Integer DCTs and Fast Algorithms

Yonghong Zeng, Lizhi Cheng, Guoan Bi, and Alex C. Kot, *Senior Member, IEEE*

***Abstract*—A method is proposed to factor the type-II discrete cosine transform (DCT-II) into lifting steps and additions. After approximating the lifting matrices, we get a new type-II integer discrete cosine transform (IntDCT-II) that is float-point multiplication free. Based on the relationships among the various types of DCTs, we can generally factor any DCTs into lifting steps and additions and then get four types of integer DCTs, which need no float-point multiplications. By combining the polynomial transform and the one-dimensional (1-D) integer cosine transform, a two-dimensional (2-D) integer discrete cosine transform is proposed. The proposed transform needs only integer operations and shifts. Furthermore, it is nonseparable and requires a far fewer number of operations than that used by the corresponding row-column 2-D integer discrete cosine transform.**

***Index Terms*—Data compression, discrete cosine transform, fast algorithm, integer transform, mobile.**

I. Introduction

DISCRETE cosine transforms (DCTs) have a wide range of applications such as data compression, feature extraction, multiframe detection, and filterbanks [1], [11], [17]–[19]. However, float-point multiplications are inevitable for implementing such transforms, which prevents them from being widely used in areas such as mobile devices and lossless compression. In mobile devices, the power consumption used for computation, especially for float-point multiplications, cannot be neglected. Since there exist errors for quantizing the transforming coefficients, it is impossible to use them for lossless compression. Therefore, it is not surprising that lossless coding schemes are hardly based on the DCTs. Therefore, transforms without float-point multiplications or integer transforms have been studied in recent years [4], [5], [7], [13], [15], [16], [20]–[22]. Among them, the so-called second generation wavelets based on the lifting scheme [4], [20] and the eight-point and 16-point integer DCT [7], [16], [21] have been proposed. In addition, there have been other integer transforms [5], [15]. Generally speaking, integer transforms possess some features of their corresponding float-point transforms such as the decorrelation property, whereas their computational cost is less. Since integer transforms require only integer arithmetic (additions and possibly multiplications), their implementation is greatly simplified. In addition, if sufficient word length is used to represent the intermediate data, the round-off error can completely be eliminated. Therefore, they have been applied for image coding, filterbanks, and other areas [6], [7], [13], [16], [21]. For example, using the integer DCT based on lifting scheme for image compression, the compression ratio is comparable with that using the original DCT, whereas the computational complexity is substantially reduced [16]. Furthermore, it can be used for lossless compression [16], [21]. In the JPEG-2000 proposal, the use of the integer discrete transform for lossless image coding was recommended [12].

Although some work has been done for integer cosine transforms, many problems still exist. For example, although [3], [4], and [20] have given methods to factor a matrix with determinant 1 and order N into products of lifting matrices, it is not practical to use them for the DCT matrix since they generally need $O(N^2)$ lifting matrices in the factorization, which is unacceptable for real signal processing. In [7], [16], and [21], eight-point and 16-point integer DCT-IIs are derived using the Walsh–Hadamard transform of DCT matrix given in [22]. However, because of the difficulty in finding the factorization of DCT matrix based on the Walsh–Hadamard transform, it is unlikely to generalize their method to producing integer DCTs of larger sizes. Finding integer DCTs of arbitrary length with low complexity still remains unsolved. The integer transforms in [5] and [15] lack the fast algorithm and are generally not the numerical approximation of the original transforms. Furthermore, [5] and [15] also do not give a unified method for generating integer transforms with various length. In this paper, we first present an efficient method for factoring the transform matrix of DCT-II into products of lifting matrices and simple matrices. Then, we give methods for factoring any kind of DCTs based on the relationships among them. As a result, we get four types of new integer DCTs that need no float-point multiplications. For the two-dimensional (2-D) case, tensor products of one-dimensional (1-D) integer transforms is the method currently used to construct 2-D integer transform, that is, a 2-D input array is processed by implementing the 1-D transform on its rows and columns consecutively (also called row-column method). However, other means are possible and sometimes more efficient. For example, in wavelet transforms, we know that sometimes, a nonseparable 2-D transform may be more useful for 2-D problems than the separable ones [19]. Therefore, in this paper, we propose a 2-D nonseparable integer discrete cosine transform by combining the polynomial transform [14] and the 1-D integer cosine transform. The proposed transform needs only integer additions and shifts. Furthermore, it uses far fewer operations than the corresponding tensor-product 2-D integer discrete cosine transform does.

Manuscript received July 17, 2000; revised July 25, 2001. The associate editor coordinating the review of this paper and approving it for publication was Dr. Xiang-Gen Xia.

Y. Zeng is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, and the National University of Defense Technology, Changsha, China (e-mail: eyhzeng@ntu.edu.sg).

L. Cheng is with the School of Science, National University of Defense Technology, Changsha, China (e-mail: lzchen@nudt.edu.cn).

G. Bi and A. C. Kot are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore (e-mail: egbi@ntu.edu.sg; eackot@ntu.edu.sg).

Publisher Item Identifier S 1053-587X(01)09219-4.

The paper is organized as follows. Since lifting matrix is a major tool for our purpose, we first give the definitions and properties of the lifting matrix and lifting step in Section II. Then, we propose the integer DCT-II and its fast algorithm in Section III. Based on the relationships among DCTs, we propose the integer DCT-IV and DCT-I their fast algorithms in Section IV. By combining the polynomial transform and the 1-D integer transform, a nonseparable 2-D integer discrete cosine transform is proposed in Section V. Comparison with the tensor-product 2-D integer transform is also given in this section. A concise comparison between IntDCT and DCT is given in Section VI.

II. Lifting Matrix, Lifting Step, and Properties

The lifting matrix is a major tool for constructing integer wavelet and integer discrete transforms [4], [7], [20], [21]. We first give its definition and some simple properties in the following.

Definition 1: A lifting matrix is a matrix whose diagonal elements are 1s, and only one nondiagonal element is nonzero [4], [20]. If the order of a lifting matrix is N, we use the notation $L_{i,j}(s)$ $(i \neq j)$ to denote the lifting matrix whose only nonzero element is at the ith row and the jth column $(i, j = 0, 1, \ldots, N-1)$ and whose nondiagonal nonzero element is s.

For example, matrix

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 2.3 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

is a lifting matrix, and based on Definition 1, we can use the notation $L_{2,1}$ (2.3) to denote it. To multiply a lifting matrix with a vector, say, $y = L_{i,j}(s)x$, where x and y are column vector of length N (the order of the lifting matrix), we have

$$y(i) = x(i) + sx(j), \quad y(k) = x(k), \quad k \neq i \tag{2.2}$$

which is illustrated in Fig. 1.

Definition 2: A lifting step is multiplying a lifting matrix with a vector, which is shown in (2.2).

A distinguished feature of lifting matrix is that its inverse is still a lifting matrix with the same shape. In fact, we have

$$L_{i,j}^{-1}(s) = L_{i,j}(-s). \tag{2.3}$$

Therefore, if a matrix can be factored into products of lifting matrices, its inverse is also products of lifting matrices. In (2.2), float-point multiplication is needed if s is an irrational number or even a rational number with unlimited digits. In this case, we should approximate s by another number. For easy realization, the number is desired to be of the form $\beta/2^{\lambda}$, where β and λ are integers.

Definition 3: The notation $\mathrm{RB}(s)$ is used to denote a number that is of the form $\beta/2^{\lambda}$ (dyadic rational number) and approximates to the real number s.

$\mathrm{RB}(s)$ is not uniquely depended on s. It depends on the required accuracy of the approximation. For example, 2.3 in (2.1) can be approximated by 2, $5/2, 9/4$, or other numbers. When

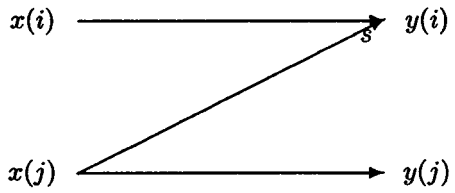

Fig. 1. Flowchart of a lifting step.

s is approximated by $\mathrm{RB}(s)$, the matrix $L_{i,j}(s)$ is then approximated by $L_{i,j}(\mathrm{RB}(s))$, which is still invertible and whose inverse is $L_{i,j}(-\mathrm{RB}(s))$.

We can also approximate the lifting step by using a nonlinear transform. For example, y can be approximated by $\hat{y}$, which is defined as

$$\hat{y}(i) = x(i) + \lfloor sx(j) \rfloor, \quad \hat{y}(k) = x(k), \quad k \neq i. \tag{2.4}$$

This transform is nonlinear! In addition, it maps integer into integer! The nonlinear transform is invertible, and its inverse is as follows:

$$x(i) = \hat{y}(i) - \lfloor s\hat{y}(j) \rfloor, \quad x(k) = \hat{y}(k), \quad k \neq i. \tag{2.5}$$

Therefore, when a transform is factored into lifting steps, it is easy to approximate it by another transform, which needs no floating-point multiplications or even by an integer-to-integer nonlinear transform. Furthermore, the resultant transform is invertible, and its inverse needs no floating-point multiplications as well. That is why we want to factor DCT into lifting steps.

Theoretically, any matrix with determinant 1 can be factored into products of some lifting matrices [3], [4], [20]. However, generally, $O(N^2)$ lifting matrices are needed in the factorization for a matrix of order N, which is not acceptable for signal processing. For special matrices, such as the transform matrices of DCT-IIs, in order to reach fast realization, we want to factor them into products of at most $O(N \log_2 N)$ lifting matrices. In [7] and [21], such factorizations have been achieved for DCT-II matrix of order 8 and 16, and eight-point and 16-point integer DCT-IIs are then presented. However, their methods are based on the Walsh–Hadamard transform of the DCT matrix given in [22]. It is unlikely to generalize them to producing integer DCTs of larger sizes. In this paper, we will propose an efficient method that can give the factorization of any DCT transform matrix with order $N = 2^t$. The following lemma is crucial for our purpose.

Lemma 1: Assume that D is a diagonal matrix with determinant 1, say, $D = \mathrm{diag}(b_0, b_1, \ldots, b_{L-1})$ where $b_0 b_1 \ldots b_{L-1} = 1$. Let $\alpha_0 = b_0$, $\alpha_k = \alpha_{k-1} b_k$, $k = 1, 2, \ldots, L-1$. Then $D = B_1 B_2 = B_2 B_1$, where

$$B_1 = \mathrm{diag}\left(\alpha_0, \frac{1}{\alpha_0}, \alpha_2, \frac{1}{\alpha_2}, \ldots, \alpha_{L-2}, \frac{1}{\alpha_{L-2}}\right)$$
$$B_2 = \mathrm{diag}\left(1, \alpha_1, \frac{1}{\alpha_1}, \alpha_3, \frac{1}{\alpha_3}, \ldots, 1\right)$$

if L is even, and

$$B_1 = \mathrm{diag}\left(\alpha_0, \frac{1}{\alpha_0}, \alpha_2, \frac{1}{\alpha_2}, \ldots, 1\right)$$
$$B_2 = \mathrm{diag}\left(1, \alpha_1, \frac{1}{\alpha_1}, \alpha_3, \frac{1}{\alpha_3}, \ldots, \alpha_{L-2}, \frac{1}{\alpha_{L-2}}\right)$$

if L is odd. The matrices B_1 and B_2 can be factored as products of lifting matrices, respectively.

Proof: Based on the definition of D, B_1, and B_2, it is easy to verify that $D = B_1B_2 = B_2B_1$. To factor B_1 and B_2 into lifting matrices, we first notice that they can be expressed as block diagonal matrices, where each block is of the form

$$\begin{bmatrix} c & 0 \\ 0 & \frac{1}{c} \end{bmatrix}.$$

The matrix of order 2 can be expressed as product of lifting matrices:

$$\begin{bmatrix} c & 0 \\ 0 & \frac{1}{c} \end{bmatrix} = \begin{bmatrix} 1 & c-1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{c}-1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -c & 1 \end{bmatrix}.$$

Therefore, B_1 and B_2 can also be factored into products of lifting matrices. ♣

III. Integer DCT-II, DCT-III, and Fast Algorithm

Let $x(n)$ $(n = 0, 1, \ldots, N-1)$ be a real input sequence. We assume that $N = 2^t$, where $t > 0$. The scaled DCT-II of $x(n)$ is defined as follows:

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

Let C_N^{II} be the transform matrix of the DCT-II, that is

$$C_N^{\mathrm{II}} = \left(\cos \frac{\pi k(2n+1)}{2N} \right)_{k,n=0,1,\ldots,N-1}.$$

In [9] and [23], a fast algorithm is proposed for the scaled DCT-II, which uses $(1/2)N \log_2 N$ multiplications and $(3/2)N \log_2 N - N + 1$ additions. According to its number of operations, the algorithm is among the best of all algorithms for the DCT-II. Other algorithms using the same number of operations include Lee's and Hou's algorithms in [8] and [10]. Based on the algorithm, we can get a factorization of the transform matrix.

Lemma 2: The transform matrix C_N^{II} can be factored as

$$C_N^{\mathrm{II}} = P_N \begin{bmatrix} I_{N/2} & 0 \\ 0 & U_{N/2} \end{bmatrix} \begin{bmatrix} C_{N/2}^{\mathrm{II}} & 0 \\ 0 & C_{N/2}^{\mathrm{II}} \end{bmatrix} \times \begin{bmatrix} I_{N/2} & 0 \\ 0 & D_{N/2} \end{bmatrix} \begin{bmatrix} I_{N/2} & \hat{I}_{N/2} \\ I_{N/2} & -\hat{I}_{N/2} \end{bmatrix} \tag{3.1}$$

where $I_{N/2}$ is the identity matrix of order $N/2$, $\hat{I}_{N/2}$ is the matrix by reversing the rows of $I_{N/2}$, $D_{N/2}$ is a diagonal matrix defined by

$$D_{N/2} = \operatorname{diag}\left(2\cos\frac{\pi}{2N}, 2\cos\frac{3\pi}{2N}, \ldots, 2\cos\frac{(N-1)\pi}{2N} \right) \tag{3.2}$$

$$U_{N/2} = \begin{bmatrix} \frac{1}{2} & 0 & 0 & \cdots & \cdots & 0 & 0 \\ -\frac{1}{2} & 1 & 0 & \cdots & \cdots & 0 & 0 \\ \frac{1}{2} & -1 & 1 & \cdots & \cdots & 0 & 0 \\ & \cdots & & & & \cdots & \\ \frac{1}{2} & -1 & 1 & \cdots & \cdots & 1 & 0 \\ -\frac{1}{2} & 1 & -1 & \cdots & \cdots & -1 & 1 \end{bmatrix} \tag{3.3}$$

P_N is a permutation matrix defined by

$$P_N = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & 1 & \cdots & 0 \\ & & \cdots & & & \cdots & & \\ 0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \tag{3.4}$$

and $C_{N/2}^{\mathrm{II}}$ is the transform matrix of the scaled DCT-II with length $N/2$.

Float-point multiplications are needed for the algorithm when the matrix $D_{N/2}$ is multiplied by a vector. In order to avoid float multiplications, we want to turn this matrix into products of lifting matrices and then approximate the elements of the lifting matrices by numbers that are of the form $\beta/2^{\lambda}$, where β and λ are integers. To use Lemma 1, we turn the matrix $D_{N/2}$ into $D_{N/2} = E_{N/2}F_{N/2}$, where

$$E_{N/2} = \operatorname{diag}(\sqrt{2}, 1, \ldots, 1) \tag{3.5}$$

$$F_{N/2} = \operatorname{diag}\left(\sqrt{2}\cos\frac{\pi}{2N}, 2\cos\frac{3\pi}{2N}, \ldots, 2\cos\frac{(N-1)\pi}{2N} \right). \tag{3.6}$$

Lemma 3: The determinant of the matrix $F_{N/2}$ is 1, that is, $\det(F_{N/2}) = 1$.

Proof: Based on the mathematical induction, the conclusion can be easily verified. ♣

Based on Lemma 1, we know that $F_{N/2}$ can be factored into the product of lifting matrices. More accurately, we have

$$\begin{aligned} F_{N/2} = {} & \left\{ \prod_{k=0}^{N/4-1} \left[L_{2k,2k+1}(\alpha_{2k} - 1) L_{2k+1,2k}(1) \times L_{2k,2k+1}\left(\frac{1}{\alpha_{2k}} - 1 \right) L_{2k+1,2k}(-\alpha_{2k}) \right] \right\} \\ & \cdot \left\{ \prod_{k=1}^{N/4-1} \left[L_{2k-1,2k}(\alpha_{2k-1} - 1) L_{2k,2k-1}(1) \times L_{2k-1,2k}\left(\frac{1}{\alpha_{2k-1}} - 1 \right) L_{2k,2k-1}(-\alpha_{2k-1}) \right] \right\} \end{aligned} \tag{3.7}$$

where $L_{i,j}(s)$ are lifting matrices with order $N/2$ as defined in Definition 1, and

$$\alpha_0 = \sqrt{2}\cos\frac{\pi}{2N}, \quad \alpha_k = \alpha_{k-1} 2\cos\frac{(2k+1)\pi}{2N} \quad k = 1, 2, \ldots, N/2 - 1. \tag{3.8}$$

In order to avoid float multiplications, we now approximate the nonzero element of the lifting matrices by numbers that are of the form $\beta/2^{\lambda}$, where β and λ are integers, that is, we replace every nonzero element s by $\mathrm{RB}(s)$ as defined in Definition 3. Therefore, we replace α_j and $1/\alpha_j$ in (3.7) by $\mathrm{RB}(\alpha_j)$ and

$\mathrm{RB}(1/\alpha_j)$ and get an approximating matrix for $F_{N/2}$ as

$$\begin{aligned}\bar{F}_{N/2} = &\left\{\prod_{k=0}^{N/4-1}\left[L_{2k,2k+1}(\mathrm{RB}(\alpha_{2k})-1)L_{2k+1,2k}(1)\right.\right.\\ &\times L_{2k,2k+1}\left(\mathrm{RB}\left(\frac{1}{\alpha_{2k}}\right)-1\right)L_{2k+1,2k}\\ &\left.\left.\times(-\mathrm{RB}(\alpha_{2k}))\right]\right\}\\ &\cdot\left\{\prod_{k=1}^{N/4-1}\left[L_{2k-1,2k}(\mathrm{RB}(\alpha_{2k-1})-1)L_{2k,2k-1}(1)\right.\right.\\ &\cdot L_{2k-1,2k}\left(\mathrm{RB}\left(\frac{1}{\alpha_{2k-1}}\right)-1\right)\\ &\left.\left.\times L_{2k,2k-1}(-\mathrm{RB}(\alpha_{2k-1}))\right]\right\}. \qquad (3.9)\end{aligned}$$

We also approximate the matrix $E_{N/2}$ by

$$\bar{E}_{N/2} = \mathrm{diag}(\mathrm{RB}(\sqrt{2}), 1, \ldots, 1). \qquad (3.10)$$

Then, the correspondent approximating matrix for the transform matrix C_N^{II} is

$$\begin{aligned}\bar{C}_N^{\mathrm{II}} = P_N &\begin{bmatrix} I_{N/2} & 0 \\ 0 & U_{N/2}\end{bmatrix}\begin{bmatrix}\bar{C}_{N/2}^{\mathrm{II}} & 0 \\ 0 & \bar{C}_{N/2}^{\mathrm{II}}\end{bmatrix}\\ &\times\begin{bmatrix} I_{N/2} & 0 \\ 0 & \bar{D}_{N/2}\end{bmatrix}\begin{bmatrix} I_{N/2} & \hat{I}_{N/2} \\ I_{N/2} & -\hat{I}_{N/2}\end{bmatrix} \qquad (3.11)\end{aligned}$$

where

$$\bar{D}_{N/2} = \bar{E}_{N/2}\bar{F}_{N/2}. \qquad (3.12)$$

If the same method is used to factor the matrix $\bar{C}_{N/2}^{\mathrm{II}}$ recursively until the order is 1, we get the complete factorization of $\bar{C}_N^{\mathrm{II}}$. The matrix $\bar{C}_N^{\mathrm{II}}$ defines a new transform that does not need float multiplications. As in [7] and [21], we call it a type-II integer discrete cosine transform (IntDCT-II).

Definition 4: Assume that $N = 2^t$. The transform matrix of a type-II integer discrete cosine transform (IntDCT-II) $\bar{C}_N^{\mathrm{II}}$ is defined recursively by $\bar{C}_1^{\mathrm{II}} = (1)$, and

$$\begin{aligned}\bar{C}_{2^j}^{\mathrm{II}} = P_{2^j}&\begin{bmatrix} I_{2^{j-1}} & 0 \\ 0 & U_{2^{j-1}}\end{bmatrix}\begin{bmatrix}\bar{C}_{2^{j-1}}^{\mathrm{II}} & 0 \\ 0 & \bar{C}_{2^{j-1}}^{\mathrm{II}}\end{bmatrix}\\ &\times\begin{bmatrix} I_{2^{j-1}} & 0 \\ 0 & \bar{D}_{2^{j-1}}\end{bmatrix}\begin{bmatrix} I_{2^{j-1}} & \hat{I}_{2^{j-1}} \\ I_{2^{j-1}} & -\hat{I}_{2^{j-1}}\end{bmatrix}\\ &j = 1, 2, \ldots, t. \qquad (3.13)\end{aligned}$$

The transform is not unique. Actually, any choice of function RB determines a transform. Based on the definition, we get a fast algorithm for the IntDCT-II as follows.

Algorithm 1—Fast Algorithm for IntDCT-II:

Step 1) Compute

$$\begin{aligned}g(n) &= x(n) + x(N-1-n)\\ h(n) &= x(n) - x(N-1-n), \quad n = 0, 1, \ldots, N/2-1.\end{aligned}$$

Step 2) Compute $\hat{h} = \bar{E}_{N/2}\bar{F}_{N/2}h$, where

$$\begin{aligned}\hat{h} &= (\hat{h}(0), \hat{h}(1), \ldots, \hat{h}(N/2-1))'\\ h &= (h(0), h(1), \ldots, h(N/2-1))'.\end{aligned}$$

Step 3) Compute the IntDCT-II with length $N/2$ of sequence $g(n)$ and $\hat{h}(n)$, and let the outputs be $G(k)$ and $H(k)$, respectively. The Steps 1 and 2 can be used recursively for the computations.

Step 4) Compute

$$\begin{aligned}X(2k) &= G(k), \quad k = 0, 1, \ldots, N/2-1\\ X(1) &= H(0)/2, \quad X(2k+1) = H(k) - X(2k-1)\\ &\qquad k = 1, 2, \ldots, N/2-1.\end{aligned}$$

Now, we consider the computational complexity of Algorithm 1.

Lemma 4: The computation of Step 2 in Algorithm 1 needs $3((N/2)-1)$ lifting steps (see Definition 2), $(N/2)-1$ additions, and one integer multiplication.

Proof: From (3.9), we know that $\bar{F}_{N/2}$ is factored into $4((N/2)-1)$ lifting matrices. However, among them, $(N/2)-1$ matrices are of the form $L_{i,j}(1)$, that is, the nonzero nondiagonal element is 1. Such a lifting step needs only one addition. Therefore, we need $3((N/2)-1)$ lifting steps and $(N/2)-1$ additions to realize the operation of multiplying $\bar{F}_{N/2}$ with a vector. To multiply $\bar{E}_{N/2}$ with a vector needs one integer multiplication. Therefore, the Lemma is true. ♣

Let $\mathrm{LC}^{\mathrm{II}}(N)$ and $\mathrm{AC}^{\mathrm{II}}(N)$ represent the number of lifting steps and additions for computing an Int-DCT-II with length N. Then, from Lemma 4 and Algorithm 1, we get

$$\begin{aligned}\mathrm{LC}^{\mathrm{II}}(N) &= 2\mathrm{LC}^{\mathrm{II}}\left(\frac{N}{2}\right) + 3\left(\frac{N}{2}-1\right)\\ \mathrm{AC}^{\mathrm{II}}(N) &= 2\mathrm{AC}^{\mathrm{II}}\left(\frac{N}{2}\right) + 2(N-1).\end{aligned}$$

Therefore, we finally get

$$\begin{aligned}\mathrm{LC}^{\mathrm{II}}(N) &= \frac{3}{2}N\log_2 N - 3N + 3\\ \mathrm{AC}^{\mathrm{II}}(N) &= 2N\log_2 N - 2N + 2. \qquad (3.14)\end{aligned}$$

Apart from the lifting steps and the additions, $N-1$ integer multiplications are also needed. When $N = 16$, 51 lifting steps are needed, which is the same as that used in [7], [21], and [22].

The matrix $\bar{C}_N^{\mathrm{II}}$ is invertible, and its inverse matrix can also be factored as products of lifting matrices and simple matrices whose elements are 0 or 1 or 1/2. In fact, we have

$$\begin{aligned}\left(\bar{C}_N^{\mathrm{II}}\right)^{-1} = &\begin{bmatrix}\frac{1}{2}I_{N/2} & \frac{1}{2}I_{N/2} \\ \frac{1}{2}\hat{I}_{N/2} & -\frac{1}{2}\hat{I}_{N/2}\end{bmatrix}\begin{bmatrix} I_{N/2} & 0 \\ 0 & \bar{F}_{N/2}^{-1}\bar{E}_{N/2}^{-1}\end{bmatrix}\\ &\times\begin{bmatrix}\left(\bar{C}_{N/2}^{\mathrm{II}}\right)^{-1} & 0 \\ 0 & \left(\bar{C}_{N/2}^{\mathrm{II}}\right)^{-1}\end{bmatrix}\\ &\times\begin{bmatrix} I_{N/2} & 0 \\ 0 & U_{N/2}^{-1}\end{bmatrix}P_N^{-1} \qquad (3.15)\end{aligned}$$

where

$$U_{N/2}^{-1} = \begin{bmatrix} 2 & 0 & 0 & \cdots & \cdots & 0 & 0 \\ 1 & 1 & 0 & \cdots & \cdots & 0 & 0 \\ 0 & 1 & 1 & \cdots & \cdots & 0 & 0 \\ & \cdots & & & & \cdots & \\ 0 & 0 & 0 & \cdots & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & \cdots & 1 & 1 \end{bmatrix} \tag{3.16}$$

$$\begin{aligned} \bar{F}_{N/2}^{-1} = & \left\{ \prod_{k=1}^{N/4-1} \Big[L_{2k,2k-1}(\mathrm{RB}(\alpha_{2k-1})) \right. \\ & \times L_{2k-1,2k}\left(-\mathrm{RB}\left(\frac{1}{\alpha_{2k-1}}\right)+1\right) L_{2k,2k-1}(-1) \\ & \left. \cdot L_{2k-1,2k}(-\mathrm{RB}(\alpha_{2k-1})+1) \Big] \right\} \\ & \times \left\{ \prod_{k=0}^{N/4-1} \Big[L_{2k+1,2k}(-\mathrm{RB}(\alpha_{2k})) L_{2k,2k+1} \right. \\ & \times \left(-\mathrm{RB}\left(\frac{1}{\alpha_{2k}}\right)+1\right) \cdot L_{2k+1,2k}(-1) L_{2k,2k+1} \\ & \left. \times (-\mathrm{RB}(\alpha_{2k})+1) \Big] \right\} \end{aligned} \tag{3.17}$$

$$\bar{E}_{N/2}^{-1} = \mathrm{diag}\left(\frac{1}{\mathrm{RB}(\sqrt{2})}, 1, \ldots, 1\right). \tag{3.18}$$

We know that the inverse of DCT-II is DCT-III. Therefore, we define an integer DCT-III (**IntDCT-III**) by giving its transform matrix as $(\bar{C}_N^{\mathrm{II}})^{-1}$ in (3.15). Based on the factorization (3.15), we get a fast algorithm for IntDCT-III (reconstruction algorithm for IntDCT-II) as follows, where $X(k)$ is the input, and $x(n)$ is the output.

Algorithm 2—Fast Algorithm for IntDCT-III:

Step 1) Compute

$$\begin{aligned} G(k) &= X(2k), \quad k = 0, 1, \ldots, N/2-1 \\ H(0) &= 2X(1), \quad H(k) = X(2k+1) + X(2k-1) \\ & \qquad k = 1, 2, \ldots, N/2-1. \end{aligned}$$

Step 2) Compute the inverse IntDCT-II of sequence $G(k)$ and $H(k)$, and let the outputs be $g(n)$ and $h(n)$, respectively. Steps 1, 3, and 4 can be used recursively for the computations.

Step 3) Compute $p = \bar{F}_{N/2}^{-1} \bar{E}_{N/2}^{-1} h$, where

$$\begin{aligned} p &= (p(0), p(1), \ldots, p(N/2-1))' \\ h &= (h(0), h(1), \ldots, h(N/2-1))'. \end{aligned}$$

Step 4) Compute

$$\begin{aligned} x(n) &= (g(n)+p(n))/2, \quad x(N-1-n) = (g(n)-p(n))/2 \\ & \qquad n = 0, 1, \ldots, \frac{N}{2}-1. \end{aligned}$$

The number of operations for reconstruction is the same as that for the forward transform. It should be noted that although $(C_N^{\mathrm{II}})^{-1}$ is not much different from $(C_N^{\mathrm{II}})'$, $(\bar{C}_N^{\mathrm{II}})^{-1}$ may be very different from $(\bar{C}_N^{\mathrm{II}})'$. In other words, the matrix $\bar{C}_N^{\mathrm{II}}$ is not orthogonal in general. Therefore, we cannot use $(\bar{C}_N^{\mathrm{II}})'$ for reconstruction.

IV. Integer DCT-IV, DCT-I, and Fast Algorithms

All discrete cosine transform can be computed by the DCT-II and DCT-III [2], [9]. In this section, we use the relationships among DCTs to factor the DCT-I and DCT-IV into lifting steps and additions. Based on the factorization, we will propose new Integer DCT-IV and DCT-I. All types of DST have very simple relationships with the corresponding DCTs [2], [9]; therefore, the integral versions and the corresponding fast algorithms of them can also be achieved directly.

A. Integer DCT-IV and Fast Algorithm

Apart from the DCT-II and DCT-III, another most useful DCT is the DCT-IV [11], [19]. Let $x(n)$ $(n = 0, 1, \ldots, N-1)$ be a real input sequence. We assume that $N = 2^t$, where $t > 0$. The scaled DCT-IV of $x(n)$ is defined as follows:

$$X(k) = \sum_{n=0}^{N-1} x(n) \cos \frac{\pi(2n+1)(2k+1)}{4N} \qquad k = 0, 1, \ldots, N-1.$$

DCT-IV can be computed by using DCT-II or DCT-III as stated in [2] and [9]. We give a new simple relationship between DCT-II and DCT-IV in Lemma 3, which can be proved easily.

Lemma 5: Let C_N^{IV} be the transform matrix of the DCT-IV, that is, $C_N^{\mathrm{IV}} = (\cos(\pi(2k+1)(2n+1))/(4N))$. Then

$$C_N^{\mathrm{IV}} = U_N C_N^{\mathrm{II}} E_N F_N \tag{4.1}$$

where E_N, F_N, and U_N are matrices of order N and are defined in (3.5), (3.6), and (3.3), respectively.

Therefore, we can approximate C_N^{IV} by $\bar{C}_N^{\mathrm{IV}}$, which is defined as

$$\bar{C}_N^{\mathrm{IV}} = U_N \bar{C}_N^{\mathrm{II}} \bar{E}_N \bar{F}_N \tag{4.2}$$

where $\bar{E}_N$, $\bar{F}_N$, and $\bar{C}_N^{\mathrm{II}}$ are defined in (3.10), (3.9), and (3.13), respectively. This matrix defines a new type-IV integer discrete cosine transform (IntDCT-IV).

Definition 5: Assume that $N = 2^t$. The transform matrix of a type-IV integer discrete cosine transform (IntDCT-IV) with length N is defined by $\bar{C}_N^{\mathrm{IV}} = U_N \bar{C}_N^{\mathrm{II}} \bar{E}_N \bar{F}_N$.

The inverse of $\bar{C}_N^{\mathrm{IV}}$ is

$$\left(\bar{C}_N^{\mathrm{IV}}\right)^{-1} = \bar{F}_N^{-1} \bar{E}_N^{-1} \left(\bar{C}_N^{\mathrm{II}}\right)^{-1} U_N^{-1}. \tag{4.3}$$

It should be noted that $(\bar{C}_N^{\mathrm{IV}})^{-1}$ may be different from $\bar{C}_N^{\mathrm{IV}}$. We can not use $(2/N)\bar{C}_N^{\mathrm{IV}}$ for reconstruction. Based on the factorization, we get a float-point multiplication-free fast algorithm for the IntDCT-IV.

Algorithm 3-Fast Algorithm for IntDCT-IV:

Step 1) Compute $y = \bar{E}_N \bar{F}_N x$, where

$$\begin{aligned} y &= (y(0), y(1), \ldots, y(N-1))' \\ x &= (x(0), x(1), \ldots, x(N-1))'. \end{aligned}$$

Step 2) Use Algorithm 1 to compute the IntDCT-II of sequence $y(n)$, and let the outputs be $Y(k)$.

Step 3) Compute $X(0) = Y(0)/2$, $X(k) = Y(k) - X(k-1)$, $k = 1, 2, \ldots, N-1$.

The numbers of lifting steps and additions are, respectively, as follows.

$$\mathrm{LC}^{\mathrm{IV}}(N) = \frac{3}{2} N \log_2 N, \quad \mathrm{AC}^{\mathrm{IV}}(N) = 2N \log_2 N. \quad (4.4)$$

B. Integer DCT-I and Fast Algorithm

Based on the relationships between DCT-I and DCT-II proposed in [9], we will give a definition to integer DCT-I and propose its fast algorithm.

Let $x(n)$ $(n = 0, 1, \ldots, N)$ be a real number sequence, where N is assumed to be a power of 2. The scaled 1-D type-I discrete cosine transform (DCT-I) is defined as

$$X(k) = \sum_{n=0}^{N} x(n) \cos \frac{\pi n k}{N}, \quad k = 0, 1, \ldots, N.$$

Lemma 6: Let the transform matrix of the DCT-I be $C_N^{\mathrm{I}} = (\cos(\pi kn)/(N))_{k,n=0,1,\ldots,N}$, which is a matrix with order $N+1$. Then, we have

$$C_N^{\mathrm{I}} = \begin{bmatrix} I_{N/2} & 0 & I_{N/2} \\ 0 & 1 & 0 \\ \hat{\mathrm{I}}_{N/2} & 0 & -\hat{I}_{N/2} \end{bmatrix} \begin{bmatrix} C_{N/2}^{\mathrm{I}} & 0 \\ 0 & C_{N/2}^{\mathrm{II}} \end{bmatrix} Q_N \quad (4.5)$$

where Q_N is a permutation matrix with order $N+1$ defined as follows. For any vector $y = (y_0, y_1, \ldots, y_N)$

$$Q_N y = (y_0, y_2, \ldots, y_N, y_1, y_3, \ldots, y_{N-1}). \quad (4.6)$$

By replacing $C_{N/2}^{\mathrm{II}}$ by its integer version $\bar{C}_{N/2}^{\mathrm{II}}$, we define the integer DCT-I as follows.

Definition 6: Assume that $N = 2^t$. We define a type-I Integer discrete cosine transform (IntDCT-I) by giving its transform matrix recursively as

$$\bar{C}_1^{\mathrm{I}} = C_1^{\mathrm{I}} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$\bar{C}_{2^{j+1}}^{\mathrm{I}} = \begin{bmatrix} I_{2^j} & 0 & I_{2^j} \\ 0 & 1 & 0 \\ \hat{I}_{2^j} & 0 & -\hat{I}_{2^j} \end{bmatrix} \begin{bmatrix} \bar{C}_{2^j}^{\mathrm{I}} & 0 \\ 0 & \bar{C}_{2^j}^{\mathrm{II}} \end{bmatrix} Q_{2^{j+1}} \quad j = 0, 1, \ldots, t-1. \quad (4.7)$$

Based on the definition and the factorization, we get a fast algorithm for the IntDCT-I as follows.

Algorithm 4—Fast Algorithm for IntDCT-I:

Step 1) Compute the $((N/2)+1)$-point IntDCT-I of sequence $x(2n)$ $(n = 0, 1, \ldots, N/2)$ and the $N/2$-point IntDCT-II of sequence $x(2n+1)$ $(n = 0, 1, \ldots, N/2-1)$. Let the outputs be $V(k)$ and $W(k)$, respectively. If $N/2 > 1$, we should factor the $((N/2)+1)$-point IntDCT-I into smaller ones using (4.7).

Step 2) Compute

$$X(k) = V(k) + W(k), \quad k = 0, 1, \ldots, N/2-1$$
$$X(N/2) = V(N/2)$$
$$X(N-k) = V(k) - W(k), \quad k = 0, 1, \ldots, N/2-1.$$

The numbers of lifting steps and additions for the algorithm are, respectively, as follows.

$$\mathrm{LC}^{\mathrm{I}}(N) = \frac{3}{2} N \log_2 N - 6N + 3 \log_2 N + 6$$
$$\mathrm{AC}^{\mathrm{I}}(N) = 2N \log_2 N - 4N + 2 \log_2 N + 6. \quad (4.8)$$

The inverse of the matrix $\bar{C}_N^{\mathrm{I}}$ is defined recursively as follows:

$$\left(\bar{C}_N^{\mathrm{I}}\right)^{-1} = Q_N^{-1} \begin{bmatrix} \left(\bar{C}_{N/2}^{\mathrm{I}}\right)^{-1} & 0 \\ 0 & \left(\bar{C}_{N/2}^{\mathrm{II}}\right)^{-1} \end{bmatrix} \times \begin{bmatrix} \frac{1}{2} I_{N/2} & 0 & \frac{1}{2} \hat{I}_{N/2} \\ 0 & 1 & 0 \\ \frac{1}{2} I_{N/2} & 0 & -\frac{1}{2} \hat{I}_{N/2} \end{bmatrix}. \quad (4.9)$$

$x(n)$ can be reconstructed from $X(k)$ by using the inverse matrix or inverting the steps of the algorithm.

V. Two-Dimensional Integer DCT-II and Fast Algorithm

This section disscusses the 2-D integer DCTs. In general, a 2-D transform can be generated by simply using the tensor products of the corresponding 1-D transform, that is, we process the 2-D input array by implementing the 1-D transform along its rows and columns consecutively (also called row-column method). This method gives a 2-D transform that has a separable kernel. In this section, we will propose a nonseparable 2-D integer transform by combining the 1-D integer transform and the polynomial transform. The proposed 2-D transform needs far fewer numbers of operations than the row-column 2-D transform does.

Let $x(n, m)$ $(n = 0, 1, \ldots, N-1;\ m = 0, 1, \ldots, M-1)$ be the 2-D input sequence. We assume that M and N are powers of 2 and $M \geq N$. Therefore, we can write $N = 2^t$ and $M = 2^J N$, where $t > 0$ and $J \geq 0$, respectively. If $M < N$, the definition can also be used by simply interchanging N and M. The scaled 2D-DCT-II is defined as follows:

$$X(k,l) = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} x(n,m) \cos \frac{\pi(2n+1)k}{2N} \times \cos \frac{\pi(2m+1)l}{2M}$$
$$k = 0, 1, \ldots, N-1; \quad l = 0, 1, \ldots, M-1. \quad (5.1)$$

A. Two-Dimensional Integer DCT-II

Based on the 1-D IntDCT-II and the polynomial transform [14], we propose the following integer 2-D IntDCT-II.

Definition 7: The new integer 2-D-DCT (2-D-IntDCT) of $x(n,m)$ is $X(k,l)$ $(k=0,1,\ldots,N-1; l=0,1,\ldots,M-1)$, which can be computed in the following steps.

Step 1) Compute $y(p,m)=x(q(p,m),m)$, where we have the equation at the bottom of the page, and the function $f(p,m)=((4p+1)m+p) \bmod N$.

Step 2) Compute the 1-D IntDCT-II of each row of the array $y(p,m)$, and let the output array be $V(p,l)$.

Step 3) Compute a polynomial transform

$$A_k(z) \equiv \sum_{p=0}^{N-1} V_p(z)\hat{z}^{pk} \bmod z^{2M}+1; \quad k=0,1,\ldots,N-1 \tag{5.2}$$

where $V_p(z) = \sum_{l=0}^{M-1} V(p,l)z^l - \sum_{l=M+1}^{2M-1} V(p,2M-l)z^l$; $\hat{z} \equiv z^{2^{J+2}} \bmod z^{2M}+1$, and then get $B_k(z) \equiv A_k(z)z^{2^J k} \bmod z^{2M}+1$.

Step 4) Compute

$$X_k(z) = \sum_{l=0}^{2M-1} X(k,l)z^l \equiv \frac{1}{2}(B_k(z)+B_k(z^{-1})) \bmod z^{2M}+1 \tag{5.3}$$

where only $X(k,l)$ $(k=0,1,\ldots,N-1;\ l=0,1,\ldots,M-1)$ is needed.

In [24], we proved that if the IntDCT-II in Step 2 is replaced by the ordinary DCT-II, the above defined 2-D-IntDCT is the scaled version of the ordinary 2-D-DCT-II defined in (5.1). Therefore, the 2-D-IntDCT can also be viewed as the integer approximation to the float-point 2-D-DCT-II. However, it is different from the row-column 2-D integer DCT.

B. Computational Complexity of the Algorithm

If the input array is integer, the whole computation can be done by integer additions and shifting. Although Step 1 needs some integer multiplications, these operations are not related to the input and, hence, can be done in advance. The polynomial transform in Step 3 has been discussed in detail in [24], which requires $NM\log_2 N - N/2+1$ additions. Fig. 2 gives the flow chart of Step 3 when $N=M=8$. Therefore, the numbers of lifting steps and additions for the proposed 2-D-IntDCT are, respectively, $N \cdot \mathrm{LC}^{\mathrm{II}}(M)$ and $N \cdot \mathrm{AC}^{\mathrm{II}}(M) + NM\log_2 N + NM - M - 3N/2 + 2$, whereas those for the row-column 2-D integer DCT are, respectively, $N \cdot \mathrm{LC}^{\mathrm{II}}(M) + M \cdot \mathrm{LC}^{\mathrm{II}}(N)$ and $N \cdot \mathrm{AC}^{\mathrm{II}}(M) + M \cdot \mathrm{AC}^{\mathrm{II}}(N)$. Therefore, the proposed 2-D-IntDCT generally needs far fewer lifting steps and additions than that needed by the row-column 2-D integer DCT. The larger the size is, the more operations are reduced. Table I gives the comparisons when $N=M$.

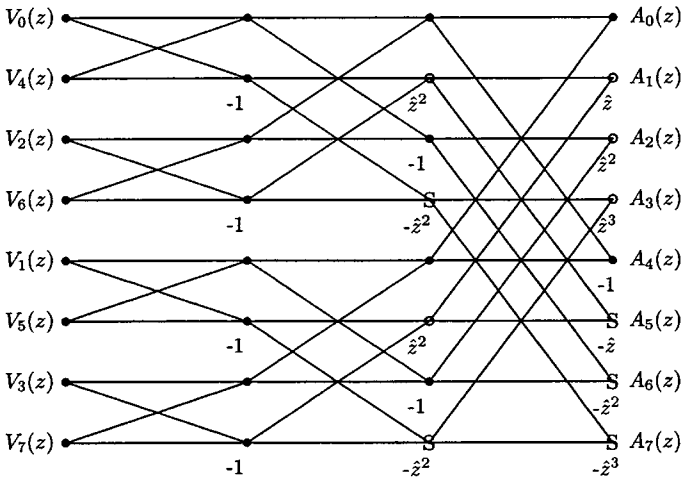

Fig. 2. Flowchart of Step 3.

TABLE I
COMPARISON OF THE NUMBER OF OPERATIONS

method	lifting steps	additions
proposed	$\frac{3}{2}N^2\log_2 N - 3N^2 + 3N$	$3N^2\log_2 N - N^2 - \frac{1}{2}N + 2$
row-column	$3N^2\log_2 N - 6N^2 + 6N$	$4N^2\log_2 N - 4N^2 + 4N$

C. Reconstruction Algorithm for 2-D IntDCT-II

The proposed transform is invertible if the 1-D IntDCT used in Step 1 is invertible. In addition, the inverting process can be described in the following.

Algorithm 5-Fast Algorithm for Inverse 2-D IntDCT-II:

Step 1) Generate the polynomials $X_k(z) = \sum_{l=0}^{M-1} X(k,l)z^l - \sum_{l=M+1}^{2M-1} X(k,2M-l)z^l$, and compute

$$A_k(z) \equiv (X_k(z) + X_{N-k}(z)z^{-M})z^{k2^J} \bmod (z^{2M}+1).$$

Step 2) Compute a polynomial transform

$$V_p(z) \equiv \frac{1}{N}\sum_{k=0}^{N-1} A_k(z)\hat{z}^{pk} \bmod z^{2M}+1 \qquad p=0,1,\ldots,N-1$$

where the coefficient of $V_p(z)$ is denoted by $V(p,l)$.

$$q(p,m) = \begin{cases} 2f\left(p,\frac{m}{2}\right), & m \text{ is even and } f\left(p,\frac{m}{2}\right) < \frac{N}{2} \\ 2N-1-2f\left(p,\frac{m}{2}\right), & m \text{ is even and } f\left(p,\frac{m}{2}\right) \geq \frac{N}{2} \\ 2f\left(p,M-\frac{m+1}{2}\right), & m \text{ is odd and } f\left(p,M-\frac{m+1}{2}\right) < \frac{N}{2} \\ 2N-1-2f\left(p,M-\frac{m+1}{2}\right), & m \text{ is odd and } f\left(p,M-\frac{m+1}{2}\right) \geq \frac{N}{2} \end{cases}$$
$$p=0,1,\ldots,N-1; \quad m=0,1,\ldots,M-1$$

Step 3) Compute the inverse 1-D IntDCT of each row of the array $V(p,l)$, and let the output array be $y(p,m)$.

Step 4) Reorder the array $y(p,m)$ to get $x(n,m) = y(r(n,m),m)$, where

$$r(n,m) = \begin{cases} g\left(\frac{n}{2},\frac{m}{2}\right), & m \text{ and } n \text{ are even} \\ g\left(N-\frac{n+1}{2},\frac{m}{2}\right), & m \text{ is even and } n \text{ is odd} \\ g\left(\frac{n}{2},M-\frac{m+1}{2}\right), & m \text{ is odd and } n \text{ is even} \\ g\left(N-\frac{n+1}{2},M-\frac{m+1}{2}\right), & m \text{ and } n \text{ are odd} \end{cases}$$

$$n = 0,1,\ldots,N-1; \quad m = 0,1,\ldots,M-1$$

and the function $g(n,m) = ((4m+1)^{-1}(n-m)) \bmod N$.

If the inverse 1-D IntDCT in Step 3 is replaced by the ordinary DCT-III, the above procedure computes the scaled 2-D-DCT-III. Therefore, the inverse 2-D IntDCT can also be viewed as an approximation to the ordinary 2-D-DCT-III.

VI. Comparison of DCT and IntDCT

IntDCT has at least two advantages over DCT. First, IntDCT needs no floating-point multiplications. The floating-point multiplications are replaced by lifting steps that need only integer operations and shifting. This is very important for applications in mobile devices since it is easier and cheaper (power saving) to realize integer operations than to implement float-point multiplications. Second, if sufficient word length is used to represent the intermediate data of IntDCT, the round-off error can be eliminated completely. There is no information lost after the transform even if it is computed in a fixed-point computer. Therefore, it can be used for lossless compression. Furthermore, using (2.4) and (2.5), we can also approximate a DCT by an integer to integer transform, which is invertible. Although DCT is also an invertible transform, round-off error is always exists when we approximate the trigonometric functions. It is impossible to perfectly construct the original data. Therefore, it can not be used for lossless image compression. However, since IntDCT is a new transform, its performance, such as the decorrelation property, remains to be questioned, and more experiments are needed to be done. Generally speaking, the performance of IntDCT is related to the number of bits used for the lifting multipliers. In our algorithms, it is equivalent to the accuracy of approximating α_j or $1/\alpha_j$ by $\mathrm{RB}(\alpha_j)$ or $\mathrm{RB}(1/\alpha_j)$ (see Definition 3 and Section III). It is very difficult to give a theoretic analysis on the accuracy and performance. Fortunately, experiments in [7] and [21] have shown that even with a very coarse approximation, the performance of eight-point and 16-point IntDCT is still close to that of DCT. We have not done experiments on the proposed IntDCT with sizes larger than 16 yet, but we do expect the feature remains.

VII. Conclusion

Methods are proposed to factor the DCTs into lifting steps and additions. By approximating the lifting matrices, new integer DCTs are obtained that are free of float-point multiplication. By combining the polynomial transform and the 1-D integer cosine transforms, a nonseparable 2-D integer discrete cosine transform is presented. The proposed 2-D integer transform uses far fewer operations than that used by the corresponding row-column 2-D integer transforms. Fast algorithms are given for the integer transforms and their computational complexities are analyzed. The proposed transforms can be used in mobile computing, lossless image coding, the multiplierless filterbank, and other related fields.

References

[1] G. P. Apousleman, M. W. Marcellin, and B. R. Hunt, "Compression of hyperspectral imagery using the 3-D DCT and hybrid DPCM/DCT," *IEEE Trans. Geosci. Remote Sens.*, vol. 33, pp. 26–34, Jan. 1995.

[2] V. Britanak, "A unified approach to the fast computation of discrete sinusoidal transform I: DCT and DST transform," *Comput. Artif. Intell.*, vol. 17, pp. 583–607, Dec. 1998.

[3] F. Bruekers and A. V. D. Enden, "New networks for perfect inversion and perfect reconstruction," *IEEE J. Select. Areas Commun.*, vol. 10, pp. 130–137, Jan. 1992.

[4] R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo, "Wavelet transforms that map integers to integers," *Appl. Comput. Harmon. Anal.*, vol. 5, no. 3, pp. 332–369, 1998.

[5] W. K. Cham and P. C. Yip, "Integer sinusoidal transforms for image processing," *Int. J. Electron.*, vol. 70, no. 6, pp. 1015–1030, 1991.

[6] S. C. Chan, W. Liu, and K. L. Ho, "Perfect reconstruction modulated filter banks with sum of powers-of-two coefficients," *Proc. IEEE Int. Symp. Circuits Syst.*, vol. II, pp. 73–76, May 28–31, 2000.

[7] Y.-J. Chen, "Integer discrete cosine transform (IntDCT)," presented at the Second Int. Conf. Inform. Commun. Signal Process., Singapore, Dec. 1999. Invited paper.

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

[9] Z. R. Jiang, Y. Zeng, and P. N. Yu, *Fast Algorithms*, Changsha, China: National Univ. Defense Technol. Press, 1994. In Chinese.

[10] B. G. Lee, "A new algorithm to compute the discrete cosine transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP–32, pp. 1243–1245, May 1984.

[11] H. S. Malvar, *Signal Processing with Lapped Transform*. Norwood, MA: Artech House, 1991.

[12] M. W. Marcellin, M. J. Gormish, A. Bilgin, and M. P. Boliek, "An overview of JPEG-2000," in *Proc. Data Compression Conf.*, 2000, pp. 523–541.

[13] N. Memon, X. Wu, and B. L. Yeo, "Improved techniques for lossless image compression with reversible integer wavelet transforms," in *Proc. IEEE Int. Conf. Image Process.*, vol. 3, 1998, pp. 891–895.

[14] H. J. Nussbaumer, "New polynomial transform algorithms for multidimensional DFT's and convolutions," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP–29, pp. 74–84, Feb. 1981.

[15] S. C. Pei and J. J. Ding, "Integer discrete Fourier transform and its extension to integer trigomatric transforms," *Proc. IEEE Int. Symp. Circuits Syst.*, vol. V, pp. 513–516, May 28–31, 2000.

[16] W. Philps, "Lossless DCT for combined lossy/lossless image coding," in *Proc. IEEE Int. Conf. Image Process.*, vol. 3, 1998, pp. 871–875.

[17] K. R. Rao and P. Yip, *Discrete Cosine Transform: Algorithms, Advantages and Applications*. New York: Academic, 1990.

[18] 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.

[19] G. Strang and T. Nguyen, *Wavelets and Filter Banks*. Wellesley, MA: Wellesley-Cambridge Press, 1997.

[20] W. Sweldens, "The lifting scheme: A construction of second generation wavelets," *SIAM J. Math. Anal.*, vol. 29, no. 2, pp. 511–546, 1998.

[21] T. D. Tran. Fast Multiplierless Approximation of the DCT. [Online]. Available: http://thanglong.ece.jhu.edu/Tran/Pub/intDCT.ps.gz

[22] S. Vankataraman, V. R. Kanchan, K. R. Rao, and M. Mohanty, "Discrete transforms via the Walsh-Hadamard transform," *Signal Process.*, vol. 14, no. 4, pp. 371–382, June 1988.

[23] Y. Zeng, "Vector parallel algorithms for DCT and DST," presented at the Proc. First Nat. Conf. Parallel Algorithms, Beijing, China, 1988. in Chinese.

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

Yonghong Zeng received the B.S. degree in mathematics from Beijing University, Beijing, China, in 1983 and the M.S. degree in applied mathematics and the Ph.D. degree in computer science and technology from the National University of Defense Technology (NUDT), Changsha, China, in 1986 and 1998 respectively.

Since 1986, he has been with the Department of System Engineering and Mathematics, NUDT. He has served as an associate professor since 1993. He joined the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, to work as a research fellow in 1999. His main research interests include algorithms and transforms for signal processing, parallel computing, radar and sonar, and filterbanks and wavelets. He has published four books and a number of papers.

Dr. Zeng received ministry-level Scientific and Technological Development Awards in China three times.

Lizhi Cheng received the B.S. degree in mathematics from the Hunan Normal University, Hunan, China, in 1983 and the M.S. degree in applied mathematics from the National University of Defense Technology (NUDT), Chengshe, China, in 1988.

Since 1988, he has been with the Department of Mathematics and System Science, School of Science, NUDT. He has served as an associate professor since 1994. His main research interests include algorithms and transforms for signal processing, filterbanks and wavelets, image processing. and matrix computation. He has published two books and a number of papers.

Mr. Cheng twice received ministry-level Scientific and Technological Development Awards.

Guoan Bi received the B.Eng. degree in electrical engineering from the Dalian University of Technology, Dalian, China, in 1982 and the M.Sc. degree in telecommunication systems and the Ph.D. degree in electronic systems from Essex University, Essex, U.K., in 1985 and 1988, respectively.

In 1988, he joined the Department of Electrical and Electronic Engineering, University of Surrey, Surrey, U.K. Since 1991, he has been with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. His research interests include signal processing algorithms, ASIC hardware structures, and signal processing for mobile communications.

Alex C. Kot (SM'98) graduated from the University of Rochester, Rochester, NY, and received the Ph.D. degree in electrical engineering from the University of Rhode Island, Kingston, in 1989.

He was with the AT&T Bell Company, New York. Since 1991, he has been with the Nanyang Technological University (NTU), Singapore, where he is currently Head of the Information Engineering Division. His research and teaching interests are in the areas of signal processing for communications, digital signal processing, and information technology.

Dr. Kot served as the General Co-Chair for the Second International Conference on Information, Communications, and Signal Processing (ICICS) in December 1999. He also received the NTU Best Teacher of the Year Award and has served as the Chairman of the IEEE Signal Processing Chapter in Singapore. He is currently an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING and the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY.