FAST ALGORITHM FOR THE 3-D DCT

O. Alshibami and *S. Boussakta*

Institute of Integrated Information Systems
School of Electronic and Electrical Engineering
University of Leeds, Leeds,
LS2 9JT, United Kingdom
E-mail: eenoha@leeds.ac.uk and s.boussakta@ee.leeds.ac.uk

ABSTRACT

The three-dimensional discrete cosine transform (3-D DCT) has been used in many 3-D applications such as video coding and compression. Many fast algorithms have been developed for the calculation of 1-D DCT. These algorithms are then used for the calculation of 3-D DCT using the row-column approach. However, 3-D algorithms involve less arithmetic operations and can be faster. In this paper, the 3-D decimation in frequency vector-radix algorithm (3-D DIF VR), for the 3-D DCT-II, is developed and its arithmetic complexity analysed and compared to similar algorithms. In Comparison with the familiar row-column approach, the 3-D vector-radix reduces the number of multiplications significantly while keeping the number of additions the same and hence can be used for fast 3-D image and video coding and compression.

1. INTRODUCTION

The discrete cosine transform (DCT) is one of the most popular transforms in the field of digital signal processing and communications [1]. It has been widely used in many applications such as speech and image compression [1,2]. Its use has been expanded to cover three-dimensional applications such as 3-D image and video compression and coding [3-5]. In these applications, the 3-D image is divided into 8×8×8 or 16×16×16 cubes which are then transformed using the 3-D DCT. Because the 3-D cosine transform is separable, it is usually calculated using algorithms developed for the 1-D transform in a row-column approach. However, three-dimensional algorithms can be faster and involves fewer arithmetic operations [6].

Therefore, it is the aim of this paper to develop the 3-D decimation-in-frequency vector-radix algorithm (3-D DIF VR) for fast calculation of the 3-D type-II discrete cosine transform. Compared to the row-column approach, the 3-D vector-radix reduces the number of multiplications significantly while keeping the number of additions the same.

2. THE 3-D DCT-II

The 3-D type-II discrete cosine transform of *x(n₁,n₂,n₃)*, of size $N_1 \times N_2 \times N_3$, can be defined as:

$$X(k_1,k_2,k_3) = f_{3D} \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}\sum_{n_3=0}^{N_3-1} x(n_1,n_2,n_3) \cos(\alpha_1 k_1)\cos(\alpha_2 k_2)\cos(\alpha_3 k_3) \qquad (1)$$

$$k_i = 0,1,\ldots,N_i-1$$

where

$$\alpha_i = \frac{\pi(2n_i+1)}{2N_i} \qquad i=1,2,3$$

$$f_{3D} = \frac{8}{N_1 N_2 N_3}\varepsilon_{k1}\varepsilon_{k2}\varepsilon_{k3}$$

and

$$\varepsilon_{ki} = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } k_1,k_2,k_3=0 \\ 1 & \text{otherwise} \end{cases} \qquad i=1,2,3$$

3. THE 3-D DIF VR ALGORITHM

The 3-D DCT is usually computed using algorithms developed for the 1-D DCT applied over each dimension successively in a row-column style [3-5]. However multidimensional algorithms involve less arithmetic operations and can be faster [6,7-9].

In this paper, a 3-D decimation in frequency vector-radix algorithm that calculates the 3-D DCT-II directly is introduced. In this algorithm, the N×N×N 3-D DCT-II is first decomposed into eight N/2×N/2×N/2-point 3-D DCTs. Each N/2×N/2×N/2 3-D DCT is then divided further until we get 2×2×2 transforms.

Let $N_1=N_2=N_3=N$ and assume that the factor f_{3D} is merged into $X(k_1,k_2,k_3)$. First, we need to rearrange the input data, $x(n_1,n_2,n_3)$, as follows:

$$\tilde{x}(n_1,n_2,n_3)=\begin{cases} x(2n_1,2n_2,2n_3) \\ \quad 0\le n_1,n_2,n_3\le N/2-1 \\ x(2n_1,2n_2,2N-2n_3-1) \\ \quad 0\le n_1,n_2\le N/2-1,\ N/2\le n_3\le N-1 \\ x(2n_1,2N-2n_2-1,2n_3) \\ \quad 0\le n_1,n_3\le N/2-1,\ N/2\le n_2\le N-1 \\ x(2n_1,2N-2n_2-1,2N-2n_3-1) \\ \quad 0\le n_1\le N/2-1,\ N/2\le n_2,n_3\le N-1 \\ x(2N-2n_1-1,2n_2,2n_3) \\ \quad N/2\le n_1\le N-1, 0\le n_2,n_3\le N/2-1 \\ x(2N-2n_1-1,2n_2,2N-2n_3-1) \\ \quad N/2\le n_1,n_3\le N-1, 0\le n_2\le N/2-1 \\ x(2N-2n_1-1,2N-2n_2-1,2n_3) \\ \quad N/2\le n_1,n_2\le N-1, 0\le n_3\le N/2-1 \\ x(2N-2n_1-1,2N-2n_2-1,2N-2n_3-1) \\ \quad N/2\le n_1,n_2,n_3\le N-1 \end{cases} \quad (2)$$

Replacing *x(n₁,n₂,n₃)* in Eq. (1) by $\tilde{x}(n_1,n_2,n_3)$ in Eq. (2), the 3-D DCT-II can be written as:

$$X(k_1,k_2,k_3)=\sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N-1}\sum_{n_3=0}^{N-1}\tilde{x}(n_1,n_2,n_3) \cos(\phi_1 k_1)\cos(\phi_2 k_2)\cos(\phi_3 k_3) \quad (3)$$

where $\phi_i=\pi(4n_i+1)/2N \quad , \quad i=1,2,3$

If we consider the even and the odd parts of k_1, k_2, and k_3, the general formula for the calculation of the 3-D DCT-II can be written as:

$$X(2k_1+i,2k_2+j,2k_3+l) = \sum_{n_1=0}^{N/2-1}\sum_{n_2=0}^{N/2-1}\sum_{n_3=0}^{N/2-1}\tilde{x}_{ijl}(n_1,n_2,n_3)\cos(\phi_1(2k_1+i)) \cos(\phi_2(2k_2+j))\cos(\phi_3(2k_3+l)) \quad (4)$$

where

$$\begin{aligned}\tilde{x}_{ijl}(n_1,n_2,n_3) &= \tilde{x}(n_1,n_2,n_3) \\ &+(-1)^{l}\,\tilde{x}(n_1,n_2,n_3+n/2) \\ &+(-1)^{j}\,\tilde{x}(n_1,n_2+n/2,n_3) \\ &+(-1)^{j+l}\,\tilde{x}(n_1,n_2+n/2,n_3+n/2) \\ &+(-1)^{i}\,\tilde{x}(n_1+n/2,n_2,n_3) \\ &+(-1)^{i+l}\,\tilde{x}(n_1+n/2,n_2,n_3+n/2) \\ &+(-1)^{i+j}\,\tilde{x}(n_1+n/2,n_2+n/2,n_3) \\ &+(-1)^{i+j+l}\,\tilde{x}(n_1+n/2,n_2+n/2,n_3+n/2)\end{aligned} \quad (5)$$

i, *j* and *l* equal zero for even indices and 1 for odd indices.

Using the following trigonometric identities:

$$\cos((2k_i+1)\phi_i)=2\cos\phi_i\cos(2k_i\phi_i) - \cos((2k_i-1)\phi_i) \quad (6)$$

$$\begin{aligned}&\cos((2k_i+1)\phi_i)\cos((2k_j+1)\phi_j) \\ &= 4\cos\phi_i\cos\phi_j\cos(2k_i\phi_i)\cos(2k_j\phi_j) \\ &-\cos((2k_i-1)\phi_i)\cos((2k_j+1)\phi_j) \\ &-\cos((2k_i+1)\phi_i)\cos((2k_j-1)\phi_j) \\ &-\cos((2k_i-1)\phi_i)\cos((2k_j-1)\phi_j)\end{aligned} \quad (7)$$

and

$$\begin{aligned}&\cos((2k_i+1)\phi_i)\cos((2k_j+1)\phi_j)\cos((2k_l+1)\phi_l) \\ &=8\cos\phi_i\cos\phi_j\cos\phi_l\cos(2k_i\phi_i)\cos(2k_j\phi_j)\cos(2k_l\phi_l) \\ &-\cos((2k_i+1)\phi_i)\cos((2k_j+1)\phi_j)\cos((2k_l-1)\phi_l) \\ &-\cos((2k_i+1)\phi_i)\cos((2k_j-1)\phi_j)\cos((2k_l+1)\phi_l) \\ &-\cos((2k_i+1)\phi_i)\cos((2k_j-1)\phi_j)\cos((2k_l-1)\phi_l) \\ &-\cos((2k_i-1)\phi_i)\cos((2k_j+1)\phi_j)\cos((2k_l+1)\phi_l) \\ &-\cos((2k_i-1)\phi_i)\cos((2k_j+1)\phi_j)\cos((2k_l-1)\phi_l) \\ &-\cos((2k_i-1)\phi_i)\cos((2k_j-1)\phi_j)\cos((2k_l+1)\phi_l) \\ &-\cos((2k_i-1)\phi_i)\cos((2k_j-1)\phi_j)\cos((2k_l-1)\phi_l)\end{aligned} \quad (8)$$

The general equation for the 3-D DIF VR algorithm for the 3-D DCT-II can be written as shown in Eq. (9):

$$\begin{bmatrix} X(2k_1,2k_2,2k_3) \\ X(2k_1,2k_2,2k_3+1) \\ X(2k_1,2k_2+1,2k_3) \\ X(2k_1,2k_2+1,k_3+1) \\ X(2k_1+1,2k_2,2k_3) \\ X(2k_1+1,2k_2,2k_3+1) \\ X(2k_1+1,2k_2+1,2k_3) \\ X(2k_1+1,2k_2+1,2k_3+1) \end{bmatrix} = \underbrace{\begin{bmatrix} X_{000}(2k_1,2k_2,2k_3) \\ X_{001}(2k_1,2k_2,2k_3) \\ X_{010}(2k_1,2k_2,2k_3) \\ X_{011}(2k_1,2k_2,2k_3) \\ X_{100}(2k_1,2k_2,2k_3) \\ X_{101}(2k_1,2k_2,2k_3) \\ X_{110}(2k_1,2k_2,2k_3) \\ X_{111}(2k_1,2k_2,2k_3) \end{bmatrix}}_{Butterflies\ Calculation} + \underbrace{\begin{bmatrix} 0 \\ -X(2k_1,2k_2,2k_3-1) \\ -X(2k_1,2k_2-1,2k_3) \\ -X(2k_1,2k_2-1,2k_3+1)-X(2k_1,2k_2+1,2k_3-1)-X(2k_1,2k_2-1,2k_3-1) \\ -X(2k_1-1,2k_2,2k_3) \\ -X(2k_1-1,2k_2,2k_3+1)-X(2k_1+1,2k_2,2k_3-1)-X(2k_1-1,2k_2,2k_3-1) \\ -X(2k_1-1,2k_2+1,2k_3)-X(2k_1+1,2k_2-1,2k_3)-X(2k_1-1,2k_2-1,2k_3) \\ X_{r111} \end{bmatrix}}_{Recursive\ Additions} \quad (9)$$

where

$$X_{abc}(2k_1,2k_2,2k_3) = \sum_{n_1=0}^{N/2-1}\sum_{n_2=0}^{N/2-1}\sum_{n_3=0}^{N/2-1}\tilde{x}_{abc}(n_1,n_2,n_3)$$
$$(2\cos\phi_1)^a(2\cos\phi_2)^b(2\cos\phi_3)^c \qquad (10)$$
$$\cos(\phi_1 2k_1)\cos(\phi_2 2k_2)\cos(\phi_3 2k_3)$$

and

$$X_{r111} = -X(2k_1+1,2k_2+1,2k_3-1)$$
$$-X(2k_1+1,2k_2-1,2k_3+1) - X(2k_1+1,2k_2-1,2k_3-1)$$
$$-X(2k_1-1,2k_2+1,2k_3+1) - X(2k_1-1,2k_2+1,2k_3-1)$$
$$-X(2k_1-1,2k_2-1,2k_3+1) - X(2k_1-1,2k_2-1,2k_3-1)$$
$$(11)$$

4. ARITHMETIC COMPLEXITY

Figure 1 shows the required stages to calculate the 3-D DCT-II using the developed 3-D DIF VR algorithm. It consists of four stages. Firstly, the 3-D input array is rearranged according to Eq. (2). The rearranged data is then applied to the second stage, which is the butterflies calculation. The input data to this stage is calculated using eight-points butterfly unit as shown in Figure 2. Each butterfly involves 7 real multiplications and 24 real additions. The whole transform requires $(\log_2 N)\times(N^3/8)$ butterflies. The output data from the second stage is then bit-reversed. The recursive additions described in the second part of Eq. (9) is calculated in the last stage.

The calculation of the whole transform using a single butterfly requires:

Multiplications = $(7/8)\,N^3\log_2 N$

Additions = $(9/2)N^3\log_2 N-3N^3+3N^2$

On the other hand, if the 3-D DCT-II is calculated using the row column approach, the number of real multiplications, based on 1-D DCT-II in [10], will be $(3/2)N^3\log_2 N$ and the total number of real additions will be $(9/2)N^3\log_2 N-3N^3+3N^2$.

Table 1 shows the operations count for both the row-column method and the new 3-D DIF VR algorithm using a single butterfly. The 3-D DIF VR algorithm reduces the number of multiplications significantly while keeping the total number of additions the same.

5. CONCLUSION

In this paper, a fast three-dimensional decimation in frequency vector-radix algorithm for the 3-D DCT-II is developed and implemented. This algorithm involves the same number of additions as the familiar row-column approach, but reduces the number of multiplications considerably and hence can be used for fast 3-D image and video coding.

Table 1. Comparison between the row-column and the 3-D vector-radix algorithms based on a single butterfly.

Transform Size $N_1\times N_2\times N_3$	**Row-column approach**		**3-D vector-radix algorithm**	
	Mults./point	Adds. /point	Mults. /point	Adds. /point
$2^2\times2^2\times2^2$	3	6.75	1.75	6.75
$2^3\times2^3\times2^3$	4.5	10.88	2.625	10.88
$2^4\times2^4\times2^4$	6	15.19	3.5	15.19
$2^5\times2^5\times2^5$	7.5	19.59	4.375	19.59
$2^6\times2^6\times2^6$	9	24.05	5.25	24.05
$2^7\times2^7\times2^7$	10.5	28.52	6.125	28.52
$2^8\times2^8\times2^8$	12	33.01	7	33.01
$2^9\times2^9\times2^9$	13.5	37.51	7.875	37.51
$2^{10}\times2^{10}\times2^{10}$	15	42	8.75	42

6. REFERENCES

[1] K. R. Rao, and P. Yip, *Discrete cosine transform: algorithms, advantages, and applications*. Academic Press Inc., Sept. 1990.

[2] W. Kuo, *Digital Image Compression: Algorithms and Standards.* Kluwer Academic Publishers, Boston, 1995.

[3] S. Tai, Y. Wu, and C. Lin, “An adaptive 3-D discrete cosine transform coder for medical image compression”, *IEEE Trans. Information Technology in Biomedicine*, vol. 4, pp. 259-263, Sept 2000.

[4] M. Servais, and G. De Jager, “Video compression using the three dimensional discrete cosine transform (3D-DCT)”, *Proc. of COMSIG '97*, South African, 1997, pp. 27-32.

[5] R. Westwater, and B. Furht, “The XYZ algorithm for real-time compression of full-motion video”, *Real-Time Imaging*, vol. 2, pp. 19-34, 1996.

[6] S. Boussakta, and O. Alshibami, “fast algorithm for the 3-D discrete Hartley transform”, *ICASSP-2000*, 5-9 June 2000, Istanbul, Turkey, pp. 2302-2305.

[7] S. C. Chan and K. L. Ho, “A new two-dimensional fast cosine transform

algorithm", *IEEE Trans. Signal Processing*, vol. 39, No. 2, pp. 481-485, Feb. 1991.

[8] G. Bi, G. Li, K. Ma, and T. C. Tan, "On the computation of two-dimensional DCT", *IEEE Trans. Signal Processing*, vol. 48, No. 4, April 2000.

[9] S. C. Chan and K. L. Ho, "Direct methods for computing discrete sinusoidal transforms", *IEE Proc. Radar and Signal Processing*, vol. 137, No. 6, pp. 433-442, Dec.1990.

[10] H. S. Hou, "A fast recursive algorithm for computing the discrete cosine transform", *IEEE Trans. Acoust., Speech, and Signal processing*, vol. 33, pp. 1532-1539, Dec. 1985.

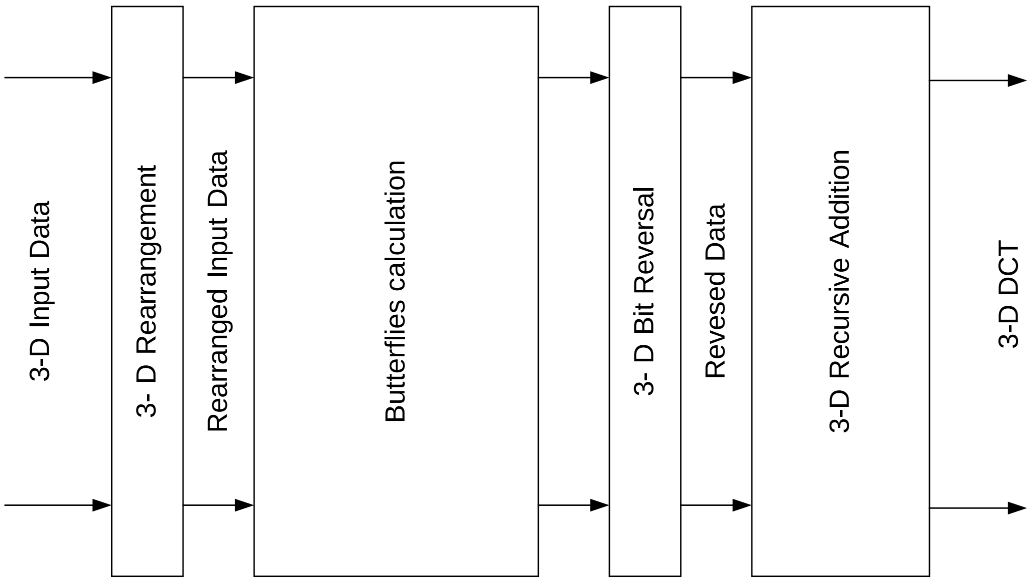

Figure 1. Calculation process for the 3-D DCT-II using the 3-D DIF VR algorithm.

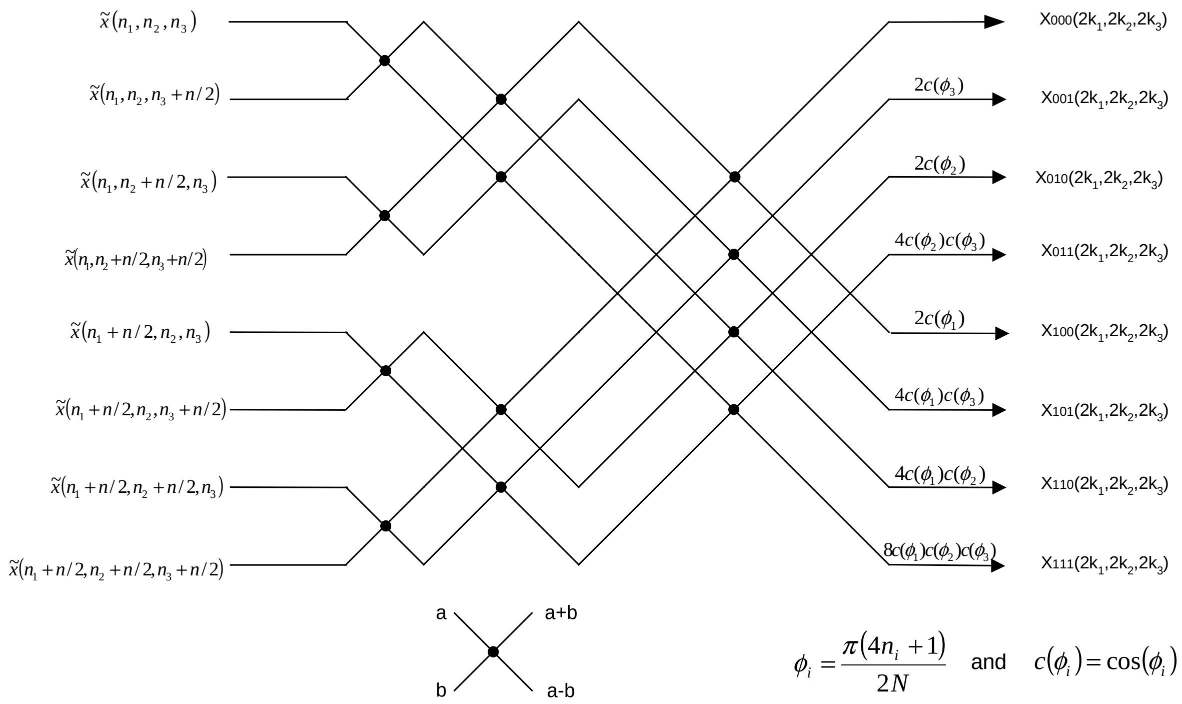
$\tilde{x}(n_1,n_2,n_3)$
$\tilde{x}(n_1,n_2,n_3+n/2)$
$\tilde{x}(n_1,n_2+n/2,n_3)$
$\tilde{x}(n_1,n_2+n/2,n_3+n/2)$
$\tilde{x}(n_1+n/2,n_2,n_3)$
$\tilde{x}(n_1+n/2,n_2,n_3+n/2)$
$\tilde{x}(n_1+n/2,n_2+n/2,n_3)$
$\tilde{x}(n_1+n/2,n_2+n/2,n_3+n/2)$
$2c(\phi_3)$
$2c(\phi_2)$
$4c(\phi_2)c(\phi_3)$
$2c(\phi_1)$
$4c(\phi_1)c(\phi_3)$
$4c(\phi_1)c(\phi_2)$
$8c(\phi_1)c(\phi_2)c(\phi_3)$
X000(2k1,2k2,2k3)
X001(2k1,2k2,2k3)
X010(2k1,2k2,2k3)
X011(2k1,2k2,2k3)
X100(2k1,2k2,2k3)
X101(2k1,2k2,2k3)
X110(2k1,2k2,2k3)
X111(2k1,2k2,2k3)
a
a+b
b
a-b
$\phi_i = \frac{\pi(4n_i+1)}{2N}$ and $c(\phi_i) = \cos(\phi_i)$

Figure 2. Single butterfly for the 3-D DIF VR.