \\ 0 & New_{\frac{N}{2}} \end{bmatrix} \cdot \begin{bmatrix} T & 0 \\ 0 & T \end{bmatrix} \cdot [B] \cdot [T^{-1}] \quad (17)$$

This method can produce as many fast DCT algorithms as possible.

2.2. Computational Complexity

From the viewpoint of matrix factorization, it is easy to show how data flows and compute the complexity. Take Hou's algorithm for example.

The recursive kernel is Formula (3):

$$H_N = \begin{bmatrix} I & 0 \\ 0 & K_H \end{bmatrix} \cdot \begin{bmatrix} H_{\frac{N}{2}} & 0 \\ 0 & H_{\frac{N}{2}} \end{bmatrix} \cdot \begin{bmatrix} I & 0 \\ 0 & Q_H \end{bmatrix} \cdot \begin{bmatrix} I & I \\ I & -I \end{bmatrix}$$

Suppose that $H_{\frac{N}{2}}$ requires $M_{\frac{N}{2}}$ multiplications and $A_{\frac{N}{2}}$ additions for transformation, we can then find the numbers of the operations:

i) Multiplications:

The 2nd matrix requires $2M_{\frac{N}{2}}$ and the 3rd requires $\frac{N}{2}$ multiplications because of $\frac{N}{2}$ diagonal trigonometric coefficients, so the number of multiplications for H_N is $2M_{\frac{N}{2}} + \frac{N}{2}$, which follows $M_N = 2M_{\frac{N}{2}} + \frac{N}{2}$, with $M_4 = 4$.

By mathematical induction, we have $M_N = \frac{N}{2}\log_2 N$.

ii) Additions:

The first matrix requires $\frac{N}{2} - 1$ additions, because K_H is a lower triangular matrix and enjoys the recursive property. The second matrix requires $2A_{\frac{N}{2}}$ additions and the last matrix requires N additions. So $A_N = 2A_{N-1} + \frac{3N}{2} - 1$, with $A_4 = 9$.

By mathematical induction, we get $A_N = \frac{3N}{2}\log_2 N - N + 1$.

The intrinsic regularity provides the new possibilities for fast DCT algorithms. However, the best one always achieves the lowest computational complexity and the simplest structure for programming and hardware implementation, which is the guide for us to design the new algorithms.

However, not all the methods built in this way require the same numbers of additions and multiplications. As long as the new matrix appended to the recursive kernel of DCT is permutation matrix, which follows that it does not bring in any more additions or multiplications, the new algorithms achieved in this way can keep the same computational complexity as the original one.

3. THE ORTHOGONAL PROPERTY

DCT matrix enjoys the property of orthogonality, which follows that the inverse of the matrix equals to its transpose. We can make use of this property to explore more new possibilities of fast DCT algorithms. According to the structure deduced with matrix factorization, we learn that once the recursive kernel is orthogonal, there must be a twin fast DCT algorithm of it.

Suppose the matrix factorization form of the recursive kernel of a fast DCT algorithm is:

$$\left[K_N\right]=\left[A\right]\cdot\left[B\right]\cdot\begin{bmatrix}K_{\frac{N}{2}} & 0\\ 0 & K_{\frac{N}{2}}\end{bmatrix}\cdot\left[C\right]\cdot\left[D\right]. \tag{18}$$

By applying the transforms of inverse and transpose to the both sides of Equation (18), we have:

$$\left[K_N\right]^{-1*T}=\left[A\right]^{-1*T}\cdot\left[B\right]^{-1*T}\cdot\begin{bmatrix}K_{\frac{N}{2}}^{-1*T} & 0\\ 0 & K_{\frac{N}{2}}^{-1*T}\end{bmatrix}\cdot C^{-1*T}\cdot D^{-1*T} \tag{19}$$

As the matrix of K_N is orthogonal, we have $\left[K_N\right]=\left[K_N\right]^{-1*T}$. Thus, we find the twin fast DCT algorithm of the original one as

$$\left[K_N\right]=\left[A\right]^{-1*T}\cdot\left[B\right]^{-1*T}\cdot\begin{bmatrix}K_{\frac{N}{2}} & 0\\ 0 & K_{\frac{N}{2}}\end{bmatrix}\cdot\left[C\right]^{-1*T}\cdot\left[D\right]^{-1*T} \tag{20}$$

We take Lee's algorithm as an example. From Equation (4), the matrix factorization form of Lee's recursive kernel is:

$$\left[\hat{IDCT}_N\right]=\left[P_{L1}\right]\cdot\begin{bmatrix}I & 0\\ 0 & Q_L\end{bmatrix}\cdot\begin{bmatrix}\hat{IDCT}_{\frac{N}{2}} & 0\\ 0 & \hat{IDCT}_{\frac{N}{2}}\end{bmatrix}\cdot\left[P_{L2}\right] \tag{21}$$

and $IDCT_N=[X]\cdot\hat{IDCT}_N$, where

$$[X]=\begin{cases}\sqrt{\frac{2}{N}}\cdot\sqrt{\frac{1}{2}} & (n=0)\\ \sqrt{\frac{2}{N}} & (n=1,2,...,N-1)\end{cases} \tag{22}$$

(n is the parameter representing the column.)
So the twin fast DCT algorithm of Lee's is (23)

$$\hat{IDCT}_N=\left[P_{L1}\right]^{-1*T}\cdot\begin{bmatrix}I & 0\\ 0 & Q_L^{-1*T}\end{bmatrix}\cdot\begin{bmatrix}\hat{IDCT}_{\frac{N}{2}} & 0\\ 0 & \hat{IDCT}_{\frac{N}{2}}\end{bmatrix}\cdot\begin{bmatrix}X_{\frac{N}{2}}^{\ 2} & 0\\ 0 & X_{\frac{N}{2}}^{\ 2}\end{bmatrix}\cdot\left[P_{L2}\right]^{-1*T}\cdot\left[X_N\right]^{-1*2}$$

The matrix $\left[Q_L\right]$, whose size is 4 by 4, is:

$$\left[Q_L\right]_4=\begin{bmatrix}\frac{1}{2}\sec\frac{\pi}{16} & 0 & 0 & 0\\ 0 & \frac{1}{2}\sec\frac{3\pi}{16} & 0 & 0\\ 0 & 0 & \frac{1}{2}\sec\frac{5\pi}{16} & 0\\ 0 & 0 & 0 & \frac{1}{2}\sec\frac{7\pi}{16}\end{bmatrix} \tag{24}$$

$$\left[Q_L\right]_4^{-1*T}=\begin{bmatrix}2\cos\frac{\pi}{16} & 0 & 0 & 0\\ 0 & 2\cos\frac{3\pi}{16} & 0 & 0\\ 0 & 0 & 2\cos\frac{5\pi}{16} & 0\\ 0 & 0 & 0 & 2\cos\frac{7\pi}{16}\end{bmatrix} \tag{25}$$

$\left[Q_L\right]$ subjects Lee's algorithm to the stability problems with the inversion or division of cosine coefficients. However, in the twin algorithm of Lee's, there are no such problems.

The above example shows that the property of orthogonality not only provides new possibilities for those algorithms whose recursive kernels are orthogonal, but also improves the existing algorithms.

4. A NEW ALGORITHM

In this section, we present a new fast DCT algorithm, which is different from the algorithms we have seen in the literature and we name it "λ algorithm" according to the structure of its factor matrices.

We first define the matrix of recursive kernel New_N, then we have $\hat{DCT}_N=[P_\lambda]\cdot[New_N]$ (26)

and $[New_N]=[P_1]\cdot\begin{bmatrix}New_{\frac{N}{2}} & 0\\ 0 & New_{\frac{N}{2}}\end{bmatrix}\cdot\begin{bmatrix}I & 0\\ 0 & Q\end{bmatrix}\cdot[P_2]$ (27)

i) $[P_1]$ is a permutation matrix, which makes the two halves of the input data interlaced. For input sequence x of even length N, $\tilde{x}=[P_1]\cdot x$, where $x=[x_1,x_2,x_3,x_4,...,x_{N-1},x_N]$ and $\tilde{x}=[x_1,x_{\frac{N}{2}+1},x_2,x_{\frac{N}{2}+2},...,x_{\frac{N}{2}},x_N]$.

ii) $[Q]$ is a trigonometric matrix. In the DCT matrix for N-length sequence, the size of $[Q]$ is $M\times M$, where $M=\frac{N}{2}$.

$Q_M=diag[\cos\phi_m]$, $\phi_m=\frac{\pi(2m+1)}{2N}$, $m=0,1...M-1$.

iii) $[P_2]$ is an integer coefficient matrix for additions and subtractions of the input. For $z=[z_1,z_2,z_3,z_4,...,z_{N-1},z_N]$, $\tilde{z}=[P_2]\cdot z$ give $\tilde{z}=[z_1+z_N,z_2+z_{N-1},...,z_{\frac{N}{2}-1}+z_{\frac{N}{2}+1},z_1-z_N,z_2-z_{N-1},...,z_{\frac{N}{2}-1}-z_{\frac{N}{2}+1}]$.

iv) $[P_\lambda]_N$ is the multiplication of $\log_2 N-1$ matrices, if the length of the input sequence is N. Each factor matrix is sparse and the nonzero elements make the matrix look like the Greek letter λ.

Suppose: $\left[P_\lambda\right]_N=\left[P_{\lambda(\log_2 N-1)}\right]_N\cdots\left[P_{\lambda 2}\right]_N\cdot\left[P_{\lambda 1}\right]_N$.

For length $2N$, we have

$$\left[P_\lambda\right]_{2N}=\left[P_{\lambda_{\log_2 N}}\right]_{2N}\cdot\begin{bmatrix}\left(P_{\lambda_{\log_2 N-1}}\right)_N & 0\\ 0 & \left(P_{\lambda_{\log_2 N-1}}\right)_N\end{bmatrix}_{2N}\cdots\begin{bmatrix}(P_{\lambda_1})_N & 0\\ 0 & (P_{\lambda_1})_N\end{bmatrix}_{2N}$$

or

$$[P_\lambda]_{2N} = \left[P_{\lambda_{\log_2 N}}\right]_{2N} \cdot \begin{bmatrix} \left(P_{\lambda_{\log_2 N-1}}\right)_N & 0 \\ 0 & \left(P_{\lambda_{\log_2 N-1}}\right)_N \end{bmatrix}_{2N} \cdots \begin{bmatrix} \left(P_{\lambda_1}\right)_4 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \left(P_{\lambda_1}\right)_4 \end{bmatrix}_{2N} \quad (28)$$

where $\left[P_{\lambda(\log_2 N-1)}\right]_N = \begin{bmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&\ddots&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&-1&0&2&0&0 \\ 0&0&\iddots&0&0&0&\ddots&0 \\ 0&-1&0&0&0&0&0&2 \end{bmatrix}_N \quad (29)$

For example: $\left[P_\lambda\right]_4 = \left[P_{\lambda 1}\right]_4 = \begin{bmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&1&0 \\ 0&-1&0&2 \end{bmatrix}$

$$\left[P_\lambda\right]_8 = \left[P_{\lambda 2}\right]_8 \cdot \left[P_{\lambda 1}\right]_8 = \begin{bmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&-1&0&2&0&0 \\ 0&0&-1&0&0&0&2&0 \\ 0&-1&0&0&0&0&0&2 \end{bmatrix} \cdot \begin{bmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&-1&0&2&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&-1&2 \end{bmatrix}$$

The analysis of the computational complexity is given below.

i) Multiplications:

The operations of multiplications for the new fast DCT algorithm are from two parts: the recursive kernel and the matrix *Q*. For the *N*-length sequence, the number of multiplications with the recursive kernel is $M1_N$ and that with *Q* is $M2_N$. So the total number of the multiplications for the *N*-length sequence is $M_N = M1_N + M2_N$.

From (27), the number of multiplications with $\begin{bmatrix} New_{\frac{N}{2}} & 0 \\ 0 & New_{\frac{N}{2}} \end{bmatrix}$ is $2M1_{\frac{N}{2}}$ and that with $\begin{bmatrix} I & 0 \\ 0 & Q \end{bmatrix}$ is $\frac{N}{2}$. No multiplication is needed for $\left[P_\lambda\right]_N$. So $M_N = M1_N = 2M1_{\frac{N}{2}} + \frac{N}{2}$ with $M_4 = 4$. By mathematical induction, we have $M_N = \frac{N}{2}\log_2 N$.

ii) Additions:

The operations of additions are also from two parts: the recursive kernel and $\left[P_\lambda\right]_N$. For the *N*-length sequence, the number of additions from the recursive kernel is $A1_N$ and that from $\left[P_\lambda\right]_N$ is $A2_N$. So the number of the additions for the *N*-length sequence is $A_N = A1_N + A2_N$.

From (27), the number of additions with the second matrix is $2A1_{\frac{N}{2}}$ and that from $[P2]$ is *N*. So $A1_N = 2A1_{\frac{N}{2}} + N$, with $A1_4 = 8$. By mathematical induction, we have $A1_N = N\log_2 N$.

From (28), the number of additions in the recursive kernel is $A2_N = 2A2_{\frac{N}{2}} + \frac{N}{2} - 1$, with $A2_4 = 1$. By mathematical induction, we have $A2_N = \frac{N}{2}\log_2 N - N + 1$.

So $A_N = A1_N + A2_N = \frac{3N}{2}\log_2 N - N + 1$.

Table 2 Number of operations of the proposed algorithm

	N=4	N=8	N=16	N=32	N=64
Multiplication	4	12	32	80	192
Addition	9	29	81	209	513

From the deduction, we learn that the proposed algorithm requires $\frac{N}{2}\log_2 N$ multiplications and $\frac{3N}{2}\log_2 N - N + 1$ additions for N-length sequence, which are the same as other fastest DCT algorithms [1-2,8-10], but the new algorithm allows simpler hardware implementation, especially, parallel implementation.

5. REFERENCES

[1] H. S. Hou, "A Fast Algorithm For Computing the Discrete Cosine Transform," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-35, No. 10, pp.1455-1461, Oct. 1987.

[2] B. G. Lee, "A New Algorithm to Compute the Discrete Cosine Transform," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-32, NO. 6, pp.1243-1245, Dec. 1984.

[3] K.R. Rao and P. Yip, Discrete Cosine Transform-Algorithm, Advantage, Applications. New York: Academic, 1990.

[4] M.T. Heideman, "Computation of an odd-length DCT from a real-valued DFT of the same length," IEEE trans. Signal Process. vol. 40, NO.1, pp.54-61, Jan 1992.

[5] H. S. Malvar, "Corrections to 'Fast computation of the discrete cosine transform and the discrete Hartley transform,'" IEEE Trans. Acoust., Speech, Signal Processing, vol. 36, pp. 610-612, Apr. 1998.

[6] Z. D. Wang, "Reconsideration of 'A Fast Computational Algorithm for the Discrete Cosine Transform'," IEEE Trans. Commun., vol. COM-31, pp. 121-123,NO. 1, Jan 1983.

[7] 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, NO. 9,pp.1004-1009, Sep. 1997

[8] C. W. Kok, "Fast Algorithm for Computing Discrete Cosine Transform," IEEE Trans. Signal Process. , vol. 45, NO.3, pp.757-760, Mar. 1977.

[9] P. Lee and F.-Y. Huang, "Restructured Recursive DCT and DST Algorithms," IEEE Trans. Signal Process. vol. 42, NO. 7, pp.1600-1609, Jul. 1994

[10] Z. Cvetkovic and M. V. Popovic, "New Fast Recursive Algorithms for the Computation of Discrete Cosine and Sine Transforms," IEEE Trans. Signal Process., vol. 40, NO. 8, pp.2083-2086, Aug. 1992.

[11] E. Feig and S.Winograd, "Fast Algorithm for the Discrete Cosine Transform," IEEE Trans. Signal Process. vol. 40, NO. 9, pp.2174-2193, Sep. 1992.

[12] G. Plonka and M. Tasche, "Fast and numerically stable algorithms for discrete cosine transform," Linear Algebra and its Application, 394 (2005), pp.309-345, 2005.