論文2003-40CI-6-14

복소연산이 없는 Polynomial 변환을 이용한 2차원 고속 DCT

(Two dimensional Fast DCT using Polynomial Transform without Complex Computations)

崔 桓 碩 *, 金 元 河 **

(Hwan-Serk Choi and Won-Ha Kim)

요 약

본 논문은 2차원 Discrete Cosine Transform (2D-DCT)의 계산을 새로운 Polynomial 변환을 통하여 1차원 DCT의 합으로 변환하여 계산하는 알고리즘을 개발한다. 기존의 2차원 계산방법인 row-column 으로는 $N\times M$ 크기의 2D-DCT에서 $3/2NM\log_2(NM)-2NM+N+M$의 합과 $1/2NM\log_2(NM)$의 곱셈이 필요한데 비하여 본 논문에서 제시한 알고리즘은 $3/2\,NM\log_2 M\ +NM\log_2 N\ -M-N/2+2$의 합과 $1/2NM\log_2 M$의 곱셈 수를 필요로 한다. 또한 기존의 polynomial 변환에 의한 2D DCT는 Euler 공식을 적용하였기 때문에 복소 연산이 필요하지만 본 논문에서 제시한 polynomial 변환은 DCT의 modular 규칙을 이용하여 2D DCT를 1D DCT의 합으로 직접 변환하므로 복소 연산이 필요하지 않다.

Abstract

This paper develops a novel algorithm of computing 2 Dimensional Discrete Cosine Transform (2D-DCT) via Polynomial Transform (PT) converting 2D-DCT to the sum of 1D-DCTs. In computing $N\times M$ size 2D-DCT, the conventional row-column algorithm needs $3/2NM\log_2(NM)-2NM+N+M$ additions and $1/2NM\log_2(NM)$ multiplications, while the proposed algorithm needs $3/2\,NM\log_2 M\ +NM\log_2 N-M-N/2+2$ additions and $1/2NM\log_2 M$ multiplications The previous polynomial transform needs complex operations because it applies the Euler equation to DCT. Since the suggested algorithm exploits the modular regularity embedded in DCT and directly decomposes 2D DCT into the sum of 1D DCTs, the suggested algorithm does not require any complex operations.

Keywords : 2D-DCT, Polynomial Transform, Fast algorithm

* 正會員, 明知大學校 情報通信工學科
(Myongji University, Electronics, Information & Communication Engineering)
** 正會員, 慶熙大學校 電子工學科
(Kyung Hee University, School of Electronics and Information, Engineering)

※ 이 논문은 2001년도 한국학술진흥재단의 지원에 의하여 연구 되었음(KRF-2001-003-E00158).
接受日字:2003年3月31日, 수정완료일:2003年10月20日

I. Introduction

2D-DCT is one of the most fundamental tools in image processing, owing to DCT's capability of producing excellent spatial frequency of images[1]. Since, the amount of 2-D data is square of 1-D data size, an algorithm for a fast 2D-DCT should reduce the complexity of computing DCT itself as well as the number of two-dimensional calculations. The

polynomial transform(PT) based algorithms were reported to offer more savings on the required number of operations for two-dimensional DCT in [2-4]. Compared with the row-column method[1], Polynomial transforms (PT) require only one half of the number of multiplications and a smaller number of additions. Therefore, the polynomial transform have the greatest potential toward lowest arithmetic complexity.

The initial use of polynomial transform for a 2 dimensional calculation was made by Nussbaumer et al[5, 6] so as to compute 2D cyclic convolutions. Their approach can be considered as a generalization of 2D Fourier transform in view of polynomial theory. The approaches of applying the polynomial transform for computing 2D-DCT were followed by Guillemot *et al*, Feig *et al*, Cho et al and so on ([3, 4, 7]). All these algorithms use the Euler equation and the symmetry of DCT kernels and transform the DCT to a z-transform polynomial. The algorithms then apply a polynomial transform to convert 2D-DCT to the sum of 1D-DCTs. Although the algorithms reduce the half of multiplications compared to the row-column computation, they require the complex calculation due to the Euler equation. In addition, the previous algorithms treat only the same size dimensions which implies that the algorithms may not support applications that require different dimensional sizes.

In this paper, we present a novel 2D fast Discrete Cosine Transform using polynomial transform free of complex number computations. Fig. 1 shows the diagram for overall 2D-DCT computation processes via polynomial transform. Instead of using the Euler equation, we exploit the modular regularity embedded in DCT in order to directly decompose 2D-DCT into 1D-DCT. The conventional fast algorithms of 2D-DCT process have no relations between its row and column. These independent processes can be related with each row and column by module operation, and then row (or column) index is subordinated to column (or row) index. The dependency of row subordinated on column, means that row indexes have the overlapped factors of column. The overlap factors are the twiddle factors for column index and those common factors can be compounded into one dimensional factor. This representation of 2D-DCT makes the series of polynomial and transform for row is acquired in the process of transform for column at a time. Then this polynomial representation has the symmetry, periodicity and fast polynomial transform algorithm. Therefore, the suggested polynomial transform does not require the complex computations. Also, the suggested polynomial transform does not assume the same dimensional sizes so that the presented 2D-DCT can be applicable to $N \times M$ image data where $N=2^n$, $M=2^m$ for integer n,m.

The remainder of this paper is organized as follows. Section II describes the 2D-DCT obtained by the rearrangement of input signal on spatial domain. In Section III, 2D-DCT is represented on polynomial domain, which induces the suggested polynomial transform. In Section IV, the fast algorithm is derived from the polynomial transforms presented in Section III. In this section, we will explain the fast algorithm by showing a certain symmetry in the suggested polynomial transform. Finally, we reach the conclusion in Section V.

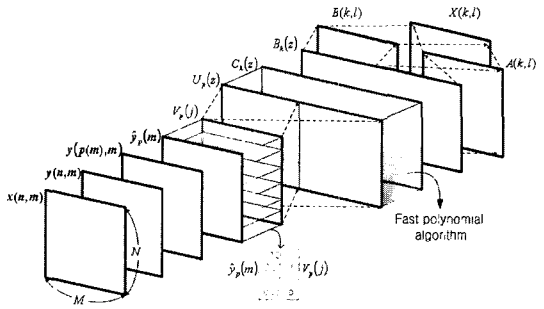


그림 1. 다항식변환을 이용한 2D-FDCT-II 과정의 전체 개념도

Fig. 1. Abstract diagram of 2D-FDCT-II via polynomial transform.

II. Decomposition of 2D-DCT into 1D-DCTs

Generally, if the 2 dimensional computations is converted to sums of one dimensional computation, the multiplications of two dimensional computations is reduced almost by half. 2D DCT can be converted to sums of 1D-DCT by converting the production of DCT kernels in 2D DCT to a Cosinse through the trigonometric formulae. For the converted form to construct the 1D-DCT, the Cosine derived by the trigonometric formulae must sustain the DCT kernel. In order for the cosine to sustain the DCT kernel, the input matrix needs to be re-arranged while deriving a cosine through the trigonometric formulae. In this section, we explain the re-ordering procedure for representing 2D-DCT to sums of 1D-DCTs.

The 2D-DCT-II of the input sequences $x(n, m)$ is defined by

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

The constant scaling factors in the DCT definition are ignored for simplicity. We assume that M and N are powers of 2 and $M \geq N$. We can write $N = 2^t$ and $M = 2^J N$, where $t > 0$ and $J \geq 0$, respectively.

In (1), the two dimensional kernel production $\cos \frac{\pi(2n+1)k}{2N} \cdot \cos \frac{\pi(2m+1)l}{2M}$ can be converted to a cosine through the trigonometric formulae. However, since the phase of the cosine is expressed with 2-dimensional indices, n, m, the cosine does not form the DCT kernel. In order for the converted cosine to form the DCT kernel, we develop a module operation that transforms the 2-dimensional indices into a one-dimensional DCT. Before applying the module operation, we also need a index re-ordering process preventing the phase inversion occurred during the module operation. The following explains the index reordering process that is performed by changing post/pre order to even/odd order.

$x(n, m)$ can be decomposed into re-ordered matrix $y(n, m)$ such as,

$$\begin{aligned} y(n, m) &= x(2n, 2m) \\ y(N-1-n, m) &= x(2n+1, 2m) \\ y(n, M-1-m) &= x(2n, 2m+1) \\ y(N-1-n, M-1-m) &= x(2n+1, 2m+1) \end{aligned} \quad (2)$$

where

$n = 0, 1, \cdots, N/2-1$
$m = 0, 1, \cdots, M/2-1.$

Then

$$X(k, l) = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} y(n.m) \cos \frac{\pi(4n+1)k}{2N} \cdot \cos \frac{\pi(4m+1)l}{2M} \quad (3)$$
$$k = 0, 1, \cdots, N-1; \quad l = 0, 1, \cdots, M-1.$$

Fig2 shows index re-ordering of $x(n, m)$ for N=4, M=4.

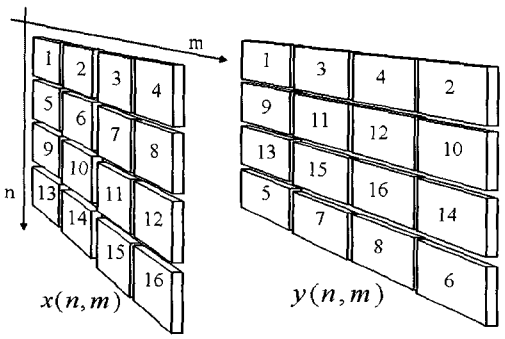


그림 2. 인덱스에 따른 재배열
Fig. 2. Matrix re-ordering by index.

Using the trigonometric formulae, (3) can be computed by $X(k, l) = \frac{1}{2}[A(k, l) + B(k, l)]$,

where

$$A(k, l) = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} y(n, m)$$

$$\cdot \cos\left[\frac{\pi(4n+1)k}{2N}+\frac{\pi(4m+1)l}{2M}\right] \quad (4)$$

$k=0,1,\cdots,N-1;\quad l=0,1,\cdots,M-1$

and

$$B(k,l)=\sum_{n=0}^{N-1}\sum_{m=0}^{M-1}y(n,m)$$
$$\cdot \cos\left[\frac{\pi(4n+1)k}{2N}-\frac{\pi(4m+1)l}{2M}\right] \quad (5)$$

$k=0,1,\cdots,N-1;\quad l=0,1,\cdots,M-1.$

In (4) and (5), since the cosine has the two dimensional indices, n,m, A(k,l) and B(k,l) do not sustain the 1D-DCT. So, if we let the cosine be the DCT kernel, A(k,l) and B(k,l) also becomes the sums of 1D-DCTs. In the following, we develop a module operation that transforms the two dimensional indices to a one-dimensional index so as for the cosine to sustain DCT kernel.

Consider $(A\times B)$ mod N for two different number A, B. If one of the two number A is coprime with N, the range values of B, that is, 0,⋯,N-1, one-to-one corresponds to those of $(A\times B)$ mod N [ref xxx]. From this observation, we develop a module operation letting the index n be expressed in (4m+1) module form. This module operation transforms two indices n, m into an one-dimensional index.

Now, we define $p(m)$ as $p(m)=[(4p+1)m+p] \bmod N$, then

$$4p(m)+1\equiv(4m+1)(4p+1) \bmod 4N$$
$$where\ p=0,1,\cdots,N-1;\ m=0,1,\cdots,M-1.$$

Through the following lemma 1, $p(m)$ can also replace n in more different order with n.

Lemma 1 :

$$A=\{(n,m)|0\le n\le N-1;\quad 0\le m\le M-1\}$$
$$B=\{(p(m),m)|0\le p\le N-1;\quad 0\le m\le M-1\}$$

Then A=B.

proof : Let $[p(m),m]$ *and* $[p'(m'),m']$ *be two elements in B. If they are equal, then*

$$p(m)=p'(m'),\ m=m'.$$

From the definition of $p(m)$, we have

$$(4p+1)m+p\equiv(4p'+1)m+p' \bmod N.$$

Hence

$$(4m+1)(p-p')\equiv 0 \bmod N.$$

Since $4m+1$ and N are co-prime with each other, we have $p\equiv p' \bmod N$. Therefore, $p=p'$, which concludes that the elements in B are different from each other and n can be replaced via $p(m)$. ∎

Fig 3 shows module re-ordering.

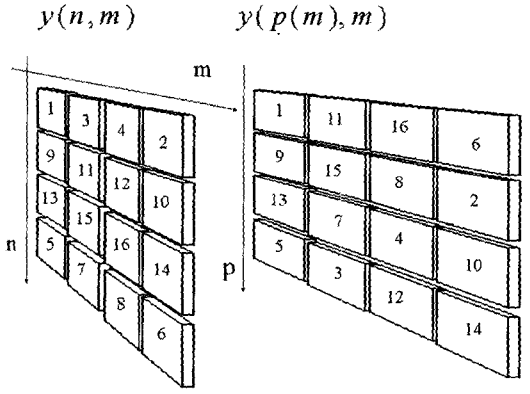


그림 3. module 연산 $p(m)$에 따른 재배열
Fig. 3. Matrix re-ordering by module operation $p(m)$.

By plugging $p(m)$ into (4) and (5), we obtain (6) and (7) as bottom.

$$A(k,l)=\sum_{p=0}^{N-1}\sum_{m=0}^{M-1}y(p(m),m)\cos\left[\frac{\pi(4p(m)+1)k}{2N}+\frac{\pi(4m+1)l}{2M}\right] \quad (6)$$
$$=\sum_{p=0}^{N-1}\sum_{m=0}^{M-1}y(p(m),m)\cos\left[\frac{\pi(4m+1)(4p+1)k}{2N}+\frac{\pi(4m+1)l}{2M}\right]=\sum_{p=0}^{N-1}\sum_{m=0}^{M-1}y(p(m),m)\cdot$$
$$\cos\left[\frac{\pi(4m+1)(2^{l}(4p+1)k+l)}{2M}\right].$$

$$B(k,l) = \sum_{p=0}^{N-1}\sum_{m=0}^{M-1} y(p(m),m)\cos\left[\frac{\pi(4p(m)+1)k}{2N} - \frac{\pi(4m+1)l}{2M}\right] \quad (7)$$

$$= \sum_{p=0}^{N-1}\sum_{m=0}^{M-1} y(p(m),m)\cos\left[\frac{\pi(4m+1)(4p+1)k}{2N} - \frac{\pi(4m+1)l}{2M}\right] = \sum_{p=0}^{N-1}\sum_{m=0}^{M-1} y(p(m),m)\cdot \cos\left[\frac{\pi(4m+1)(2^J(4p+1)k-l)}{2M}\right].$$

In (6) and (7), we can define the common factor of cosine such as

$$V_p(j) = \sum_{m=0}^{M-1} y(p(m),m)\cos\left[\frac{\pi(4m+1)j}{2M}\right] \quad (8)$$

for $p=0,1,\cdots,N-1$; $j=0,1,\cdots,M-1$.

Through re-ordering index of $y(p(m),m)$, $V_p(j)$ in (8) becomes 1D-DCT. To re-ordering index of $y(p(m),m)$, we define $\hat{y}_p(m)$ such as

$$\begin{cases} \hat{y}_p(2m) = y(p(m),m) \\ \hat{y}_p(2m+1) = y(p(M-1-m),M-1-m) \end{cases}$$

where $m=0,1,\cdots,M/2-1$ (15)

Fig.4 shows re-ordering of $y(p(m),m)$ to $\hat{y}_p(m)$.

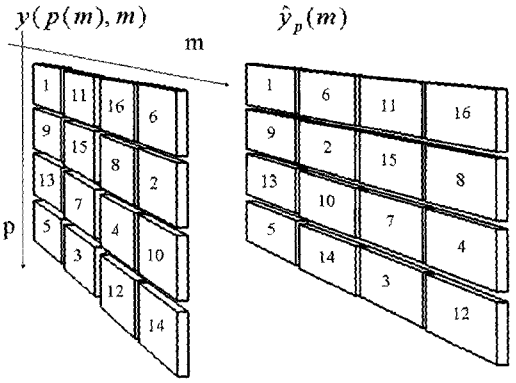


그림 4. $y(p(m),m)$에 대한 재배열
Fig. 4. the re-ordering of $y(p(m),m)$.

This reordering is similar to index re-ordering of (2). Now, $V_p(j)$ can be expressed into one dimensional DCT for each p.

$$V_p(j) = \sum_{m=0}^{M-1} \hat{y}_p(m)\cos\left[\frac{\pi(2m+1)j}{2M}\right] \quad (10)$$

where $p=0,1,\cdots,N-1; j=0,1,\cdots,M-1$.

Fig. 5 shows the diagram of Eq.(9) for each p in rows.

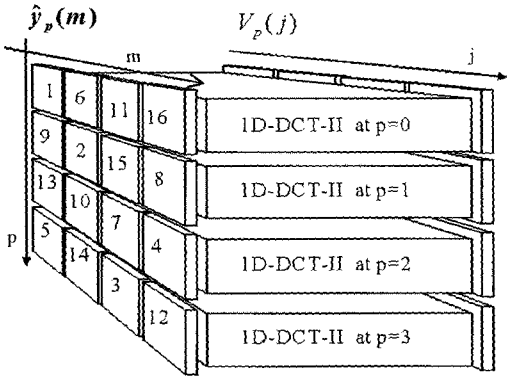


그림 5. 각 행에 대한 1D-DCT들
Fig. 5. 1D-DCTs for each row.

$A(k,l)$ and $B(k,l)$ can also be factored with $V_p(j)$ as follows

$$A(k,l) = \sum_{p=0}^{N-1} V_p[2^J(4p+1)k+l] \quad (11)$$

$$B(k,l) = \sum_{p=0}^{N-1} V_p[2^J(4p+1)k-l] \quad (12)$$

for $k=0,1,\cdots,N-1$; $l=0,1,\cdots,M-1$.

In order that 2D-DCT X(k,l) becomes the sums of 1D-DCTs, $A(k,l)$ and $B(k,l)$ in (10) and (11) should be the sum of $V_p(j)$ for p . This indicates that $V_p(2^J(4p+1)k-l)$ in (10) and (11) must be converted to $V_p(j)$. The following section explains the procedure converting $V_p(2^J(4p+1)k-l)$ to $V_p(j)$ by polynomial transform.

III. Polynomial Representation of 2D-DCT

As mentioned in previous section, in order for $A(k,l)$ and $B(k,l)$ to be the sums of 1D-DCT,

$V_p(2^J(4p+1)k-l)$ must be transformed to $V_p(j)$. Transforming the index $2^J(4p+1)k-l$ to j implies re-ordering $V_p(2^J(4p+1)k-l)$ to $V_p(j)$. This reordering makes use of the periodic properties of $V_p(j)$. The reordering process can be easily developed by a polynomial's module operation. Thus, in this section, we develop a polynomial to make use of the polynomial's module operations. The polynomial's coefficients are $V_p(j)$ and the order of the polynomial's terms indicates where $V_p(j)$ is on the transform domain.

Both $A(k,l)$ and $B(k,l)$ contains $V_p(\cdot)$, and $V_p(\cdot)$ is periodic. So, $A(k,l)$ and $B(k,l)$ can be related through the periodic properties of $V_p(j)$. Once $A(k,l)$ and $B(k,l)$ are related, a polynomial calculating $A(k,l)$ and $B(k,l)$ simultaneously can be derived.

The periodic properties of $V_p(j)$ are

$$\begin{aligned} &V_p(j+u2M)=(-1)^u V_p(j)\\ &V_p(2M-j)=-V_p(j), \text{ and}\\ &V_p(M)=0, \qquad \textit{where}\quad j=0,1,\cdots,M-1 \end{aligned} \tag{13}$$

Based on the periodic properties of $V_p(j)$,

$$\begin{aligned} A(k,2M-l) &= \sum_{p=0}^{N-1} V_p[2^J(4p+1)k+2M-l]\\ &= -\sum_{p=0}^{N-1} V_p[2^J(4p+1)k-l]\\ &= -B(k,l) \end{aligned} \tag{14}$$

and, $A(k,0)=B(k,0), \qquad A(0,l)=B(0,l).$ (15)

In order to transform the index $2^J(4p+1)k-l$ to j, $V_p(2^J(4p+1)k-l)$ must be reordered. The reordering process can be easily developed by a polynomial's module operation. Thus, we define a polynomial $B_k(z)$ with respect to column index l, which permits the module operation, such as

$$B_k(z)=\sum_{l=0}^{M-1} B(k,l)z^l-\sum_{l=M}^{2M-1} A(k,2M-l)z^l. \tag{16}$$

By substituting (10) and (14) in (16),

$$B_k(z)=\sum_{l=0}^{2M-1}\sum_{p=0}^{N-1} V_p(2^J(4p+1)k-l)z^l. \tag{17}$$

In (17), if $V_p(2^J(4p+1)k-l)$ is $V_p(j)$, $B_k(z)$ is the sum of 1D-DCT. For that reason, to exchange the index of $V_p(\cdot)$, $2^J(4p+1)k-l$ into j, we have to re-ordering $V_p(\cdot)$. This re-ordering process can be explained by the periodic property of polynomial module operation. The following theorem justifies the periodic properties in polynomial module operations.

In (18), if $V_p(2^J(4p+1)k-l)$ is transformed to $V_p(j)$, $B_k(z)$ constructs the sums of 1D-DCT. The following theorem justifies the periodic properties in polynomial module operations.

Theorem 1:

If g_j *is a coefficient of* z^j *in a polynomial* $g(z)=\sum_{j=0}^{N-1} g_j\cdot z^j$, *then* $(g(z)\cdot z^L) \bmod (z^N+1) \equiv \sum_{j=0}^{N-1}(-1)^{\left\lfloor \frac{j+L}{N} \right\rfloor} g_j\cdot z^{(j+L) \bmod N}$ *,where an integer* L *is* $0\le L\le N-1$.

proof: Since $g_j\cdot z^j = g_j z^{j-l}(z^l+1)-g_j z^{j-l}$,

$$(g_j\cdot z^j) \bmod (z^l+1) \equiv \begin{cases} -g_j z^{j-l} & \textit{where } j\ge l\\ g_i\cdot z^j & \textit{where } j<l \end{cases} \tag{18}$$

For an integer L that is $0\le L\le N-1$,

$$\begin{aligned} &(g(z)\cdot z^L) \bmod (z^N+1)\\ &\equiv \left(\sum_{j=0}^{N-1} g_j\cdot z^{j+L}\right) \bmod (z^N+1)\\ &\equiv \sum_{j=0}^{N-1}\{(g_j\cdot z^{j+L}) \bmod (z^N+1)\}\\ &\equiv \sum_{j=0}^{N-L-1}\{(g_j\cdot z^{j+L}) \bmod (z^N+1)\}\\ &\quad + \sum_{j=N-L}^{N-1}\{(g_j\cdot z^{j+L}) \bmod (z^N+1)\} \end{aligned}$$

from (18)

$$\equiv \sum_{j=0}^{N-L-1} g_j\cdot z^{j+L}+\sum_{j=N-L}^{N-1} -g_j\cdot z^{j+L-N}$$

$$= \sum_{j=0}^{L-1} -g_{j-L+N} \cdot z^j + \sum_{i=L}^{N-1} g_{j-L} \cdot z^j$$

$$= -g_{N-L} z^0 - g_{N-L+1} z^1 - \cdots - g_{N-1} z^{L-1} + g_0 z^L + g_1 z^{L+1} + \cdots + g_{N-L-1} z^{N-1}$$

$$= \sum_{j=0}^{N-1} (-1)^{\left\lfloor \frac{j+L}{N} \right\rfloor} g_j \cdot z^{(j+L) \bmod N}$$

This proves the theorem 1. ∎

Using Theorem 1, we derive that $B_k(z)$ becomes a module operation and then turns out to be the sum of $V_p(j)$.

$$B_k(z) = \sum_{l=0}^{2M-1} \sum_{p=0}^{N-1} V_p(2^J(4p+1)k - l) z^l$$

$$= \sum_{p=0}^{N-1} \sum_{l=0}^{2M-1} V_p(2^J(4p+1)k - l) z^l$$

Since $l < 2M$,

$$= \sum_{p=0}^{N-1} \sum_{l=0}^{2M-1} (-1)^{\left\lfloor \frac{l}{2M} \right\rfloor} V_p(2^J(4p+1)k - l) z^{l \bmod 2M}$$

Since $V_p(j)$ is 2M-periodic with respect to j, $V_p(j) = V_p(j \bmod 2M)$. Let $j = (l-L) \bmod 2M$, where $L = 2^J(4p+1)k$. Then $V_p(l-L) = V_p((l-L) \bmod 2M) = V_p(j)$. By definition of j, the index set $\{l\} = \{0, 1, \cdots, 2M-1\}$ one-to-one maps to the index set $\{j\} = \{2M-L, 2M-L+1, \cdots, 0, 1, \cdots, 2M-L-1\}$. $l = (j+L) \bmod 2M$ is also trivial. From these observations, we can change $B_k(z)$ in module form such as

$$B_k(z) = \sum_{p=0}^{N-1} \sum_{j=0}^{2M-1} (-1)^{\left\lfloor \frac{(2^J(4p+1)k + j) \bmod 2M}{2M} \right\rfloor} \cdot V_p(j) z^{(j + 2^J(4p+1)k) \bmod 2M}$$

by Theorem 1,

$$\equiv \sum_{p=0}^{N-1} \sum_{j=0}^{2M-1} V_p(j) z^{j + 2^J(4p+1)k} \bmod (z^{2M}+1). \quad (19)$$

In computing Eq.(19), step-by-step approach is much easier than direct computation. For this purpose, we factorize $B_k(z)$ as following;

$$B_k(z) \equiv \left[\sum_{p=0}^{N-1} U_p(z) \hat{z}^{pk} \right] z^{2^J k} \bmod (z^{2M}+1)$$

$$\equiv C_k(z) z^{2^J k} \bmod (z^{2M}+1), \quad (20)$$

where

$$U_p(z) \equiv \sum_{j=0}^{2M-1} V_p(j) z^j \bmod (z^{2M}+1)$$

$$= \sum_{j=0}^{2M-1} V_p(j) z^j \quad (21)$$

$$C_k(z) \equiv \sum_{p=0}^{N-1} U_p(z) \hat{z}^{pk} \bmod (z^{2M}+1) \quad (22)$$

$k = 0, 1, \cdots, N-1$; $\hat{z} = z^{2^{J+2}} \bmod (z^{2M}+1)$.

$U_p(z)$ is the polynomial whose coefficients are 1D-DCT values of the pth row. The jth term's coefficient is equivalent to the value of jth DCT. For actual computations, $V_p(j)$ only for $j = 0 \sim M-1$ are calculated and $V_p(j)$ for $j = M \sim 2M-1$ use the calculated values from the properties of $V_p(j)$, Fig. 6 shows the procedure of generating $U_p(z)$ from $V_p(j)$ in eq.(21).

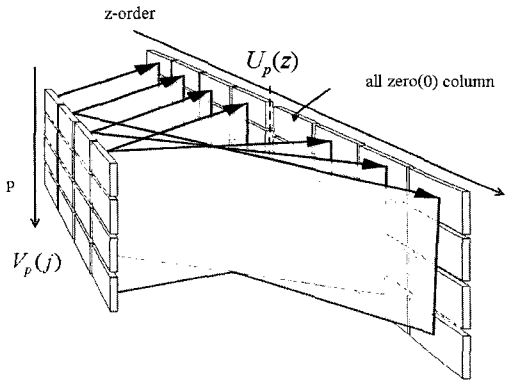


그림 6. $U_p(z)$: $V_p(j)$에 의한 다항식
Fig. 6. $U_p(z)$: the polynomial via $V_p(j)$.

The coefficients of $C_k(z)$ are the row directional sums of $U_p(z)$'s coefficients which are reordered along each column direction. The orders of z represent the order of reordering on the row. The coefficients of $C_k(z)$ are 2D-DCT values of the reordered 2D matrix. That is, 2D-DCT values of the

original input 2D matrix are obtained by reordering the $C_k(z)$'s coefficients to spatial domain. Furthermore, since $C_k(z)$ forms the polynomial transform, a fast alogrithm for calculating $C_k(z)$ is available.

Fig.7 shows the process of polynomial transform in Eq.(22). In Fig.7, $C_0(z)$ is the sum of 4 polynomials, $U_p(z)$, and each polynomial has 8 coefficients. So, a polynomial $C_0(z)$ is computed by 8×3 additions. Therefore, $C_k(z)$, the polynomial transform for $U_p(z)$ (p=0.1.2.3) needs 4×8×3 additions. In general, polynomial transform requires $N \cdot 2M \cdot (N-1)$ additions. But the number of additions are reduced by fast algorithm, which will be explained in next section.

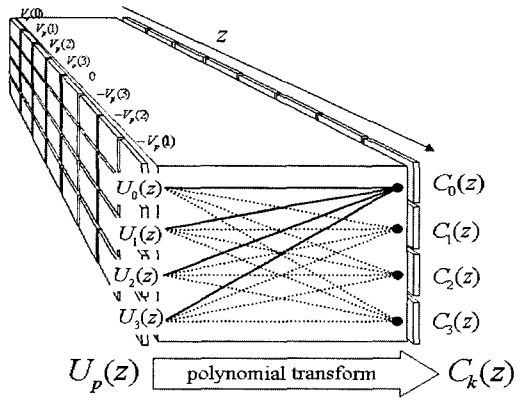


그림 7. 다항식 변환
Fig. 7. the Polynomial Transform : $U_p(z) \xrightarrow{P.T.} C_k(z)$.

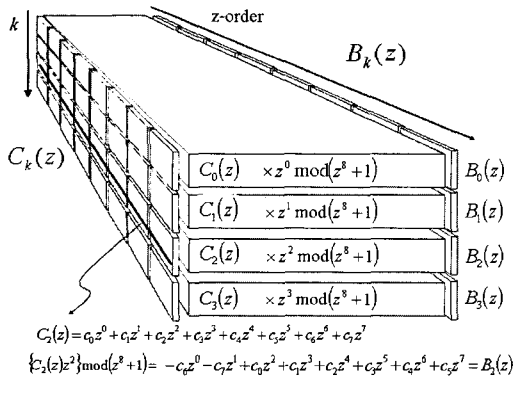


그림 8. M=4일 때, $C_k(z)$에 의해 구하여지는 $B_k(z)$
Fig. 8. $B_k(z)$ driven from $C_k(z)$ in case of $M=4$.

Now, we have got the polynomial transformed value, $C_k(z)$. And, $B_k(z)$ is computed from $C_k(z)$ from Eq.(20). Fig. 8 shows the procedure of Eq.(20). We re-order $C_k(z)$ by polynomial module operation. Each k-th row array means k-th polynomial $C_k(z)$. And, polynomial module operations are only re-mapping of coefficients of polynomial $C_k(z)$.

The coefficients of $B_k(z)$ are $A(k,l)$ and $B(k,l)$ from Eq.(16),(17). In Eq.(16), $B(k,M)$ is not required by our computation, because we need $A(k,l)$ and $B(k,l)$ only for $l=0,\cdots,M-1$. So, $N/2-1$ additions are saved in computation of polynomial transform, Eq.(22). Fig.9 shows the procedure of Eq.(16) in case of M=N=4.

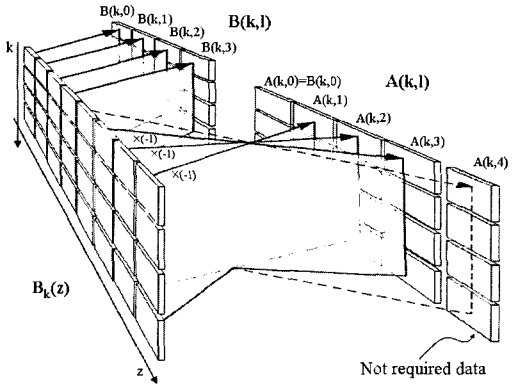


그림 9. 다항식 $B_k(z)$에 의해 구하여지는 $A(k,l)$와 $B(k,l)$
Fig. 9. The derivation of $B(k,l)$ and $A(k,l)$ from polynomial $B_k(z)$.

From Eq.(4),(5), $X(k,l)$ is the sum of $A(k,l)$ and $B(k,l)$, and the sum requires $NM-N-M+1$ additions when the properties in Eq.(15) are exploited. Consequently, 2D-DCT for $x(n,m)$ is completed by getting $X(k,l)$.

Ⅳ. Fast Polynomial Transform

In this section, we develop the fast algorithm for evaluating $C_k(z)$. This fast polynomial transform can be derived via the symmetry property of z.

This means that $C_k(z)$ can be evaluated by the fast algorithm which is similar to the Fast Fourier transform (FFT).

Define $(H(z)) \bmod (z^{2M}+1) \triangleq (H(z))_{\langle 2M \rangle}$. For example, $(z^i)_{\langle 2M \rangle} = (z^i) \bmod (z^{2M}+1)$.

The following lemma 3 indicates that $\hat{z}$ is the twiddle factor for the period of N.

Lemma 3 : $\hat{z}$ satisfies symmetry property for the period of N as following;

$$(\hat{z}^{i+N})_{\langle 2M \rangle} = \hat{z}^i, \quad (\hat{z}^{i+\frac{N}{2}})_{\langle 2M \rangle} = -\hat{z}^i,$$

where $i = 0, \cdots, 2M-1$, $M = 2^J N$.

proof :

From the previous definition of $\hat{z} = (z^{2^{J+2}})_{\langle 2M \rangle}$ in eq.(22), generally,

In general,

$$\begin{aligned}
&\hat{z}^L \bmod (z^{2M}+1) \\
&= (z^{2^{J+2}} \bmod (z^{2M}+1))^L \bmod (z^{2M}+1) \\
&= (z^{2^{J+2}})^L \bmod (z^{2M}+1) \\
&= \begin{cases} z^{4M} \bmod (z^{2M}+1) = 1, & L = N \\ z^{4M} \bmod (z^{2M}+1) = -1, & L = N/2, \end{cases}
\end{aligned}$$

Thus

$$\begin{aligned}
&(\hat{z}^{i+N})_{\langle 2M \rangle} = \hat{z}^{i+N} \bmod (z^{2M}+1) \\
&= (\hat{z}^i \bmod (z^{2M}+1) \cdot \hat{z} \bmod (z^{2M}+1)) \bmod (2^{2M}+1) \\
&= (\hat{z}^i \bmod (z^{2M}+1) \cdot 1) \bmod (z^{2M}+1) \\
&= \hat{z}^i \bmod (z^{2M}+1) = \hat{z}^i.
\end{aligned}$$

In similar way,

$$\begin{aligned}
&(\hat{z}^{i+N/2})_{\langle 2M \rangle} = \hat{z}^{i+N/2} \bmod (z^{2M}+1) \\
&= ((\hat{z}^i \bmod (z^{2M}+1)) \cdot -1) \bmod (2^{2M}+1) = -\hat{z}^i. \ \blacksquare
\end{aligned}$$

Considering that $\hat{z}$ is compared to the twiddle factor $W_N \left(= e^{-j\frac{2\pi}{N}}\right)$ in DFT, we can drive a fast polynomial transform algorithm in similar way to FFT. The fast polynomial transform computes a polynomial transform by $\log_2 N$ stages. The computation of each stage constructs recursive computations based on symmetric property of Lemma 3.

Express p and k with the binary expression such as

$$\begin{aligned}
p &= p_{t-1}2^{t-1} + \cdots + p_{t-r}2^{t-r} + \cdots + p_0 2^0 \\
&\triangleq p_{t-1}p_{t-2}\cdots p_0 \\
k &= k_{t-1}2^{t-1} + \cdots + k_{t-r}2^{t-r} + \cdots + k_0 2^0 \\
&\triangleq k_{t-1}k_{t-2}\cdots k_0
\end{aligned}$$

where $p_{r-1} \in \{0,1\}$, $k_{r-1} \in \{0,1\}$ for $r = 1, 2, \cdots, t(= \log_2 N)$. Then,

$$\begin{aligned}
C_k(z) &= C_{k_{t-1}k_{t-2}\cdots k_0}(z) \qquad (23) \\
&\equiv \Big(\sum_{p_0=0}^{1} \cdots \sum_{p_{t-1}=0}^{1} U_{p_{t-1}p_{t-2}\cdots p_0}(z) \\
&\quad \cdot \hat{z}^{(p_{t-1}2^{t-1}k_0 + \cdots + p_1 2^1 (2^{t-1}k_{t-1} + \cdots + 2^0 k_0))} \Big)_{\langle 2M \rangle} \\
&= \Big\{ \Big(\sum_{p_1=0}^{1} \cdots \sum_{p_{t-1}=0}^{1} U_{p_{t-1}\cdots p_1 0}(z) \cdot \hat{z}^{(p_{t-1}2^{t-1}k_0 + \cdots + p_1 2^1 (2^{t-1}k_{t-1} + \cdots + 2^0 k_0))} \Big)_{\langle 2M \rangle} + \\
&\quad \Big(\sum_{p_1=0}^{1} \cdots \sum_{p_{t-1}=0}^{1} U_{p_{t-1}\cdots p_1 1}(z) \cdot \hat{z}^{(p_{t-1}2^{t-1}k_0 + \cdots + p_1 2^1 (2^{t-2}k_{t-2} + \cdots + 2^0 k_1))} \\
&\quad \cdot \hat{z}^{(2^{t-1}k_{t-1} + \cdots + 2^0 k_0))} \Big)_{\langle 2M \rangle} \Big\}_{\langle 2M \rangle}
\end{aligned}$$

We can separate even ($p_o = 0$) and odd ($p_o = 1$) terms with respect to p in following way.

$$\begin{aligned}
C^0_{k_{t-1}\cdots k_0}(z) &\equiv \Big(\sum_{p_1=0}^{1} \cdots \sum_{p_{t-1}=0}^{1} U_{p_{t-1}\cdots p_1 0}(z) \\
&\quad \cdot \hat{z}^{(p_{t-1}2^{t-1}k_0 + \cdots + p_1 2^1 (2^{t-2}k_{t-2} + \cdots + 2^0 k_0))} \Big)_{\langle 2M \rangle} \\
&\text{for } p_o = 0 \qquad (24) \\
C^1_{k_{t-1}\cdots k_0}(z) &\equiv \Big(\sum_{p_1=0}^{1} \cdots \sum_{p_{t-1}=0}^{1} U_{p_{t-1}\cdots p_1 1}(z) \\
&\quad \cdot \hat{z}^{(p_{t-1}2^{t-1}k_0 + \cdots + p_1 2^1 (2^{t-2}k_{t-2} + \cdots + 2^0 k_0))} \Big)_{\langle 2M \rangle}. \\
&\text{for } p_o = 1 \qquad (25)
\end{aligned}$$

Then,

$$\begin{aligned}
C_{k_{t-1}k_{t-2}\cdots k_0}(z) &\equiv \Big(C^0_{k_{t-2}\cdots k_0}(z) \\
&\quad + \hat{z}^{2^{t-1}k_{t-1}} \cdot \hat{z}^{(2^{t-2}k_{t-2} + \cdots + 2^0 k_0)} \cdot \\
&\quad C^1_{k_{t-2}\cdots k_0}(z) \Big)_{\langle 2M \rangle}
\end{aligned}$$

from Lemma 3,

$$\equiv\Big(C^{0}_{k_{t-2}\cdots k_0}(z)+(-1)^{k_{t-1}}\cdot \hat{z}^{(2^{t-2}k_{t-2}+\cdots+2^0k_0)}\cdot C^{1}_{k_{t-2}\cdots k_0}(z)\Big)_{\langle 2M\rangle}.$$

Eq.(24),(25) are the polynomial transforms for $U_p(z)$ of even p ($p_o=0$) and odd p ($p_o=1$), respectively. Recursively,

$$C^{0}_{k_{t-2}\cdots k_t}(z)=\Big(C^{00}_{k_{t-3}\cdots k_0}(z)+(-1)^{k_{t-2}}\cdot \hat{z}^{2(2^{t-3}k_{t-3}+\cdots+2^0k_0)}\cdot C^{01}_{k_{t-3}\cdots k_0}(z)\Big)_{\langle 2M\rangle},$$

and

$$C^{1}_{k_{t-2}\cdots k_t}(z)=\Big(C^{10}_{k_{t-3}\cdots k_0}(z)+(-1)^{k_{t-2}}\cdot \hat{z}^{2(2^{t-3}k_{t-3}+\cdots+2^0k_0)}\cdot C^{11}_{k_{t-3}\cdots k_0}(z)\Big)_{\langle 2M\rangle}$$

In this recursive structure, we define a polynomial transform such as

$$C^{p_{t-r-1}\cdots p_0}_{k_{r-1}\cdots k_0}(z)=\begin{cases} U_{p_{t-1}\cdots p_0}(z) & \text{for } r=0\\ \Big(\sum^{1}_{p_r=0}\cdots\sum^{1}_{p_{t-1}=0} U_{p_{t-1}\cdots p_0}(z)\cdot \hat{z}^{(p_{t-1}2^{t-1}k_t+\cdots+p_{t-r}2^{r}(2^{r-1}k_{r-1}+\cdots+k_0))}\Big)_{\langle 2M\rangle} & \text{for } r=1,\cdots,t-1,\\ C_{k_{t-1}\cdots k_0}(z) & \text{for } r=t\end{cases}$$

For convenience, we define the circular shifting factor q_r at the r th stage such as

$$q_r \triangleq \begin{cases} 0 & \text{for } r=1\\ \sum_{l=0}^{r-2} k_l 2^{t-r+l} & \text{for } r=2,3,\cdots,t\end{cases}$$

Then, we can express the general recursive structure at r $(=1,2,\cdots,t)$ stage in following way

$$C^{p_{t-r-1}\cdots p_0}_{k_{r-1}\cdots k_0}(z)\equiv\Big(C^{0p_{t-r-1}\cdots p_0}_{k_{r-2}\cdots k_0}(z)+(-1)^{k_{r-1}}\cdot \hat{z}^{q_r}\cdot C^{1p_{t-r-1}\cdots p_0}_{k_{r-2}\cdots k_0}(z)\Big)_{\langle 2M\rangle}$$

$$\equiv\begin{cases}\Big(C^{0p_{t-r-1}\cdots p_0}_{k_{r-2}\cdots k_0}(z)+\hat{z}^{q_r}\cdot C^{1p_{t-r-1}\cdots p_0}_{k_{r-2}\cdots k_0}(z)\Big)_{\langle 2M\rangle} & \text{for } k_{r-1}=0\\ \Big(C^{0p_{t-r-1}\cdots p_0}_{k_{r-2}\cdots k_0}(z)-\hat{z}^{q_r}\cdot C^{1p_{t-r-1}\cdots p_0}_{k_{r-2}\cdots k_0}(z)\Big)_{\langle 2M\rangle} & \text{for } k_{r-1}=1\end{cases} \quad (26)$$

The computational structure of $C^{p_{t-r-1}\cdots p_0}_{k_{r-1}\cdots k_0}(z)$ in Eq.(26) constructs the butterfly operation shown in Fig.10. This implies that the fast computation of $C_k(z)$ forms the same recursive structure as that of FFT.

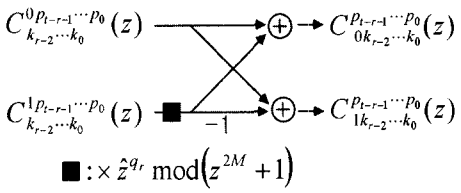


그림 10. r 번째 단계에서 Butterfly 연산
Fig. 10. Butterfly operation at the rth stage.

Also, the polynomial transform $C^{p_{t-r-1}\cdots p_0}_{k_{r-1}\cdots k_0}(z)$ has symmetry property for k. This can reduce computational complexity by half. The symmetry of $C^{p_{t-r-1}\cdots p_0}_{k_{r-1}\cdots k_0}(z)$ comes from that of $U_p(z)$. The following lemma justifies the symmetry of $U_p(z)$.

Lemma 4 :

$$U_p(z)\equiv U_p(z^{-1}) \bmod (z^{2M}+1). \quad (27)$$

proof :

$$U_p(z^{-1})\equiv\sum_{j=0}^{2M-1} V_p(j)z^{-j} \bmod (z^{2M}+1)$$

$$\equiv\sum_{k=-2M+1}^{0} V_p(-k)z^{k} \bmod (z^{2M}+1)$$

since $V_p(-j)=V_p(2M-j)$ *from* (12)

$$\equiv\sum_{j=1}^{2M} V_p(2M-j)z^{j-2M} \bmod (z^{2M}+1)$$

$$\equiv\sum_{j=1}^{2M} -V_p(j)z^{j}\cdot z^{-2M} \bmod (z^{2M}+1)$$

since

$z^{-2M}\equiv -1 \bmod (z^{2M}+1)$ and $V_p(2M)=V_p(0)$,

$$\equiv\sum_{j=0}^{2M-1} V_p(j)z^{j} \bmod (z^{2M}+1)$$

$$\equiv U_p(z) \bmod (z^{2M}+1)\ .\ \blacksquare$$

Based on Lemma 4, the symmetry of $C^{p_{t-r-1}\cdots p_0}_{k_{r-1}\cdots k_0}(z)$ is as follows.

Lemma 5 : $C^{p_{t-r-1}\cdots p_0}_{k_{r-1}\cdots k_0}(z)$ satisfies the following symmetric property;

$$C^{p_{t-r-1}\cdots p_0}_{2^r-k_{r-1}\cdots k_0}(z)\equiv C^{p_{t-r-1}\cdots p_0}_{k_{r-1}\cdots k_0}(z^{-1}) \bmod (z^{2M}+1) \quad (28)$$

roof :

$$C^{p_{l,r-1}\cdots p_0}_{2^r-k_{r-1}\cdots k_0}(z) \equiv \Big(\sum_{p_{l,r}=0}^{1} \cdots \sum_{p_{l,1}=0}^{1} U_{p_{l,1}\cdots p_0}(z) \cdot \hat{z}^{(p_{l,1}\cdots p_{l,r})\cdot(2^r-k_{r-1}\cdots k_0)} \Big)_{\langle 2M\rangle}$$

$$\equiv \Big(\sum_{p_{l,r}=0}^{1} \cdots \sum_{p_{l,1}=0}^{1} U_{p_{l,1}\cdots p_0}(z) \cdot \hat{z}^{2^r(p_{l,1}2^{l-1}+\cdots+p_{l,r}2^{l-r}) - (p_{l,1}2^{l-1}k_0+\cdots+p_{l,r}2^{l-r}(2^{r-1}k_{r-1}+\cdots+k_0))} \Big)_{\langle 2M\rangle}$$

$$\equiv \Big(\sum_{p_{l,r}=0}^{1} \cdots \sum_{p_{l,1}=0}^{1} U_{p_{l,1}\cdots p_0}(z) \cdot \hat{z}^{2^r(p_{l,1}2^{l-1}+\cdots+p_{l,r}2^0) - (p_{l,1}2^{l-1}k_0+\cdots+p_{l,r}2^{l-r}(2^{r-1}k_{r-1}+\cdots+k_0))} \Big)_{\langle 2M\rangle}$$

from Lemma 3,

$$\equiv \Big(\sum_{p_{l,r}=0}^{1} \cdots \sum_{p_{l,1}=0}^{1} U_{p_{l,1}\cdots p_0}(z) \cdot \hat{z}^{-(p_{l,1}2^{l-1}k_0+\cdots+p_{l,r}2^{l-r}(2^{r-1}k_{r-1}+\cdots+k_0))} \Big)_{\langle 2M\rangle}$$

from Lemma 4,

$$\equiv \Big(\sum_{p_{l,r}=0}^{1} \cdots \sum_{p_{l,1}=0}^{1} U_{p_{l,1}\cdots p_0}(z^{-1}) \cdot \hat{z}^{-(p_{l,1}2^{l-1}k_0+\cdots+p_{l,r}2^{l-r}(2^{r-1}k_{r-1}+\cdots+k_0))} \Big)_{\langle 2M\rangle}$$

$$\equiv C^{p_{l,r-1}\cdots p_0}_{k_{r-1}\cdots k_0}(z^{-1}) \bmod (z^{2M}+1). \quad \blacksquare$$

Lemma 5 indicates that only half of the coefficients $C_k(z)$ need to be computed. This property enables us to obtain $C^{p_{l,r-1}\cdots p_0}_{k_{r-1}\cdots k_0}(z)$, $2^{r-1}+1 \le k_{r-1}\cdots k_0 \le 2^r-1$ from $C^{p_{l,r-1}\cdots p_0}_{k_{r-1}\cdots k_0}(z)$, $1 \le k_{r-1}\cdots k_0 \le 2^{r-1}$. For example, in case of M=4, and r=3,

$$C^{p_{l,r-1}\cdots p_0}_{k_{r-1}\cdots k_0}(z) = c_0z^0 + c_1z^1 + c_2z^2 + c_3z^3 + c_4z^4 + c_5z^5 + c_6z^6 + c_7z^7 \text{ and then}$$

$$C^{p_{l,r-1}\cdots p_0}_{2^r-k_{r-1}\cdots k_0}(z) \equiv C^{p_{l,r-1}\cdots p_0}_{k_{r-1}\cdots k_0}(z^{-1}) \bmod (z^8+1)$$

$$\equiv (c_0z^{-0} + c_1z^{-1} + c_2z^{-2} + c_3z^{-3} + c_4z^{-4} + c_5z^{-5} + c_6z^{-6} + c_7z^{-7}) \bmod (z^8+1)$$

$$= c_0z^{-0} - c_1z^7 - c_2z^6 - c_3z^5 - c_4z^4 - c_5z^3 - c_6z^2 - c_7z^1 \quad = \quad c_0z^0 - c_7z^1 - c_6z^2 - c_5z^3 - c_4z^4 - c_3z^5 - c_2z^6 - c_1z^7.$$

Therefore, Eq.(26) needs only $N \cdot M$ additions at each stage. So, the polynomial transform in Eq.(22) requires overall $N \cdot M \cdot \log_2 N$ additions and no multiplication. Without the fast polynomial transform, the polynomial transform requires $N \cdot 2M \cdot (N-1)$ additions.

Fig.11 shows the computational flow of the fast polynomial transform for N=8. At mark ○, half coefficients are computed by the butterfly-operation. The other half coefficients are obtained from the computed half of coefficients. At Mark •, all coefficients are computed by butterfly-operation.

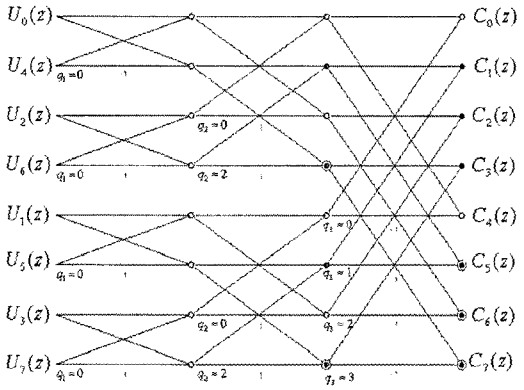


그림 11. N=8일 때, 다항식변환 알고리즘의 흐름도. $\overset{q}{\times}$ 는 <그림 10>의 butterfly 연산을 나타낸다.

Fig. 11. where N=8, the flow graph of polynomial transform algorithm. $\overset{q}{\times}$ represents the butterfly operations in Fig. 10.

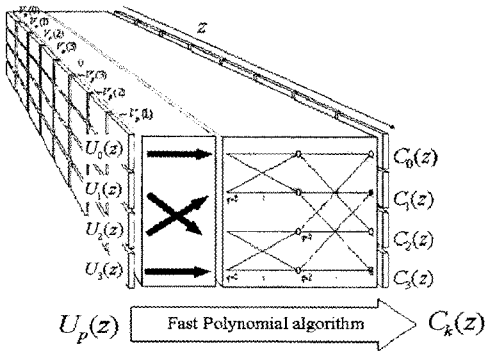


그림 12. 고속 다항식변환

Fig. 12. the Fast Polynomial Transform $U_p(z) \leftarrow FastP.T. \rightarrow C_k(z)$.

Mark ◉ fetches all coefficients computed at those of mark •. For example, at $N=M=8$, the polynomial transform in Eq.(22) requires only 192 additions by using the fast algorithm, where the polynomial transform without using the fast algorithm requires 896 additions.

Fig.12 represents the computational procedure of Eq.(22) via fast polynomial transform.

V. Overall procedure

In this section, we summarize that the 2D-DCT-II of $x(n,m)$ is $X(k,l)$ $(k=0,1,\cdots,N-1;\ l=0,1,\cdots,M-1)$, which can be computed in the following steps.

Step 1) Permute $x(n,m)$ into $\hat{y}_p(m)$

1-1) Permute $x(n,m)$ into $y(n,m)$ from Eq.(2),

$$\begin{aligned} &y(n,m)=x(2n,2m)\\ &y(N-1-n,m)=x(2n+1,2m)\\ &y(n,M-1-m)=x(2n,2m+1)\\ &y(N-1-n,M-1-m)=x(2n+1,2m+1) \end{aligned}$$, where

$n=0,1,\cdots,N/2-1$
$m=0,1,\cdots,M/2-1.$

1-2) Permute $y(n,m)$ into $\hat{y}_p(m)$ from Eq.(9),

$$\begin{cases} \hat{y}_p(2m)=y(p(m),m)\\ \hat{y}_p(2m+1)=y(p(M-1-m),M-1-m) \end{cases}$$

where $m=0,1,\cdots,M/2-1$;

$p(m)=[(4p+1)m+p] \bmod N$,

Step 2) Compute 1D-DCT-II for $\hat{y}_p(m)$ from Eq.(10).

$$V_p(j)=\sum_{m=0}^{M-1}\hat{y}_p(m)\cos\left[\frac{\pi(2m+1)j}{2M}\right].$$

Step 3) Compute the fast Polynomial transform from Eq.(22).

$$C_k(z)\equiv\sum_{p=0}^{N-1}U_p(z)\,\hat{z}^{\,pk} \bmod (z^{2M}+1),\ \text{where}$$

$$U_p(z)=\sum_{j=0}^{2M-1}V_p(j)z^j.$$

Step 4) Re-ordering of coefficients of $C_k(z)$ from Eq.(20).

$$B_k(z)\equiv C_k(z)z^{2'k} \bmod (z^{2M}+1).$$

Step 5) Compute $X(k,l)=\frac{1}{2}[A(k,l)+B(k,l)]$, where $A(k,l)$ and $B(k,l)$ are coefficients of $B_k(z)$ as defined by

$$B_k(z)=\sum_{l=0}^{M-1}B(k,l)z^l-\sum_{l=M}^{2M-1}A(k,2M-l)z^l.$$

Step1) is index mapping with no computational operations. Each sub-step 1-1) and 1-2) is actually combined into one mapping. Step2) is computation of 1D-DCT. We assume that the most efficient 1D-DCT-II fast algorithm[9] is used. Then the number of multiplications and additions of 1D-DCT are $1/2M\log_2 M$ and $3/2M\log_2 M-M+1$ for each $p\,(=0,\cdots,N-1)$. So, in step2), $1/2NM\log_2 M$ multiplications and $3/2NM\log_2 M-NM+N$ additions are needed. Step3) requires only $NM\log_2 N-N/2+1$ additions. Step4) is permutation of $C_k(z)$'s coefficients. So, this step does not also involve any computational operations. Step5) needs $NM-N-M+1$ additions as explained in section III. Therefore, assuming that 1D-DCT is computed by Lee's fast algorithm[9], computing NxM 2D-DCT-II with the suggested algorithm needs total $3/2NM\log_2 M+NM\log_2 N-N/2-M+2$ additions

표 1. N×N 입력에 대한 2D-DCT-II의 계산 복잡도의 비교(a : 덧셈의 수, μ : 곱셈의 수)

Table 1. Comparison of computational complexity of 2D-DCT-II for an N×N block input.(α : the number of additions, μ : the number of multiplications).

input sequence size(NxM)	row-column algorithm[1]	other fast algorithm[10]	Proposed algorithm
4×4	α=72, μ=32	α=70, μ=16	α=76, μ=16
8×8	α=464, μ=192	α=474, μ=104	α=470, μ=96
16×16	α=2592, μ=1024	α=2570, μ=568	α=2538, μ=512
32×32	α=13376, μ=5120	α=12970, μ=2840	α=12754, μ=2560

and $1/2NM\log_2 M$ multiplication, where the conventional row-column method requires $3/2NM\log_2 NM - 2NM + N + M$ additions and $1/2NM\log_2 NM$ multiplications.

Table.1 compares computational complexities of the proposed algorithm, M.Vetterli's fast algorithm[10] and conventional row-column algorithm. And we exclude other algorithms using polynomial transform[2, 3, 8] in the comparison, because those algorithms involve complex operations so that the algorithms inherently exceed complexities of the compared algorithms.

VI. Conclusions

This paper suggests a novel fast algorithm computing 2D-DCT. The suggested algorithm is based on the polynomial transform that does not involve complex operations. The suggested algorithm exploits the symmetric properties of polynomial so as to reduce the computational complexity and construct recursive computational structure. Since the previous 2D-DCT algorithms using polynomial transform were derived from the fast algorithm based on FFT[2, 3, 8], the algorithms include complex computation. However, the proposed algorithm directly applies the polynomial transform to DCT computation so that the proposed method does not involve any complex operations.

참 고 문 헌

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

[2] J. Prado and P. Duhamel, "A polynomial transform based. computation of the 2-D DCT *with minimum multiplicative complexity*", *in* Proc. ICASSP, vol.3, 1996, pp.1347-1350.

[3] P. Duhamel and C. Guillemot, "Polynomial transform computation of 2-D DCT," in Proc. ICASSP'90, pp.1515-1518, Apr. 1990.

[4] E. Feig and S. Winograd, "Fast Algorithms for the discrete cosine transform" in IEEE Trans. Signal Processing. vol.40, pp.2171-2193, Sept. 1992.

[5] R. E. Blahut, Fast Algorithms for Digital Signal Processing, MA: Addison-Wesley, 1984.

[6] H. J. Nussbaumer, "New algorithms for convolutions and DFT based on polynomial transforms" in Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing, pp.638-641, 1978.

[7] N. I. Cho and S. U. Lee, "Fast Algorithm and implementation of 2-D discrete cosine transform" in IEEE Trans. Circuits Syst., vol.38, pp.297-305, March 1991.

[8] H. J. Nussbaumer and P. Quandalle, "Polynomial transform computation of the 2-D DCT," in Proc. ICASSP, 1990, pp.1515-1518.

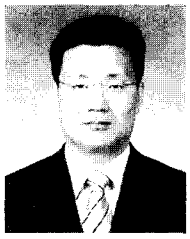
[9] B. G. Lee, "A New algorithm to compute the discrete cosine transform." IEEE Trans. Acoust. Speech, Signal Processing, vol. ASSP-32, pp.1243-1245, Apr. 1984.

[10] M. Vetterli, "Fast 2D discrete cosine transform," in Proc. ICASSP'85, Mar. 1985.

저 자 소 개

崔 桓 碩(正會員)

2002년 2월 : 명지대학교 전자공학과 졸업. 2002년 3월~현재 : 명지대학교 대학원 정보통신공학과 석사과정. <주관심분야 : 멀티미디어 신호처리, 동영상 압축>



金 元 河(正會員)

1985년 2월 : 연세대학교 전자공학과(학사). 1988년 5월 : University of Wisconsin-Madison 전기공학과(석사). 1996년 1월~1996년 8월 : (미) Motorola Software 연구원. 1997년 5월 : University of Wisconsin-Madison 전기공학과(박사). 1997년 8월~2002년 2월 : (미) Los Alamos 국립 연구소 연구원. 2000년 3월~2003년 8월 : 명지대학교 전자정보통신공학부 조교수. 2003년 9월~현재 : 경희대학교 전자정보대학 전자공학전공 조교수. <주관심분야 : 멀티미디어 신호처리, Wavelet 응용, Wavelet 기반 동영상 코딩, 영상처리>