Express Letter

Efficient Implementation of Discrete Cosine Transform using Recursive Filter Structure

Yuk-Hee Chan, Lap-Pui Chau and Wan-Chi Siu

Abstract— **In this paper, we generalize a formulation for converting a length-2^n discrete cosine transform into n groups of equations, then apply a novel technique for its implementation. The sizes of the groups are $2^{n-1}, 2^{n-2}, \ldots 2^0$ respectively, while their structures are extremely regular. The realization can then be converted into recursive filter form, which is of particularly simple for practical implementation.**

I. Introduction

The discrete cosine transform [1] is widely used in digital signal processing, particularly for digital image processing. Because of the complicated computational complexity, many efficient algorithms were proposed to improve the computing speed and hardware complexity. These algorithms can broadly be classified into the following categories: 1) indirect computation through the discrete Fourier transform or the Walsh-Hadamard transform [2]–[4], 2) direct factorization [5]–[8], and 3) recursive computation [9]–[12]. Among them, the ones using indirect computation method often involve extra operations. The direct factorization decomposes the DCT directly, so that the total number of operations can be reduced. By implementing the recursive structure in an effective way, a regular and parallel VLSI structure can possibly be used and the computational complexity is greatly reduced.

In this paper, we present a formulation for converting a length-2^n DCT into n groups of equations and the sizes of the groups are $2^{n-1}, 2^{n-2}, \ldots, 2^0$ respectively. The resultant formulation is extremely regular, which is suitable for the implementation using a recursive filter structure. Furthermore, the beauty of the formulation is enhanced by expanding the multiple angle cosine function into a series of high order cosine functions to effect the realization of the recursive filter structure.

II. Derivation of Algorithm

The DCT of a data sequence $\{x^0(i): i = 0, 1, \ldots, N-1\}$ can be written as

$$Y(k) = \sum_{i=0}^{N-1} x^0(i) \cos\left(\frac{(2i+1)k\pi}{2N}\right) \tag{1}$$

for $k = 0, 1, \ldots, N-1$, where $N = 2^n$, n is an integer and the index 0 of x gives the stage of data representation (see below).

For the convenience of realization, let us introduce a formulation, such that $Y(k)$ is split into n groups, namely

$$Y(2r+1), Y(2(2r+1)), \cdots, Y(2^{n-1}(2r+1)) \text{ and } Y(0)$$
$$\text{for } r = 0, 1, \ldots, 2^{n-(m+1)} - 1.$$

Let us rewrite $Y(k)$ in the form of $Y(2^m(2r+1))$ and make some simplifications, for $m = 0, 1, \ldots, n-1$, where m is the group number starting from zero

$$Y(2^m(2r+1)) = \sum_{i=0}^{N-1} x^0(i) \cos\left((2i+1)\frac{2^m(2r+1)\pi}{2N}\right)$$
$$= \sum_{i=0}^{N-1} x^0(i) \cos((2i+1)2^m\theta_r)$$
$$\text{for } \theta_r = \frac{(2r+1)\pi}{2N}. \tag{2}$$

We may rearrange the order of computation of the lower half of the LHS, hence,

$$\begin{aligned} &Y(2^m(2r+1)) \\ &= \sum_{i=0}^{N/2-1} x^0(i) \cos((2i+1)2^m\theta_r) \\ &\quad + \sum_{i=0}^{N/2-1} x^0(N-1-i) \cos((2N-2i-1)2^m\theta_r) \\ &= \sum_{i=0}^{N/2-1} (x^0(i) \cos((2i+1)2^m\theta_r) \\ &\quad + x^0(N-1-i) \cos((2N-2i-1)2^m\theta_r)). \end{aligned} \tag{3}$$

Now let us make use of the property of the factor 2^m in the formulation that $\sin(2^m\theta_r t) = 0$, for t is any multiple of $2N/2^m$. Hence (3) becomes

$$\begin{aligned} &Y(2^m(2r+1)) \\ &= \sum_{i=0}^{N/2-1} (x^0(i) + x^0(N-1-i) \\ &\quad \cdot \cos(2^m(2r+1)\pi)) \cos((2i+1)2^m\theta_r). \end{aligned} \tag{4}$$

In order to see further decomposition, let us formulate (4) exactly in the form of (2). Substitute $[x^0(i) + x^0(N-1-i)]$ by $x^1(i)$ into (4), we have

$$Y(2^m(2r+1)) = \sum_{i=0}^{N/2-1} (x^1(i)) \cos((2i+1)2^m\theta_r). \tag{5}$$

It is clear that a similar procedure can be used to make further decomposition of (5). Hence we have

$$Y(2^m(2r+1)) = \sum_{i=0}^{N/4-1} (x^2(i)) \cos((2i+1)2^m\theta_r)$$

and so on

In general we have

$$Y(2^m(2r+1)) = \sum_{i=0}^{N/2^{m+1}-1} \left(x^m(i) - x^m\left(\frac{N}{2^m} - 1 - i\right)\right) \cdot \cos((2i+1)2^m\theta_r) \tag{6}$$

where

$$x^{m+1}(i) = x^m(i) + x^m\left(\frac{N}{2^m} - 1 - i\right). \tag{7}$$

Eq. (6) gives a decomposition equation for $Y(2^m(2r+1))$, for $m = 0, 1, \ldots, n-1$ and $r = 0, 1, \ldots, (N2^{-m-1} - 1)$. This means that there are n groups of equations and each of which gives $2^{n-(m+1)}$ results of $Y(k)$'s. The number of multiplications for each

Manuscript received May 2, 1994; revised September 10, 1994. This paper was recommended by Dr. Ming-Ting Sun.

The authors are with the Department of Electronic Engineering, Hong Kong Polytechnic, Hong Hom, Hong Kong.

IEEE Log Number 9406422.

equation in a group is $2^{n-(m+1)}$ and only half of the number of multiplications are required as compared to the previous group. Let us clarify our ideas with an example. If $N = 8, Y(2^m(2r+1))$ can be expressed as follows,

for $m = 0$,

$$Y(2r+1) = \sum_{i=0}^{3}(x^0(i) - x^0(7-i)) \cdot \cos\left((2i+1)\frac{(2r+1)\pi}{16}\right) \quad \text{where } r = 0, 1, 2, 3 \quad (8)$$

for $m = 1$,

$$Y(2(2r+1)) = \sum_{i=0}^{1}(x^1(i) - x^1(3-i)) \cdot \cos\left((2i+1)\frac{2(2r+1)\pi}{16}\right) \quad \text{where } r = 0, 1 \quad (9)$$

for $m = 2$,

$$Y(4(2r+1)) = \sum_{i=0}^{0}(x^2(i) - x^2(1-i)) \cdot \cos\left((2i+1)\frac{4(2r+1)\pi}{16}\right) \quad \text{where } r = 0 \quad (10)$$

and $Y(0) = x^3(0)$.

Hence 4, 2, 1 and 0 multiplications are required for expressions in groups with $m = 0, 1.2$ and 3 respectively.

III. Recursive Filter Form

It is obvious that the arguments of the cosine terms in (8)–(10) are with similar kernels, so these regular structures are possible for VLSI implementation. Our purpose is to realize the equation in a recursive structure. The technique for which we suggest here is to convert the arguments of the cosine terms in (6) into a series of high order expressions.[1]

Let $\theta_{m,r} = (2^m(2r+1)\pi)/2N$, hence

$$Y(2^m(2r+1)) = \sum_{i=0}^{N/2^{m+1}-1}\left(x^m(i) - x^m\left(\frac{N}{2^m} - 1 - i\right)\right) \cdot \cos((2i+1)\theta_{m,r}) \quad (11)$$

Note that $\cos((2i+1)\theta_{m,r}) = \sum_{j=0}^{i} A_{ij}\cos^{2j+1}\theta_{m,r}$ where A_{ij}'s are some well-defined integers. For the example $N = 8$ again, the 4×4 matrix A_{ij} is defined as:

$$\begin{pmatrix} A_{00} & A_{01} & A_{02} & A_{03} \\ A_{10} & A_{11} & A_{12} & A_{13} \\ A_{20} & A_{21} & A_{22} & A_{23} \\ A_{30} & A_{31} & A_{32} & A_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -3 & 4 & 0 & 0 \\ 5 & -20 & 16 & 0 \\ -7 & 56 & -112 & 64 \end{pmatrix} \quad (12)$$

Eq. (11) can then be written as:

$$Y(2^m(2r+1)) = \sum_{i=0}^{N/2^{m+1}-1}\left(x^m(i) - x^m\left(\frac{N}{2^m} - 1 - i\right)\right) \cdot \sum_{j=0}^{i} A_{ij}\cos^{2j+1}\theta_{m,r}$$

$$= \sum_{j=0}^{N/2^{m+1}-1}\sum_{i=j}^{N/2^{m+1}-1}(x^m(i) - x^m \cdot \left(\frac{N}{2^m} - 1 - i\right))A_{ij}\cos^{2j+1}\theta_{m,r} \quad (13)$$

or

$$Y(2^m(2r+1)) = \sum_{j=0}^{N/2^{m+1}-1} g^m(j)\cos^{2j+1}\theta_{m,r} \quad (14)$$

where

$$g^m(j) = \sum_{i=j}^{\frac{N}{2^{m+1}}-1}\left(x^m(i) - x^m\left(\frac{N}{2^m} - 1 - i\right)\right)A_{ij}. \quad (15)$$

Eqs. (14) and (15) are our final equations for the realization of the DCT. Eq. (14) looks simpler than (11) because it is now represented by a series of high order cosine terms. It can be considered as a recursive formulation which requires simple structure for its realization, while (15) involves some pre-processing before feeding data into the recursive formulation. Surprisingly, no data multiplication might be required for the realization of (15) for a careful examination of its structure as shown below.

IV. Realization

i) Data pre-processing

For this part of the realizations, we have to implement (15). For the simplicity of our discussion, let us use $n = 3$, hence $N = 8$ for our analysis. In this case, we have to consider cases with $m = 0, 1$ and 2. For $m = 0$,

$$g^0(j) = [x^0(0) - x^0(7)]A_{0j} + [x^0(1) - x^0(6)]A_{1j} + [x^0(2) - x^0(5)]A_{2j} + [x^0(3) - x^0(4)]A_{3j}$$

for $j = 0$, 1, 2, and 3

For $m = 1$,

$$g^1(j) = [x^1(0) - x^1(3)]A_{0j} + [x^1(1) - x^1(2)]A_{1j}$$

for $j = 0$, 1 and $x^1(0) = x^0(0) + x^0(7)$,

$\cdot\, x^1(1) = x^0(1) + x^0(6)$

$\cdot\, x^1(2) = x^0(2) + x^0(5), x^1(3) = x^0(3) + x^0(4)$

For $m = 2$,

$$g^2(0) = [x^2(0) - x^2(1)]A_{00}$$

for $x^2(0) = x^1(0) + x^1(3), x^2(1) = x^1(1) + x^1(2)$

Note also that A_01, A_02, A_03, A_12, A_13, and A_23 are of zero values, and $Y(0) = x^2(0) + x^2(1)$ without further processing.

Hence this part of the realization involves additions mainly. It appears that some multiplications of the factors A_{ij} as shown below can be converted into simple adds and shifts which can easily be realized using hardware techniques or the machine language of a CPU. Let us rewrite (12) as (see bottom of this page)

Hence, A_{ij}'s involve no real multiplications, however for long lengths of the DCT, some terms may require more than one shift/add operations for their implementation.

[1] $\cos 3\theta = 4\cos^3\theta - 3\cos\theta$
$\cos 5\theta = 16\cos^5\theta - 20\cos^3\theta + 5\cos\theta$
$\cos 7\theta = 64\cos^7\theta - 112\cos^5\theta + 56\cos^3\theta - 7\cos\theta$.

$$\begin{pmatrix} A_{00} & A_{01} & A_{02} & A_{03} \\ A_{10} & A_{11} & A_{12} & A_{13} \\ A_{20} & A_{21} & A_{22} & A_{23} \\ A_{30} & A_{31} & A_{32} & A_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1-2^2 & 2^2 & 0 & 0 \\ 2^2+1 & -(2^4+2^2) & 2^4 & 0 \\ 1-2^3 & 2^6-2^3 & 2^4-2^7 & 2^6 \end{pmatrix}$$

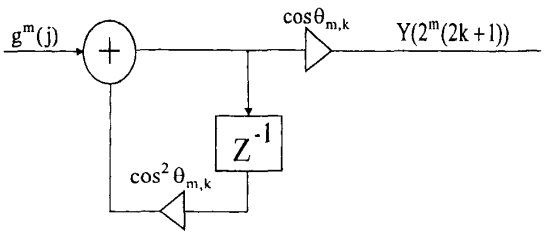

Fig. 1. Block diagram of the recursive filter.

ii) Recursive Computation

Eq. (14) can be considered as a recursive filter. Again let us take $N = 8$ as an example. Hence we have,

for $m = 0$,

$$Y(2r+1) = (((g^0(3)\cos^2\theta_{0,r} + g^0(2))\cos^2\theta_{0,r} + g^0(1)) \cdot \cos^2\theta_{0,r} + g^0(0))\cos\theta_{0,r}$$

for $r = 0$, 1, 2, and 3

for $m = 1$,

$$Y(2(2r+1)) = (g^1(1)\cos^2\theta_{1,r} + g^1(0))\cos\theta_{1,r}$$

for $r = 0$ and 1

for $m = 2$,

$$Y(2^2(2r+1)) = g^2(0)\cos\theta_{2,r} \quad \text{for } r = 0$$

where

$$\theta_{m,r} = \frac{2^m(2r+1)\pi}{16}.$$

Hence a total of 21 multiplications are required, which represents a number larger than those required for approaches [4]–[5] which are to optimize the number of operations. The major advantage of the present realizations using the recursive filter structure is its simplicity. The last point can be seen in Fig. 1.

It is seen that a single first order recursive filter is enough for its realization, which represents almost the simplest possible structure for the realization of the discrete cosine transform.

There are not many recursive filter algorithms for the computation of the DCT appeared in the literature. However, we could still recall the ones that are available in the literature for a comparison. Canaris [13] used Goertzel's [14] algorithm to implements the DCT with a second order recursive filter structure, but it requires a large number of multipliers. A second order recursive filter structure has also been proposed by Chau and Siu [12], the structure is regular and requires less multipliers as compared to Canaris' approach. However all these algorithms involve second order structures, hence extra buffers and longer computation time are required. In this paper, we have successfully derived a novel first order recursive filter structure to compute the DCT, which can be used to resolve the above problems and obviously gives a significant improvement over the previous formulations.

V. Conclusion

This paper gives a formulation to convert a length-2^n DCT into n groups of equations, then applies a novel technique for its efficient realization. The resultant structure is a first order recursive filter which represents almost the simplest possible formulation of any DSP system. Furthermore the filter structure is numerically stable, since it involves no division at all.

References

[1] N. Ahmed, T. Natarajan and K. R. Rao, "Discrete cosine transform," *IEEE Trans. on Computer,* vol. 23, no. 1, pp. 90–93, Jan. 1974.
[2] B. D. Tseng and W. C. Miller, "On computing the discrete cosine transform," *IEEE Trans. on Computer,* vol. 27, pp. 966–968, Oct. 1978.
[3] J. Makhoul, "A fast cosine transform in one and two dimensions," *IEEE Trans. on Acoustics, Speech and Signal Processing,* vol. 28, pp. 27–34, Feb. 1980.
[4] M. Vetterli and H. Nussbaumer, "Simple FFT and DCT algorithm with reduced number of operations," *Signal Processing,* vol. 6, no. 4, pp. 267–278, Aug. 1984.
[5] B. G. Lee, "A new algorithm to compute the discrete cosine transform," *IEEE Trans. on Acoustics, Speech and Signal Processing,* vol. 32, no. 6, pp. 1243–1245, Dec. 1984.
[6] M. T. Sun, L. Wu, and M. L. Liou, "A concurrent architecture for VLSI implementation of discrete cosine transform," *IEEE Trans. on Circuits and Systems,* vol. 34, pp. 992–994, 1987.
[7] P. Duhamel and H. H'Mida, "New 2^n DCT algorithms suitable for VLSI implementation," Proceedings, *IEEE International Conference on Acoustics, Speech and Signal Processing,* pp. 1805–1808, 1987.
[8] Y. H. Chan and W. C. Siu, "On the realization of discrete cosine transform using the distributed arithmetic," *IEEE Trans. on Circuits and Systems*-Part I, vol. 39, no. 9, pp. 705–712, Sept. 1992.
[9] W. Ma and R. Yiu, "New Recursive Factorization Algorithms To Compute DFT 2^m and DCT 2^m," *IEEE Asian Electronics Conference* 1987, pp. 71–76.
[10] H. S. Hou, "A fast recursive algorithm for computing the discrete cosine transform," *IEEE Trans. on Acoustics, Speech and Signal Processing,* vol. 35, pp. 1455–1461, 1987.
[11] N. I. Cho and S. U. Lee, "A Fast 4×4 Algorithm for the Recursive 2-D DCT," *IEEE Trans. on Signal Processing,* vol. 40, pp. 2166–2172, 1992.
[12] L. P. Chau and W. C. Siu, "Recursive algorithm for the discrete cosine transform with general lengths," *Electronics Letters.* vol. 30, no. 3, pp. 197–198, Feb. 1994, Errata: vol. 30, no. 7, pp. 608, 1994.
[13] J. Canaris, "A VLSI architecture for the real time computation of discrete trigonometric transforms," *J. of VLSI Signal Processing,* vol. 5, pp. 95–104, 1993.
[14] G. Goertzel, "An algorithm for the evaluation of finite trigonometric series," *Amer. Math. Monthly,* vol. 65, pp.34–35, 1958.

A Modified Moment-Based Edge Operator for Rectangular Pixel Image

Li-Min Luo, Xiao-Hua Xie, and Xu-Dong Bao

I. Introduction

An important operation in image processing is extracting edges from gray images. There are many methods to determine the location, orientation, and strength of an edge in digital images. The moment-based method has relatively good performance in the aspects of location precision and noise robustness [1]. When detecting edges in a digital image, the pixels are generally assumed square. However, this

Manuscript received January 19, 1994; revised July 14, 1994. It was recommended by Wen H. Chen.

The authors are with the Department of Biological Science and Medical Engineering, Southeast University, Nanjing 210018, People's Republic of China.

IEEE Log Number 9406760.