FAST 2-D DISCRETE COSINE TRANSFORM

MARTIN VETTERLI

ECOLE POLYTECHNIQUE FEDERALE DE LAUSANNE
16, Chemin de Bellerive
CH-1007 LAUSANNE, SWITZERLAND

Abstract

A fast radix-2 two dimensional discrete cosine transform (DCT) is presented. First, the mapping into a 2-D discrete Fourier transform (DFT) of a real signal is improved. Then an usual polynomial transform approach is used in order to map the 2-D DFT into a reduced size 2-D DFT and one dimensional odd DFT's. Finally, optimized odd DFT algorithms for real signals are developped. Alltogether, a reduction of more than 50% in the number of multiplications and a comparable amount of additions is obtained in comparison to other algorithms.

I Introduction

Block coding using the two dimensional discrete cosine transform is widely used in image data compression [1]. Usually, a small block (typically 8 by 8 or 16 by 16) is transformed and an appropriate bit allocation is made in the transform domain.
One approach to the 2-D DCT computation uses the separability property and computes a DCT of dimension N by N as 2N DCT's of N which can be computed by one of the known fast algorithms [2,3]. The other approach consists in computing directly the 2-D transform by means of polynomial transforms and was first proposed in [4,5], restated in [6] and applied in [7]. Our method is also a direct approach to the 2-D problem. The DCT of N by N is mapped into a real DFT of N by N followed by post-multiplications. These post-multiplications are shown to be rotations, thus reducing the required number of operations. The real DFT is evaluated through polynomial transforms where the fact that the data is real is taken into account, specifically by developping an optimized real odd DFT algorithm.

II Evaluation of the DCT from a DFT

First we consider the mapping from DCT to DFT and the optimization of the resulting post-multiplications. Assume a real signal $x(n_1,n_2)$ of dimension NxN. The 2-D DCT is defined as:

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

In the following, we assume N to be a power of 2. Using the mapping introduced in [3] and then in [9] given by:

$$\begin{aligned} y(n_1,n_2) &= x(2n_1,2n_2) && n_1=0..N/2\text{-}1,\ n_2=0..N/2\text{-}1 \\ &= x(2N-2n_1-1,2n_2) && n_1=N/2..N\text{-}1,\ n_2=0..N/2\text{-}1 \\ &= x(2n_1,2N-2n_2-1) && n_1=0..N/2\text{-}1,\ n_2=N/2..N\text{-}1 \\ &= x(2N-2n_1-1,2N-2n_2-1) && n_1=N/2..N\text{-}1,\ n_2=N/2..N\text{-}1 \end{aligned} \quad (2)$$

The DCT becomes:

$$DCT(k_1,k_2,N)=\sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N-1} y(n_1,n_2)\cdot\cos\left(\frac{2\pi(4n_1+1)k_1}{4N}\right)\cdot\cos\left(\frac{2\pi(4n_2+1)k_2}{4N}\right) \quad (3)$$

Using basic trigonometry, this can be rewritten as phasors times real and imaginary part of a 2-D DFT. We use the following shorthands:

$$\text{cos-DFT}(k_1,k_2,N) = \sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N-1} y(n_1,n_2)\cdot\cos\left(\frac{2\pi(n_1k_1+n_2k_2)}{N}\right)$$

$$\text{sin-DFT}(k_1,k_2,N) = \sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N-1} y(n_1,n_2)\cdot\sin\left(\frac{2\pi(n_1k_1+n_2k_2)}{N}\right) \quad (4)$$

which correspond to the real and imaginary part of a 2-D DFT. Equation (3) becomes:

$$\begin{aligned} DCT(k_1,k_2,N) = 1/2\cdot[&\cos\left(\frac{2\pi(k_1+k_2)}{4N}\right)\cdot\text{cos-DFT}(k_1,k_2,N) \\ &- \sin\left(\frac{2\pi(k_1+k_2)}{4N}\right)\cdot\text{sin-DFT}(k_1,k_2,N) \\ &+ \cos\left(\frac{2\pi(k_1-k_2)}{4N}\right)\cdot\text{cos-DFT}(k_1,N-k_2,N) \\ &- \sin\left(\frac{2\pi(k_1-k_2)}{4N}\right)\cdot\text{sin-DFT}(k_1,N-k_2,N)\ ] \end{aligned} \quad (5)$$

Now, we note that $DCT(k_1,k_2,N)$, $DCT(N-k_1,k_2,N)$, $DCT(k_1,N-k_2,N)$ and $DCT(N-k_1,N-k_2,N)$ can be derived from the corresponding cos- and sin-DFT's by 2 plane rotations and some additions as follows:

$$\begin{bmatrix} DCT(k_1,k_2,N) \\ DCT(N-k_1,k_2,N) \\ DCT(k_1,N-k_2,N) \\ DCT(N-k_1,N-k_2,N) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} R(k_1,k_2) & O_2 \\ O_2 & R(k_1,-k_2) \end{bmatrix} \cdot \begin{bmatrix} \text{cos-DFT}(k_1,k_2,N) \\ \text{sin-DFT}(k_1,k_2,N) \\ \text{cos-DFT}(k_1,N-k_2,N) \\ \text{sin-DFT}(k_1,N-k_2,N) \end{bmatrix} \quad (6)$$

where:

$$O_2 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad R(k_1,k_2) = \begin{bmatrix} \cos\left(\frac{2\pi(k_1+k_2)}{4N}\right) & -\sin\left(\frac{2\pi(k_1+k_2)}{4N}\right) \\ \sin\left(\frac{2\pi(k_1+k_2)}{4N}\right) & \cos\left(\frac{2\pi(k_1+k_2)}{4N}\right) \end{bmatrix} \quad (7)$$

40.8.1

CH2118-8/85/0000-1538 $1.00 © 1985 IEEE

Since $R(k_1,k_2)$ is a rotation matrix whose product with a 2-point vector requires 3 mults and 3 adds [9], the evaluation of (6) requires a total 6 mults and 10 adds. Since (6) has to be evaluated for k_1 and k_2 going from 0 to N/2-1, and taking into account all simplifications (eg. $k_1 = k_2$), the load for obtaining the DCT from the real DFT is:

$$3N^2/2 - 2N \text{ mu.} \qquad 5N^2/2 - 6N + 2 \text{ ad.} \tag{8}$$

While the development above is quite cumbersome, it should be noted that, especially for small transforms, the post-multiplications dominate the computational load, particularly for multiplies (Ex. 8x8 DCT: 24 multiplications for the real DFT and 80 for the post-multiplications).

III 2-D Real DFT Calculation using Polynomial Transforms

The real DFT will be computed using polynomial transforms in a way described in [5,10] to which we refer for details. Below, the main steps are simply recalled. We want to evaluate:

$$X(k_1,k_2) = \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} x(n_1,n_2)\; W^{n_1k_1}\; W^{n_2k_2} \qquad W = e^{-j2\pi/N} \tag{9}$$

Where $N = 2^m$. Since $x(n_1,n_2)$ is real, $X(k_1,k_2)$ is equal to $[X(N-k_1,N-k_2)]^*$, $*$ denoting complex conjugates.

For k_2 odd, equation (9) can be written as [5]:

$$X(k_1,k_2) \triangleq X^1_{k_1}(z) \text{ Mod}(z - W^{k_2}) \qquad k_2 \text{ odd} \tag{10}$$

$$\text{where } X^1_{k_1}(z) \triangleq \sum_{n_1=0}^{N-1} X^1_{n_1}(z)\, W^{n_1k_1} \text{ Mod}(z^{N/2} + 1) \tag{11}$$

$$\text{and } X^1_{n_1}(z) = \sum_{n_2=0}^{N/2-1} [x(n_1,n_2) - x(n_1,n_2+N/2)]\, z^{n_2} \tag{12}$$

since $(z - W^{k_2})$, k_2 being odd, is a factor of $(z^{N/2} + 1)$, which is in turn a factor of $(z^N - 1)$. For $x(n_1,n_2)$ real, (12) requires $N^2/2$ real additions. Now, k_1 in (11) is replaced by k_1k_2 (which is a valid permutation since k_2 and N are relatively prime). Then, we can replace W^{k_2} by z in (11) since they are equivalent by (10). This leads to:

$$X^1_{k_1k_2}(z) \triangleq \sum_{n_1=0}^{N-1} X^1_{n_1}(z)\, z^{n_1k_1} \quad \text{Mod}(z^{N/2} + 1) \tag{13}$$

This is a polynomial transform of length N and a root z defined modulo $(z^{N/2} + 1)$ which can be computed with an FFT radix-2 type algorithm and uses $N^2/2\log_2 N$ real additions for $X^1_{n_1}$ with real coefficients. Since $X^1_{k_1k_2}(z)$ is of degree N/2-1 and k_2 odd equal to (2u+1), (10) can be rewritten as:

$$X(k_1(2u+1),(2u+1)) = \sum_{l=0}^{N/2-1} y(k_1,l)\; W^l\; W^{2ul} \tag{14}$$

where $y(k_1,l)$ is the l-th coefficient of the polynomial $X^1_{k_1k_2}(z)$. Equation (14) represents N odd DFT's of length N/2 whose computation will be adressed in the next section.

When both k_1 and k_2 are even, equation (9) reduces to a real DFT of size N/2 by N/2 on the real signal given by:

$$x(n_1,n_2) + x(n_1,n_2+N/2) + x(n_1+N/2,n_2) + x(n_1+N/2,n_2+N/2) \tag{15}$$

When k_1 is odd and k_2 even, (9) can be rewritten, similarly to (10-12), as

$$X(k_1,2uk_1) \triangleq X^1_{2uk_1}(z) \text{ Mod}(z-W^{k_1}) \qquad k_1 \text{ odd} \tag{16}$$

$$\text{where } X^1_{2uk_1}(z) \triangleq \sum_{n_2=0}^{N-1} X^1_{n_2}(z)\, z^{2un_2} \text{ Mod}(z^{N/2} + 1) \quad u=0..N/2-1 \tag{17}$$

$$\text{and } X^1_{n_1}(z) = \sum_{n_1=0}^{N/2-1} [x(n_1,n_2) + x(n_1,n_2+N/2) - x(n_1+N/2,n_2) - x(n_1+N/2,n_2+N/2)]\, z^{n_1} \tag{18}$$

We note that (17) is a length N/2 polynomial transform with a root z^2 defined modulo $(z^{N/2} + 1)$ which can be computed similarly to (13) using $N^2/2(\log_2 N-1)$ real additions when $X(k_1,k_2)$ is real. Note that (15) and (18) require together N^2 additions. At last, (16) can be rewritten as (14), and thus evaluated as N/2 odd DFT's of length N/2.

All together, the real DFT of N by N has been mapped into 3N/2 odd DFT's of length N/2 on real data and a real DFT of N/2 by N/2, at the cost of $3N^2/4 \log_2 N + 5N^2/4$ real additions.

IV Computation of the real odd DFT's

The length-N odd DFT's required for the evaluation of 2-D DFT's have the general form:

$$X_k = \sum_{n=0}^{N-1} x(n) \quad W_{2N}^{(2k+1)n} \qquad W_{2N} = e^{-j2\pi/2N} \tag{19}$$

We introduce the following shorthands:

$$\text{ocos-DFT}(k,n) = \sum_{n=0}^{N-1} x(n) \cos\left(\frac{2\pi(2k+1)n}{2N}\right) \qquad k=0..N-1 \tag{20}$$

$$\text{osin-DFT}(k,n) = \sum_{n=0}^{N-1} x(n) \sin\left(\frac{2\pi(2k+1)n}{2N}\right) \qquad k=0..N-1 \tag{21}$$

$$\text{o-DCT}(k,n) = \sum_{n=0}^{N-1} x(n) \cos\left(\frac{2\pi(2k+1)(2n+1)}{8N}\right) \qquad k=0..N-1 \tag{22}$$

and thus (19) is equal to:

$$X_k = \text{ocos-DFT}(k,N) - j \text{ osin-DFT}(k,N) \tag{23}$$

Now we use a similar approach as in [9] to evaluate (23). The real part is reduced to:

$$\text{ocos-DFT}(k,n) = \sum_{n=0}^{N/2-1} x(2n) \cos\left(\frac{2\pi(2k+1)n}{N}\right) + \sum_{n=0}^{N/4-1} (x(2n+1)-x(N-2n-1)) \cos\left(\frac{2\pi(2k+1)(2n+1)}{2N}\right) \tag{24}$$

where we use the fact that o-DCT(2N-k-1,N) = - o-DCT(k,N). Thus, with N/4 input and N/2 output additions, the odd cos-DFT of N has been reduced to one of N/2 and an odd DCT of N/4. The imaginary part becomes:

$$\text{osin-DFT}(k,n) = \sum_{n=0}^{N/2-1} x(2n) \sin\left(\frac{2\pi(2k+1)n}{N}\right) + \sum_{n=0}^{N/4-1} (x(2n+1)+x(N-2n-1)) \sin\left(\frac{2\pi(2k+1)(2n+1)}{2N}\right) \quad (25)$$

Using trigonometric identities [9], this is equal to:

$$\text{osin-DFT}(k,n) = \sum_{n=0}^{N/2-1} x(2n) \sin\left(\frac{2\pi(2k+1)n}{N}\right) + \sum_{n=0}^{N/4-1} (-1)^n (x(2n+1)+x(N-2n-1)) \cos\left(\frac{2\pi(N/2-(2k+1))(2n+1)}{2N}\right) \quad (26)$$

or an odd sin-DFT of N/2 and an odd DCT of N/4 at the cost of 3N/4 additions. Turning to the computation of the odd DCT, we use a mapping similar to [3]:

$$y(n) = x(2n) \qquad y(N-n-1) = -x(2n+1) \quad (27)$$

Thus, (22) becomes:

$$\text{o-DCT}(k,n) = \sum_{n=0}^{N-1} x(n) \cos\left(\frac{2\pi(4k+1)(4n+1)}{8N}\right) \qquad k=0..N-1 \quad (28)$$

Now, as seen in [9] o-DCT (k,N) and o-DCT (N-k-1,N) are obtained from ocos-DFT(k,N) and osin-DFT(k,N) by a simple rotation or 3 multiplications and 3 additions.

Evaluating the computational complexity of this approach to the odd DFT computation of length N real signals, it is seen that it requires:

$$N/2\ (\log_2 N-1)\ \text{mu.} \qquad 3N/2\ (\log_2 N-1)\ \text{ad.} \quad (29)$$

Note that when this real odd DFT algorithm is used for complex signals (by transforming separately the real and imaginary part and adding the result) it leads to the same number of multiplies as the Rader/Brenner FFT but to substantially less additions.

V Complexity evaluation and comparison

Using the above introduced odd DFT algorithm leads to the following load for a real DFT of N by N, N a power of 2:

$$N^2/2\ \log_2 N - 7/6\ N^2 + 4\ \text{mu.}$$

$$5N^2/2\ \log_2 N - 13/6\ N^2 + 56/3\ \text{ad.} \quad (30)$$

and, with the optimized mapping for the DCT, the complexity for a real DCT of N by N is:

$$N^2/2\ \log_2 N - 2N + N^2/3 + 8/3\ \text{mu.}$$

$$5N^2/2\ \log_2 N - 6N + N^2/3 + 62/3\ \text{ad.} \quad (31)$$

These results compare favorably with existing algorithms. For completeness, we compare it with:

a) The Chen et al. algorithm for a row/column approach with a 1-D DCT that uses:

$$N\ \log_2 N - 3/2\ N + 4\ \text{mu.} \qquad 3/2\ N\ \log_2 N - 3/2\ N + 2\ \text{ad.} \quad (32)$$

b) The polynomial approach from [7] which is slighty more efficient then the one in [6].

c) A row/column approach with the 1-D DCT algorithm from [9] which uses:

$$N/2\ \log_2 N\ \text{mu.} \qquad N/2\ (3\ \log_2 N-2) + 1\ \text{ad.} \quad (33)$$

d) The proposed method.

The complexities are compared in table 1 and operation counts are given in table 2. Asymptotically, it reduces the number of multiplies by a factor of 4 and saves 1/6 of the adds when compared to currently proposed algorithms (a,b above).

VI Implementation considerations

The above algorithm, while achieving substancial computational savings, has a rather involved structure. But for the expected application (image coding with blocks of 8x8 or 16x16), the transform can be explicitely written in linear code [11]. Therefore, the structural complexity dissappears completely.

Furthermore, using signal processors or specialized processors with a large number of registers (for ex. TMS 320 with 144 registers) allows to perform the whole transform in the registers, thus avoiding the data transfer problem during the transform evaluation. This should lead to fast implementations where the computational savings are fully translated into time savings.

VII Conclusion

A fast 2-D DCT algorithm was proposed which reduces the number of multiplies by 50 to 75 % with comparable number of additions.

This was achieved by showing that a 2-D DCT can be obtained from a real 2-D DFT with 1.5 multiplies per point and by developping efficient real odd DFT algorithms which are used when a 2-D DFT is evaluated through polynomial transforms.

Acknowledgements

The author wishes to thank Prof. H.J. Nussbaumer for commenting the manuscript, the "Fonds National Suisse de la Recherche Scientifique" for supporting this research and Michael Unser for invaluable help.

References:

[1] A.K. Jain, "Image Data Compression : A Review", Proceedings of the IEEE, Vol.69, No.3, March 1981, pp. 349-389.

[2] W-H. Chen, C.H. Smith, and S.C. Fralick, "A Fast Computational Algorithm for the Discrete Cosine Transform", IEEE Trans. on Communications, Vol. COM-25, pp.1004-1009, Sept. 1977.

[3] M.J. Narasimha, and A.M. Peterson, "On the Computation of the Discrete Cosine Transform", IEEE Trans. on Communications, Vol. COM-26, pp. 934-936, June 1978.

[4] H.J. Nussbaumer, "Fast Polynomial Transform Computation of the 2-D DCT", Proc. of the Int. Conf. on Digital Signal Processing, pp. 276-283, Florence, 1981.

[5] H.J. Nussbaumer, Fast Fourier Transform and Convolution Algorithms, 2nd. ed., Springer, Berlin, 1982.

[6] N. Nasrabadi, and R. King, "Computationally Efficient Discrete Cosine Transform Algorithm", Elec. Letters, 6th Jan. 1983, Vol.19, No. 1, pp. 24-25.

[7] S-C. Pei, and E-F. Huang, "Improved 2D Discrete Cosine Transforms Using Generalized Polynomial Transforms and DFT's", Proc. IEEE Int. Conf. on Communications, Amsterdam, 1984, pp. 242-244.

[8] J. Makhoul, "A Fast Cosine Transform in One and Two Dimensions", IEEE Trans. on ASSP, Vol. ASSP-28, No. 1, Feb. 1980, pp. 27-34.

[9] M. Vetterli, and H.J. Nussbaumer, "Simple FFT and DCT Algorithms with Reduced Number of Operations", Signal Processing, Vol. 6, No. 4, July 1984.

[10] H.J. Nussbaumer, and P. Quandalle, "Fast Computation of Discrete Fourier Transforms Using Polynomial Transforms", IEEE Trans. on ASSP, Vol. ASSP-27, pp. 169-181, 1979.

[11] L.R. Morris, "Automatic Generation of Time Efficient Digital Signal Processing Software ", IEEE Trans. Acoust., Speech, Signal Processing, Vol. ASSP-25, pp.74-78, Feb. 1977.

TABLE I: Computational complexity for the algorithms a)-d)

Algorithm	Multiplications	Additions
a)	$2N^2 Log_2 N - 3N^2 + 8N$	$3N^2 Log_2 N - 3N^2 + 4N$
b)	$2N^2 Log_2 N$	$5N^2 Log_2 N - 2N^2$
c)	$N^2 Log_2 N$	$3N^2 Log_2 N - 2N^2 + 2N$
d)	$\frac{N^2}{2} Log_2 N + \frac{N^2}{3} - 2N + \frac{8}{3}$	$\frac{5N^2}{2} Log_2 + \frac{N^2}{3} - 6N + \frac{62}{3}$

TABLE II: Operation counts for the algorithms a)-d)

	a)		b)		c)		d)	
N	mults	adds	mults	adds	mults	adds	mults	adds
8	256	416	384	832	192	464	104	474
16	1408	2368	2048	4608	1024	2592	568	2570
32	7424	12416	10240	23552	5120	13376	2840	12970
64	37376	61696	49152	114688	24576	65664	13528	62442
128	181248	295424	229376	540672	114688	311552	62552	291434
256	854016	1377280	1048576	2490368	524288	1442304	283480	133105