An Efficient Algorithm for Computing Two-Dimensional Discrete Cosine Transforms

Nhi Ta, Yianni Attikiouzel and Greg Crebbin

Department of Electrical and Electronic Engineering
The University of Western Australia
Nedlands, Western Australia 6009
AUSTRALIA

Abstract

A fast algorithm for computing the two-dimensional discrete cosine transform (2-D DCT) is proposed. In this algorithm the 2-D DCT is converted into a form of 2-D DFT which is called the odd DFT(ODFT). The odd DFT can be calculated by a DFT followed by post-multiplications. The DFT part of odd DFT is calculated by the fast discrete Radon transform. The complexity of the proposed algorithm is comparable to the polynomial transform approach. This new algorithm produces a regular structure which makes it attractive for VLSI implementation. Furthermore, the computation can be performed in parallel.

1 Introduction

Since its introduction in 1974, the discrete cosine transform [1] has found many applications in image processing, and data compression; due to its ability to closely approximate the optimal Karhunen-Loeve transform (KLT) and its suitability for implementation by a fast algorithm. Recently, it has been adopted by the CCITT as part of the video coding standard.

In image coding, a 2-D DCT is used. The usual way of computing the 2-D DCT has been the row-column approach, where a 2-D DCT of an (NxN) block is decomposed into N DCTs for the N rows and N DCTs for the N columns. Recent studies [7] [4] have shown that direct 2-D techniques are more efficient than row-column approaches.

Most of the direct 2-D methods for computing 2-D DCT are based on polynomial transform [4] [7]. In [4] , the polynomial transform is used to calculated the 2-D FFT and a rotation stage is used to perform the complex multiplications. In a recent paper [7], the polynomial transform is performed on the W_{4N}^{ij+kl} term. The latter reduces the computational complexity at the expense of a more complex algorithm and flowgraph structure.

In this paper, a 2-D direct computation of the 2-D DCT is proposed. The method is based on the geometric relationship between points on a 2-D grid. The 2-D DCT is first formulated as a two-dimensional odd DFT (2-D ODFT). The newly formed 2-D ODFT can be calculated by a 2-D DFT and post-multiplications.

A brief description of the Gertner method [5] for computing the 2-D FFT is presented in section 2, together with the radix-2 fast discrete Radon transfrom (FDRT). The mapping used in converting the DCT into an odd DFT, and the 2-D DCT algorithm by using the FDRT are described in section 3. A discussion on the computational complexity of the algorithm is presented in section 4.

2 Computing of the 2-D DFT

In this section, we briefly describe the discrete Radon transform (DRT) for computing a 2-D DFT proposed by Gertner [5], in order to understand the computation of the 2-D DCT. A 2-D DFT is defined as

$$X_{k,l} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} x_{i,j} W_N^{ik+jl} \tag{1}$$

where $W_N = e^{-j2\pi/N}$, and $k, l = 0, \cdots, N-1$.

Gertner [5] has shown that an (NxN) point 2-D DFT can be computed by taking a number of N-point 1-D DFTs, where this number is equal to the number of linear congruences of the (NxN) grid. For $N = 2^n$, the number of linear congruences on the (NxN) grid is $3N/2$.

The 2-D DFT can now be calculated using the following equations:

$$X(< N - ml >_N, l) = \sum_{d=0}^{N-1} R_1(m,d) W_N^{ld} \tag{2a}$$

$$m = 0, 1, \cdots, N-1 \qquad l = 0, 1, \cdots, N-1$$

$$X(k, < N - 2sk >_N) = \sum_{d=0}^{N-1} R_2(s,d) W_N^{kd} \tag{2b}$$

$$s = 0, 1, \cdots, N/2-1 \qquad k = 0, 1, \cdots, N-1$$

where R_1 and R_2 are the DRTs on the input array $x(i,j)$, which are defined as:

$$R_1(m,d) = \sum_{i=0}^{N-1} x(i, < d + mi >_N) \tag{3a}$$

$$d, m = 0, \cdots, N-1$$

CH 3006-4/91/0000 – 0396 $1.00 © IEEE

$$R_2(s,d) = \sum_{i=0}^{N-1} x(< d+2si >_N, i) \quad (3b)$$

$$d = 0, \cdots, N-1 \qquad s = 0, \cdots, N/2-1$$

Normally, the DRT requires $\frac{3}{2}(N-1)N^2$ additions. A radix-2 algorithm is used to reduce the number of additions for computing the DRT [6]. Let

$$R_1^{(t)}(i,m,d) = \sum_{j=0}^{2^t-1} x(j2^{n-t}+i, < d+m(j2^{n-t}+i) >_N) \quad (4a)$$

$$R_2^{(t)}(i,s,d) = \sum_{j=0}^{2^t-1} x(< d+2s(j2^{n-t}+i) >_N, j2^{n-t}+i) \quad (4b)$$

for $i = 0, \ldots, 2^{n-t}-1$. The DRT can be computed in n stages. At each stage t,

$$R_1^t(i,m,d) = R_1^{(t-1)}(i,m,d) + R_1^{(t-1)}(i+2^{n-t},m,d) \quad (5a)$$

$$R_2^t(i,s,d) = R_2^{(t-1)}(i,s,d) + R_2^{(t-1)}(i+2^{n-t},s,d) \quad (5b)$$

where

$$R_1^{(0)}(i,m,d) = x(i, < d+mi >_N) \quad (6a)$$

$$R_2^{(0)}(i,s,d) = x(< d+2si >_N, i) \quad (6b)$$

and

$$R_1(m,d) = R_1^{(n)}(0,m,d) \quad (7a)$$

$$R_2(s,d) = R_2^{(n)}(0,s,d) \quad (7a)$$

By using the following properties:

$$R_1^{(t)}(i,m+q2^t,d) = R_1^{(t)}(i,m, < d+iq2^t >_N) \quad (8a)$$

$$R_2^{(t)}(i,s+q2^{t-1},d) = R_2^{(t)}(i,s, < d+iq2^t >_N) \quad (8b)$$

the ranges of m and s in each stage t, reduces to $m = 0, \ldots, 2^t-1$ and $s = 0, \ldots, 2^{t-1}-1$. Hence, the number of additions required to compute the DRT for an (NxN) array, where $N = 2^n$, is $\frac{3}{2}\log(N)N^2$. The flowgraph of FDRT for an (8×8) array is shown in Figures 2 and 3, for R_1 and R_2 respectively. Note that the second stage for R_2 is the same as the two blocks in the second stage of R_1. As we can see, the structure of the flowgraph is very regular. This feature makes the FDRT approach for computing 2-D FFT easily adaptable for VLSI implementation.

3 Computing 2-D DCT using FDRT

In this section, we describe the mapping that enables the computation of a DCT via a DFT. A 1-D DCT is defined as follows:

$$X(k) = \sum_{i=0}^{N-1} x(i) cos\frac{2\pi(2i+1)k}{4N} \quad (9)$$

$$k = 0, 1, \cdots, N-1$$

By using the classical mapping [2], [3]

$$y(n) = \begin{cases} x(2n) & n = 0, 1, \cdots, \frac{N}{2}-1 \\ x(2N-2n-1) & n = \frac{N}{2}, \cdots, N-1 \end{cases} \quad (10)$$

the DCT becomes

$$X(k) = \sum_{i=0}^{N-1} y(i) cos\frac{2\pi(4i+1)k}{4N} \quad (11)$$

$$k = 0, 1, \cdots, N-1$$

Using a trigonometric identity, the expression for $X(N-k)$ is similar to the above equation except that the cosine function is replaced by a sine function. Hence we can define a transform $U(k)$ as follows:

$$U(k) = X(k) + jX(N-k) = \sum_{i=0}^{N-1} y(i) W_{4N}^{(4i+1)k} \quad (12)$$

Note that we only need to evaluate k such that the set $\{k, N-k\} = \{0, \ldots, N-1\}$.

For the 2-D case, the 2-D DCT is defined as:

$$X(k,l) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} y(i,j) cos\frac{2\pi(2i+1)k}{4N} cos\frac{2\pi(2j+1)l}{4N} \quad (13)$$

$$k, l = 0, \ldots, N-1$$

By analogy with the 1-D case, it is easy to recognize that the 2-D DCT coefficients can be obtained from $U(k,l)$ by the following set of equations :

$$\begin{aligned} X(k,l) &= Re[U(k,l) + jU(N-k,l)] \\ X(k,N-l) &= -Im[U(k,l) + jU(N-k,l)] \\ X(N-k,l) &= -Im[U(k,l) - jU(N-k,l)] \\ X(N-k,N-l) &= -Re[U(k,l) - jU(N-k,l)] \end{aligned} \quad (14)$$

where

$$U(k,l) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} y(i,j) W_{4N}^{(4i+1)k+(4j+1)l} \quad (15)$$

and $y(i,j)$ is the 2-D extension of the mapping described in equation (10). [4]

$$y(i,j) = \begin{cases} x(2i,2j) & i = 0, \ldots, N/2-1 \\ & j = 0, \ldots, N/2-1 \\ x(2N-2i-1,2j) & i = N/2, \ldots, N-1 \\ & j = 0, \ldots, N/2-1 \\ x(2i,2N-2j-1) & i = 0, \ldots, N/2-1 \\ & j = N/2, \ldots, N-1 \\ x(2N-2i-1,2N-2j-1) & i = N/2, \ldots, N-1 \\ & j = N/2, \ldots, N-1 \end{cases} \quad (16)$$

Note that eqn (14) requires $U(k,l)$ to be computed for all k, and only a subset of l such that $\{l, N-l\}$ cover all possible values of l.

We can rewrite $U(k,l)$ as follows:

$$U(k,l) = W_{4N}^{k+l} \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} y(i,j) W_N^{ik+jl} \quad (17)$$

We observe that the summation in the above equation is identical to that for the 2-D DFT. This can be calculated using the FDRT approach, the post-multiplication stage can be implemented as suggested in [4].

$$\begin{bmatrix} X(k,l) \\ X(N-k,l) \\ X(k,N-l) \\ X(N-k,N-l) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & -1 & 0 \\ -1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} C^{k+l} & -S^{k+l} & 0 & 0 \\ C^{k+l} & S^{k+l} & 0 & 0 \\ 0 & 0 & C^{k-l} & -S^{k-l} \\ 0 & 0 & C^{k-l} & S^{k-l} \end{bmatrix} \begin{bmatrix} Re[(k,l)] \\ Im[(k,l)] \\ Re[(k,N-l)] \\ Im[(k,N-l)] \end{bmatrix} \quad (18)$$

where $C^i = cos(2\pi(i)/4N)$ and $S^i = sin(2\pi(i)/4N)$.

4 Arithmetic Complexity

Let the $M[\cdot]$ and $A[\cdot]$ denote the number of multiplications and additions needed to compute $[\cdot]$ respectively. Since the algorithm can be divided into two stages, the DFT and post-multiplications. The number of multiplications and additions required for the FDRT-based 2-D DFT for real $(N \times N)$ array are respectively:

$$M[\mathrm{DFT(N \times N)}] = \log(\mathrm{N})\mathrm{N}^2/2 - \frac{7}{6}\mathrm{N}^2 + 8/3$$

$$A[\mathrm{DFT(N \times N)}] = 3\log(\mathrm{N})\mathrm{N}^2 - 2\mathrm{N}^2 + 18$$

The number of multiplication and addition for the post multiplication stage are $(3N^2-2N)$ and $(5N^2-6N+2)$ respectively. Hence, the number of multiplications and additions required for 2-D DCT of a $(N \times N)$ array are:

$$M[\mathrm{DCT(N \times N)}] = \log(\mathrm{N})\mathrm{N}^2/2 + \frac{11}{6}\mathrm{N}^2 - 2\mathrm{N} + 8/3$$

$$A[\mathrm{DCT(N \times N)}] = 3\log(\mathrm{N})\mathrm{N}^2 + 3\mathrm{N}^2 - 6\mathrm{N} + 20$$

The measure of computation complexity of the proposed algorithm is shown in Table 1 together with that of other algorithms for comparison. The complexity of the proposed approach is comparable with the current polynomial transform methods. The number of multiplications is the same as for the Vetterli approach, and greater than that for the Duhamel-Guillemot. The number of additions are generally slightly more than that of the polynomial transform methods. A reduction in the number of additions can be achieved by examining the redundancies in the computing of the FDRT, for example, $R_2(0,0) = R_1(0,0)$.

5 Conclusion

In this paper we proposed a new method for computing the 2-D DCT using a number of 1-D DFTs which is less than 2N. The measure of arithmetic complexity is comparable with the other approaches in the current literature. Research in this area should be able to reduce the complexity even further. The flowgraph of the proposed algorithm has regular structure which can be easily implemented using VLSI. Finally, the algorithm is well suited for parallel implementation.

References

[1] N. Ahmed, T. Natarajan, and K.R. Rao, "Discrete Cosine Transform," *IEEE Trans. Comp.*, vol C-23, pp 90-93, 1974.

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

[3] M. Vetterli and H.J. Nussbaumer, "Simple FFT and DCT algorithms with reduced number of operations," *Signal Processing*, vol 6, pp 267-278, 1984.

[4] M. Vetterli, "Fast 2-D Discrete Cosine Transform", *Proc. IEEE ICASSP'85* pp 1538-1541, 1985.

[5] I. Gertner, "A new efficient algorithm to compute the two-dimensional discrete Fourier transform," *IEEE Trans. Acous. Speech Sig. Proc.*, vol ASSP-36, pp 1036-1050, 1988.

[6] D. Yang, "Fast Discrete Radon Transform and 2-D Discrete Fourier Transform," *Elec. Lett.*, vol 26, no 8, pp 550-551, 1990.

[7] P. Duhamel and C. Guillemot "Polynomial Transform Computation of the 2-D DCT," *Proc. IEEE ICASSP'90*, pp 1515-1518, 1990.

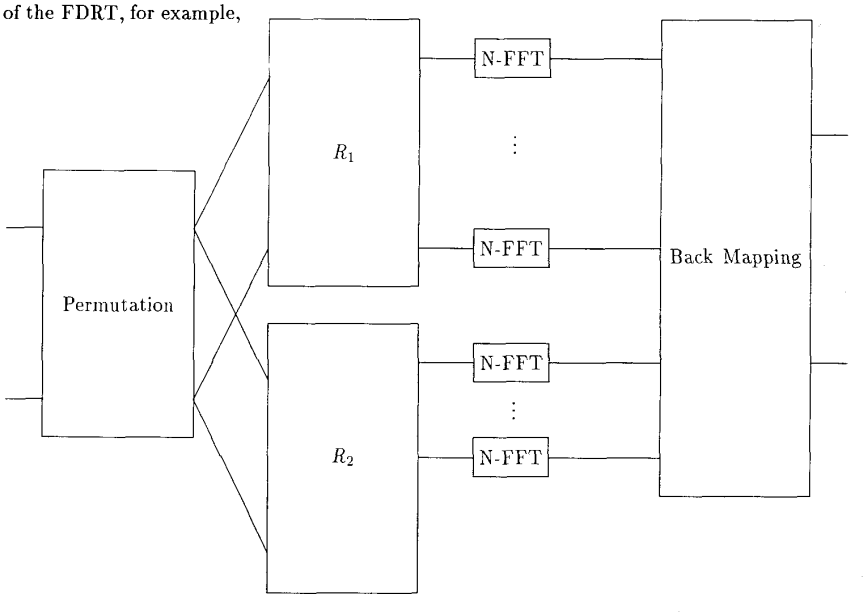

Figure 1: The Decomposition of 2-D DCT using FDRT

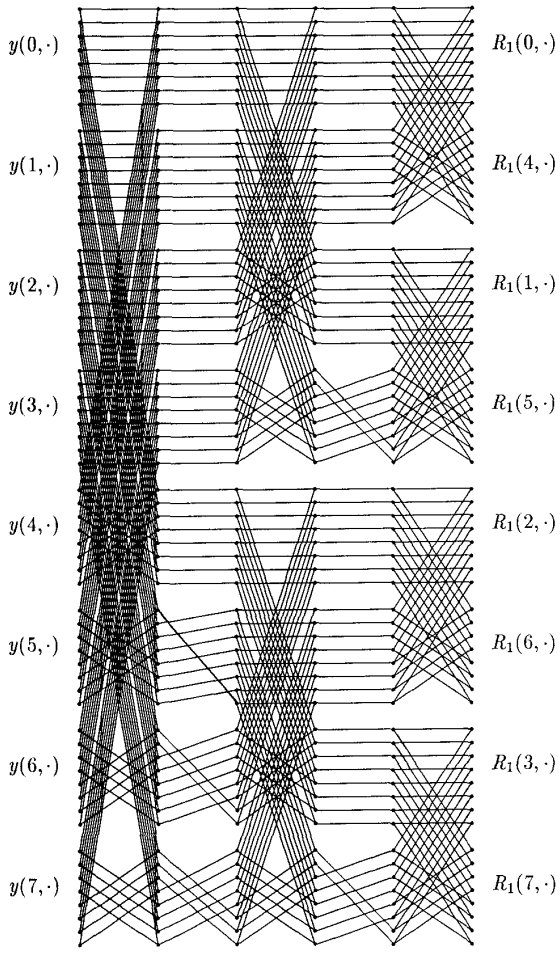

Figure 2: The flowchart for computing R_1 of a 8×8 fast discrete Radon transform

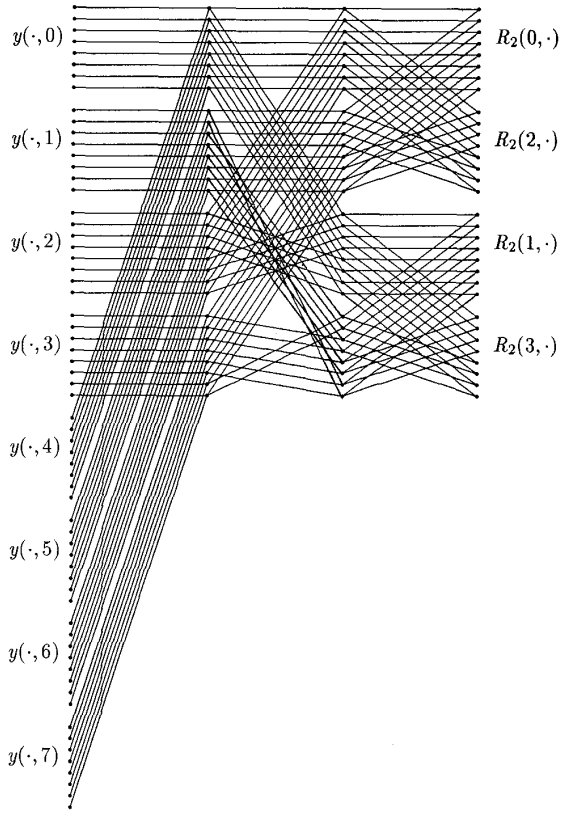

Figure 3: The Flowchart computing R_2 of a 8×8 FDRT

Table 1: Arithmetic complexity of NxN 2-D DCT algorithms.

N	Polynomial Transform [7]		Polynomial Transform [4]		Proposed Algorithm		row-col FFCT [3]	
	Mult.	Add.	Mult.	Add.	Mult.	Add.	Mult.	Add.
8	96	484	104	462	104	580	192	464
16	512	2531	568	2558	568	3124	1024	2592
32	2560	12578	2840	12950	2840	15700	5120	13376
64	12288	60578	13528	62442	13528	75412	24576	65664