

碩士學位論文

改善된 고속 DCT 알고리즘에  
관한 研究



濟州大學校 大學院

제주대학교 중앙도서관  
通信工學科  
JEJU NATIONAL UNIVERSITY LIBRARY

咸 映 希

1999年 12月

改善된 고속 DCT 알고리즘에  
관한 研究

指導教授 林 載 允

咸 映 希

이 論文을 工學 碩士學位 論文으로 提出함



咸映希의 工學 碩士學位 論文을 認准함

審査委員長 李 鎔 鶴 印

委 員 康 鎭 植 印

委 員 林 載 允 印

濟州大學校 大學院

1999年 12 月

# A Study on Improved Fast DCT Algorithm

**Young-Hee Ham**

(Supervised by professor Jae-Yun Lim)



**A THESIS SUBMITTED IN PARTIAL FULFILLMENT  
OF THE REQUIREMENTS FOR THE DEGREE OF  
MASTER OF ENGINEERING**

**DEPARTMENT OF TELECOMMUNICATION ENGINEERING  
GRADUATE SCHOOL  
CHEJU NATIONAL UNIVERSITY**

**1999. 12.**

# 목 차

Abstract .....	1
I. 서론 .....	2
II. 영상 부호화 .....	5
1. 영상의 일반적인 압축 기법 .....	5
2. 영상 압축의 종류 .....	7
3. 일반적인 DCT 알고리즘 .....	8
III. 고속 $4 \times 4$ DCT 알고리즘 .....	14
1. $4 \times 4$ DCT의 분할 .....	14
2. 4 point DCT .....	15
3. Butterfly 변환 .....	23
IV. 실험 결과 및 비교 검토 .....	27
V. 결론 .....	29
참고문헌 .....	30

## Abstract

This paper presents an effective algorithm for the computation of  $4 \times 4$  DCT(Discrete Cosine Transform). The algorithm is based on the decomposition of  $4 \times 4$  DCT into four 4-point 1-D DCT's. Thus, only 1-D transformations and some additions are required. It is shown that the proposed algorithm requires only 16 multiplications, which is half of the conventional row-column method. Since  $2^m \times 2^n$  DCT can be computed using  $4 \times 4$  DCT recursively for any  $m$ , the proposed algorithm leads to a fast algorithm for the computation of 2-D DCT.



# I. 서론

영상압축은 영상을 표현하는 데 필요한 비트의 수를 줄이기 위하여 디지털 영상을 효율적으로 부호화하는 학문이다. 영상압축을 행하는 목적은 영상의 특성을 유지하면서, 저장용량과 전송비용을 줄이는 것이다. G4 FAX, HDTV(High Definition TV), 영상전화, 레이저 인쇄기, 전자식 정지영상 카메라, 동화상 카메라, VCR 등 영상신호를 다루는 기기 들은 다량의 정보를 처리하여야 하며 큰 전송 대역폭이 요구된다. 또한 최근 주목을 받고 있는 ISDN( 종합정보통신망 : Integrated Service Digital Network )으로 영상신호를 전송하기 위해서는 높은 데이터의 압축이 필연적이다. 그래서 영상 데이터를 압축하기 위한 영상부호화(Image Coding)가 영상의 통계적 성질들을 이용하여 지금까지 지속적으로 발전하여 왔고, 이러한 영상부호화의 표준안들이 제안되고 있다.

제안된 표준안들은 처리하는 대상 영상에 따라 구분되는 데, 정지영상을 대상으로 하는 부호화의 표준안인 JPEG(Joint Photographic Experts Group) 알고리즘은 영상 데이터 압축기법인 DCT(Discrete Cosine Transform)를 근간으로 확정되어 있고, 동영상을 대상으로 하는 영상부호화의 표준안인 MPEG(Moving Picture Experts Group) 알고리즘에서도 DCT가 매우 중요한 역할을 하고 있다(I-Ming, 1999).

DCT는 하나의 변환부호화(Transform Coding)로써  $N \times N$  영상 신호들을 적은 수의 영상신호에 정보를 집중시키고 나머지 신호들은 '0' 또는 작은 값을 갖도록 처리한다(Ahmed, 1975). 이러한 특성은 통신 선로의 대역폭 감소 문제 및 정보 저장에 필요한 기억용량의 감축 문제를 동시에 해결하기 때문에 디지털 영상 신호 처리 사용 시 발생하는 경제성 문제를 해결하는 핵심 기술의 하나로 연구되고 있다. 그러나 이 영상압축 기법은 많은 계산 량으로 인해 실시간 영상 데이터 처리에 어려움이 있다.

많은 고속 DCT 알고리즘들은 계산 량을 줄이기 위해 변환 행렬을 분해하여 상당수의 계수를 1 또는 0으로 대치시켰다. 따라서 이들 알고리즘들을 적용하는 경우 곱셈 횟수는 감소하고 코사인 항의 종류도 줄어든다(Chen, 1977, Wagh, 1980,

Lee, 1984, Hou, 1987, Cho, 1990). 이와 같은 고속 DCT 알고리즘들은 대부분 1차원 DCT를 위한 것인데 일반적으로 2차원 신호인 영상신호에 대한 2차원 DCT의 계산에는  $N \times N$  입력신호에 대하여 행(열)방향으로 1차원 DCT를 수행한 후에 다시 열(행)방향으로 1차원 DCT를 수행하는 행-열법( Row-column method )이 많이 사용된다(Makhoul, 1980, Kamagar, 1982, Haque, 1985). 즉,  $N \times N$ 의 2차원 DCT 변환을 위하여  $2N$ 개의 1차원 DCT를 순차적으로 수행하는 것이다. 한편, 이 방법 외에도 2차원 DCT를 빠르게 계산하기 위하여 2차원 입력 데이터를 직접 다루는 2차원 고속 DCT 알고리즘도 소개되어왔다(Vetterli, 1985). 곱과 덧셈의 개수라는 관점에서 볼 때 이러한 고속 알고리즘 가운데 가장 효율적인 것은 직접 다항 변환 알고리즘이다(Duhamel, 1990).

이 알고리즘은 재래적인 알고리즘에 비해 곱셈의 수를 50%까지 줄였다. 반면 반복적 2차 DCT 알고리즘에서는 종전에 비해 75%의 곱의 수를 필요로 하며, Vetterli가 제시한 다항 변환 FFT에 rotation을 첨가한 간접 다항 변환법은 75%와 50%사이를 필요로 한다( Wu, 1989, Ma, 1988). 그러나 다항 변환법에 기초를 둔 방법들은 구조가 복잡하다. 특히 변환 크기가 큰 경우는 더욱 더 복잡하다. 반복적 2차 DCT 알고리즘은 Hou의 1차 DCT를 2차 데이터 집합으로 확장시킨 것이다.

반복적 2차 DCT 알고리즘에서 필요로 하는 곱의 수는 Haque가 제안한 알고리즘 혹은 Wu와 Paoloni가 제안한 알고리즘에서 필요로 하는 곱의 수와 같은 반면 간접 다항 변환 알고리즘과 직접 다항 변환 알고리즘에서 필요로 하는 곱의 수보다는 훨씬 많다. 하지만 반복적 2차 DCT 알고리즘의 관심 가질 만한 장점은 이 알고리즘의 구조가 규칙적이고 체계적이며 오로지 실수 연산만이 필요시 되어진다는 것이다. 반면 다항 변환법에 기초를 둔 알고리즘은 구조와 계산 체계가 복잡하다.

2차 반복적 DCT 알고리즘의 주된 개념은  $N \times N$  DCT가 바로 한 단계 낮은 DCT 행렬로부터 계산되어 질 수 있다는 것이다. 바꾸어 말하면  $2^n \times 2^n$  DCT는 4개의  $2^{n-1} \times 2^{n-1}$  DCT로 분해되어질 수 있다. 따라서 모든  $2^n \times 2^n$  DCT는 기본적 모듈로써  $4 \times 4$  DCT를 이용해 계산되어질 수 있다. 지금부터 고속  $4 \times 4$  DCT의 전개는  $2^n \times 2^n$  DCT의 효율적인 알고리즘으로 통한다. 그러나 반복적 2차 DCT 알고리즘에서 이용된  $4 \times 4$  DCT 알고리즘은 24개의 곱을 필요로 하는데, 이는  $2^n \times 2^n$  DCT 계산을 하는데 있어 Haque가 제안한 알고리즘과 Wu와 Paoloni가 제안한 알고리즘에서 필요로 하

는 곱의 수와 같다.

본 논문에서는  $4 \times 4$  DCT를 4개의 병렬 4-point DCT로 분해함으로써 16개의 곱만을 요구하는 새로운  $4 \times 4$  DCT알고리즘을 제시한다. 따라서  $2^n \times 2^n$  DCT 계산을 위한 곱의 수는 반복적 2차 DCT 알고리즘의 곱의 수보다는 적으나 간접 다항 변환법과 직접 다항 변환 알고리즘의 곱셈 수보다는 다소 많다.  $4 \times 4$  DCT에 한해서만 생각하면 본 논문에서 제시하는 알고리즘은 간접 다항 변환 알고리즘과 직접 다항 변환 알고리즘에서 이용된 곱의 수와 비교했을 때 같은 수가 필요하나, 제안된  $4 \times 4$  DCT 알고리즘을 위한 구조가 다른 알고리즘의 구조보다 훨씬 더 간단하다.

본 논문의 구성은 다음과 같다. II장에서는 영상 압축의 일반 및 DCT 알고리즘을 소개하고, III장에서  $2^n \times 2^n$  DCT의 기본 모듈인 고속  $4 \times 4$  DCT 알고리즘을 제시하고, IV장에서는 이 알고리즘을 반복적으로 사용하여 2차 DCT를 위해 필요한 곱셈과 덧셈의 수를 다른 알고리즘과 비교하고, 마지막으로 V장에서는 결론을 맺는다.



## II. 영상 부호화

영상압축은 영상 데이터의 중복성을 이용하여 이루어진다. 영상 데이터의 중복성에는 스펙트럼 중복성, 시간 중복성, 공간 중복성, 통계적 중복성이 있다. 스펙트럼 중복성은 다른 칼라 플레인(color planes)간의 상호 관계로 인한 것이며, 시간 중복성은 텔레비전 회의의 응용 혹은 방송 영상과 같은 연속적인 영상에 있어서 서로 다른 프레임들 간의 상호 관계로 인한 것이다. 공간 중복성은 이웃해 있는 픽셀들 간의 상호 관계에 기인한다.

영상 압축에는 무 손실 압축과 손실 압축이 있으며, 무 손실 압축 법으로는 PCX, GIF, TIFF 등이 있고, 손실 압축 법으로는 JPEG, JBIG, MPEG 등이 있다.

### 1. 영상의 일반적인 압축 기법



일반적인 영상 데이터에는 4가지 중복성이 존재하는 데, 이 4가지 중복성을 효율적으로 제거함으로써 영상의 압축률을 높일 수 있다.

사람의 눈은 색도 성분보다는 명암 성분에 민감하다는 특징을 이용하여 RGB 영상을 4:1:1의  $YCbCr$ 로 변환하여 화면 내에 존재하는 화소간의 중복성을 효과적으로 줄이며, 움직임 벡터의 추정과 움직임의 보상 및 DPCM을 이용하여 화면간에 존재하는 시간적 중복성을 제거한다. 그리고 DCT와 양자화를 이용하여 공간적인 중복성을 제거하며, 이러한 과정을 거친 데이터를 최종적으로 발생빈도에 따른 통계적 중복성을 제거한다.

#### 1) 스펙트럼 중복성(Spectrum Redundancy)

렌즈에서 수신된 영상신호(RGB)는 3원색 좌표계로 표현되며 이들 성분들은 색의 밝기를 나타내는 명암성분과 색차를 나타내는 색도 성분으로 변환된다. 인간의 시

각은 색도 성분에 둔감하므로 색도 성분을 1/4 크기로 부분 샘플링하여 대역간의 중복성을 제거한다.

하나의 블록에 대하여 그림 1과 같이 Y: C<sub>b</sub> : C<sub>r</sub>를 4 : 1 : 1로 샘플링하여 전송하면 C<sub>b</sub>와 C<sub>r</sub>의 데이터 양을 1/4로 줄일 수 있다.

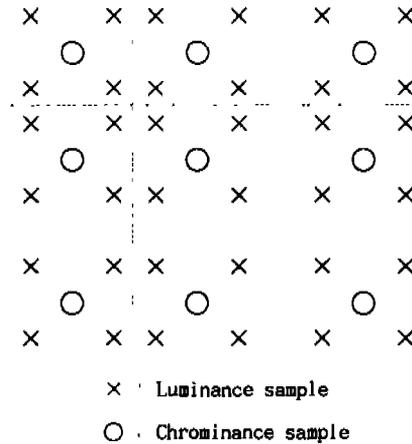


Fig 1. The sampling rate of Y : C<sub>b</sub> : C<sub>r</sub>



### 2) 시간적 중복성(Temporal Redundancy)

동영상에서 인접 프레임간에는 서로 공통된 성분을 포함하고 있으며 이러한 시간 중복성을 제거하기 위하여 움직임 벡터 추정과 움직임 벡터 보상을 이용한다.

### 3) 공간 중복성 (Spatial Redundancy)

영상 내의 화소와 화소간 상관성 즉, 공간 중복성은 변환 부호화를 이용하여 제거한다. 변환 부호화 방법으로는 DFT(Discrete Fourier Transform), KLT(Karhunen-Loeve Transform), ST(Slant Transform), WHT(Walsh Hadamard Transform), DST(Discrete Sine Transform), DCT(Discrete Cosine Transform)등이 있다.

KLT는 영상 데이터의 공분산(covariance)에 근접하여 그 영상이 가지는 정보의 대부분을 최소의 변환 계수에 집중시키는 최적의 변환 기법인데 계산 량이 많은 관계로 고속처리가 쉽지 않은 알고리즘이다. WHT는 단순하여 그 구성이 간단하고 고속처리가 가능하나 압축률이 떨어진다는 단점을 가지고 있다. DCT는 성능 면에

서 KLT에 가장 근접한 변환이고 고속 알고리즘이 개발되어 있다. 본 논문에서는  $4 \times 4$  DCT를  $4 \times 1$  DCT 4개로 분할하여 반복적 2차 DCT 알고리즘보다 속도를 높이는 반면 직접 다항 변환법 혹은 간접 다항 변환법보다 구조에 있어 강력한 알고리즘을 제안하고 있다.

#### 4) 통계적 중복성(Statistical Redundancy)

스펙트럼, 시간 및 공간 중복성을 제거하고 변환 계수에 대하여 런랭스 부호화를 적용한 후 얻어진 심벌들에 대하여 가변길이 부호화(VLC)와 같은 엔트로피 부호화를 거침으로써 통계적 중복성을 제거한다.

## 2. 영상 압축의 종류

영상 압축의 종류로는 무 손실 압축과 손실 압축이 있다.



### 1) 무 손실 압축

무 손실 압축은 의료 영상과 같이 원래의 정보가 손실되면 중대한 결과를 초래할 수 있는 영상 정보를 압축하는 데 사용되며, 압축률은 2 : 1 이하로 매우 낮다. 정보이론에 의하면 랜덤한 정보를 손실 없이 이 이상 압축하는 것은 불가능하다. 따라서 이 방법은 압축률을 중요시하기보다는 화질 저하가 나타나서는 안 되는 곳에 사용된다. PCX, GIF, TIFF 등이 이 부류에 속한다.

그림 2는 무 손실 압축을 위한 부호화의 처리 과정을 나타낸 것으로, 그림 2에서 예측기(predictor)는 인접한 3개의 화소로부터 현재의 화소 값을 예측하며, 엔트로피(entropy) 부호기에서는 예측된 화소 값과 실제의 화소 값을 비교하여 두 값의 차에 대하여 허프만(Huffman) 부호화 또는 산술 부호화 알고리즘을 적용한다.

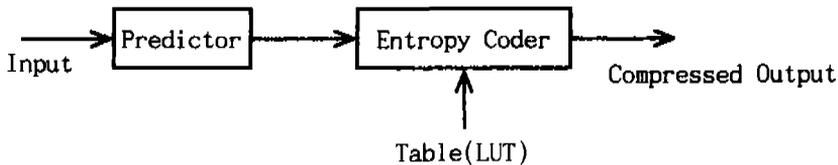


Fig 2. The block diagram of lossless compression

## 2) 손실 압축

무 손실 압축과는 달리 손실을 허용하는 압축 방법으로는, 보통 변환 부호화 (Transform Coding) 기술을 사용한다. 변환 부호화는 10 : 1 또는 수십 : 1 정도의 압축률을 내는 방법을 말하는 데, JPEG와 MPEG가 바로 여기에 속한다. 손실이 없는 방법에 비하면 계산 시간이나 메모리 요구 량이 훨씬 많지만 향상된 압축률이 이를 충분히 보상해 준다. 실제로 많은 응용에서 최소한 10 : 1 많으면 100 : 1 까지도 압축을 해야하는 경우가 있다. 하지만 변환 부호화에 사용되는 변환의 특성이 우수한 것과 함께 우리들의 눈의 정확도가 떨어지기 때문에 대부분의 응용에 사용할 수 있는 방법은 변환 부호화를 이용한 손실을 허용하는 압축이다.

변환 부호화에 사용되는 변환들은 DFT(Discrete Fourier Transform), KLT(Karhunen-Loeve Transform), ST(Slant Transform), WHT(Walsh Hadamard Transform), DST(Discrete Sine Transform), DCT(Discrete Cosine Transform) 등이 있는데, 이 중에서 DCT가 가장 압축률을 높일 수 있는 것으로 알려져 있으나, 많은 계산 량으로 인해 실시간 영상 데이터 처리에 어려움이 있다. 따라서 계산 량 및 시간을 줄이기 위해 많은 고속 DCT 알고리즘들이 연구되고 있다.

## 3. 일반적인 DCT 알고리즘

많은 변환 부호화들이 영상 부호화에 이용되어 왔으며, 그 변환의 수행 성과는 원 영상이 복원된 영상의 충실도와 비교되어짐으로써 알 수 있다. 그림 3과 그림 4는 변환 압축에서 영상 데이터의 일반적인 흐름도를 나타낸 것이다.

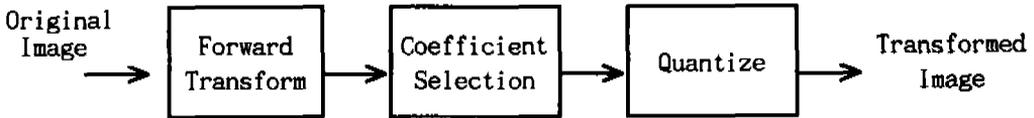


Fig 3. The transform compression

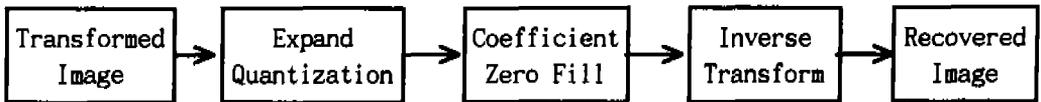


Fig 4. The recovery of compressed image

표본화된 영상이 변환되어지고, 그 결과로 나온 계수들 중 많은 부분이 선택되어지고, 남아 있는 계수들은 원래 영상보다 훨씬 적은 비트로 재 양자화되어진다. 영상의 복원을 위해서는 반대 과정이 수행되어진다. 즉, 계수들이 원래 비트 수대로 재 양자화되어지며, 없어진 계수들은 일정한 값으로 (예를 들어 0) 대체된 후, 역변환이 수행된다.

각 과정의 충실도는 배열의 각 요소에서 복원된 영상과 원래 영상간의 크기 레벨 차를 택함으로써 측정되어 질 수 있다. 만일 원래 영상이  $f_{org}[m,n]$ 이고, 복원된 영상이  $f_{rec}[m,n]$ (모두  $N \times N$  행렬)이라면, 그 때 차 배열은 식 (1)과 같다.

$$f_{diff}[m,n] = f_{rec}[m,n] - f_{org}[m,n] \quad (1)$$

충실도에 대한 예러 측정 지능 평균은 차 배열로부터 유도되어 질 수 있다. DCT는 KLT의 이론적 한계에 가까운 MSE(Mean Squared Error)결과를 제공하므로 통계학 상 최적의 변환에 접근하여 디지털 신호 처리, 특히 영상과 음성 데이터 압축에 가장 광범위하게 사용되어 왔다. 변환 부호화의 일종으로 다른 알고리즘 보다 우수한 특성을 갖고 있으며, 화소와 화소간 공간의 중복성을 제거하기 위하여 이용된다.

1) 1차원 DCT

1-D DCT 과정은 입력 데이터 열  $x(n)$ 에 대하여  $2N$ 점으로 확장 표현하고 이를  $2N$ 점 DFT을 통하여 다시 Cosine 형태로 변환하는 것이다.

입력 데이터 열  $x(n)$ 가  $0 \leq n \leq N-1$ 인 범위에서  $0 \leq n \leq 2N-1$ 인 범위로 확장되면 식 (2)와 같이 정의된다.

$$x_e(n) = \begin{cases} x(n), & 0 \leq n \leq N-1 \\ x(2N-1-n), & N \leq n \leq 2N-1 \end{cases} \quad (2)$$

여기서  $x_e(n)$ 은 입력 데이터 열  $x(n)$ 에 대하여  $N$ 점 입력 데이터 열을  $2N$ 점 입력 데이터 열로 확장한 표현이다.

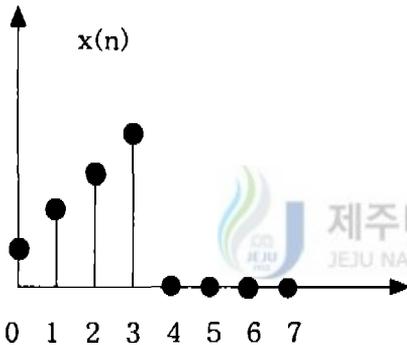


Fig 5. Input data sequence for  $N=4$

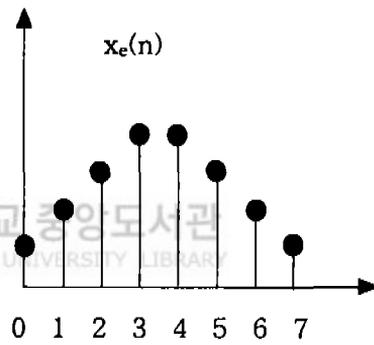


Fig 6. Expanded data sequence

그림 5와 그림 6을 보면  $N$ 점 입력 데이터 열을  $2N$ 점으로 확장했을 때 이 데이터 열이  $n = N - \frac{1}{2}$ 을 기준으로 대칭을 이루며  $N$ 점마다 반복된다.

$2N$ 점 DFT  $D(k)$ 는 식 (3)과 같이 정의된다.

$$D(K) = \sum_{n=0}^{2N-1} x_e(n) W_{2N}^{kn} \quad 0 \leq k \leq 2N-1 \quad (3)$$

여기서

$$W_{2N} = e^{-j \frac{2\pi}{2N}} \quad (4)$$

이다

식 (3)과 식 (4)에 의하여

$$D(K) = \sum_{n=0}^{N-1} x(n) W_{2N}^{kn} + \sum_{n=N}^{2N-1} x(2N-1-n) W_{2N}^{kn}; \quad 0 \leq k \leq 2N-1 \quad (5)$$

식 (5)는 다음과 같이 Cosine 형태로 표현된다.

$$D(K) = W_{2N}^{-\frac{k}{2}} \sum_{n=0}^{N-1} 2x(n) \cos \frac{\pi}{2N} k(2n+1), \quad 0 \leq k \leq N-1 \quad (6)$$

$x(n)$ 의  $N$ 점 DCT된,  $Y(k)$ 는 다음 식에 의해  $D(k)$ 로부터 구할 수 있다.

$$Y(k) = \begin{cases} W_{2N}^{\frac{k}{2}} D(k), & 0 \leq k \leq N-1 \\ 0, & \text{otherwise} \end{cases} \quad (7)$$

식 (6)과 식 (7)에 의해

$$Y(k) = \begin{cases} \sum_{n=0}^{N-1} 2x(n) \cos \frac{\pi}{2N} k(2n+1), & 0 \leq k \leq N-1 \\ 0, & \text{otherwise} \end{cases} \quad (8)$$



식 (8)은  $x(n)$ 에 대한 DCT의 정의이며  $N$ 점 DCT열이 된다. 만약  $x(n)$ 이 실수이면  $Y(k)$ 도 실수이며  $x(n)$ 이 복소수이면  $Y(k)$ 도 복소수가 된다. 여기서 역 DCT는 유도된 DCT 쌍으로부터 위의 역 과정을 통해서 구할 수 있다.

## 2) 2차원 DCT

2-D DCT 과정은 1-D DCT 과정과 유사하며 1차원 입력 데이터 열을 2차원 입력 데이터 열로 확장하여 표현한다. 2차원 입력 데이터 열  $x(i, j)$ 를  $0 \leq i < N_1 - 1, 0 \leq j < N_2 - 1$ 의 범위에서  $N_1 \times N_2$ 점을 가진 2-D 데이터 열이라 정의하면  $x(i, j)$ 를 새로운 입력  $2N_1 \times 2N_2$ 점을  $x_e(i, j)$ 로 관련시켜 표현하면 식 (9)와 같다.

$$x_e(i, j) = \begin{cases} x(i, j) & 0 \leq i \leq N_1 - 1, 0 \leq j \leq N_2 - 1 \\ x(2N_1 - 1 - i, j) & N_1 \leq i \leq 2N_1 - 1, 0 \leq j \leq N_2 - 1 \\ x(i, 2N_2 - 1 - j) & 0 \leq i \leq N_1 - 1, N_2 \leq j \leq 2N_2 - 1 \\ x(2N_1 - 1 - i, 2N_2 - 1 - j) & N_1 \leq i \leq 2N_1 - 1, N_2 \leq j \leq 2N_2 - 1 \end{cases} \quad (9)$$

$N_1=3$ 이고  $N_2=4$  일 때  $x(j, j)$ 와  $x_e(i, j)$ 의 예를 각각 그림 7과 그림 8에 보여진다.

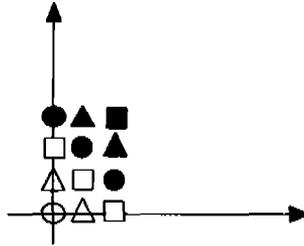


Fig 7. The input data sequence for  $N_1=3, N_2=4$

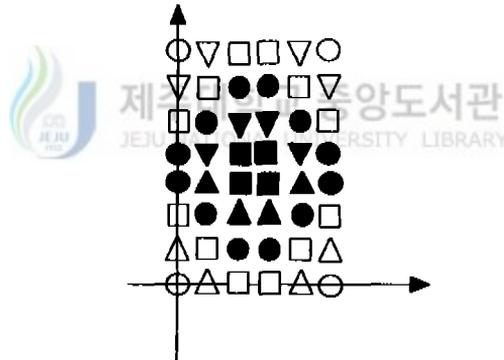


Fig 8. The expanded data sequence

$x_e(i, j)$ 는 식 (9)에 의해  $D(m, n)$ 을 구할 수 있다.

$$D(m, n) = DFT[x_e(i, j)] \quad (10)$$

입력 데이터 열  $x(i, j)$ 에 대한  $N_1 \times N_2$ 점 DCT 표현식,  $Y(m, n)$ 는  $D(m, n)$ 으로 부터 얻어진다.

$$Y(m, n) = \begin{cases} W_{2N_1}^{\frac{m}{2}} W_{2N_2}^{\frac{n}{2}} D(m, n), & 0 \leq m \leq N_1 - 1, \quad 0 \leq n \leq N_2 - 1 \\ 0, & \text{otherwise} \end{cases} \quad (11)$$

식 (10)과 식 (11)에 의해 아래와 같은 2-D DCT 알고리즘을 유도해 낼 수 있다.

$$Y(m, n) = \begin{cases} \sum_{i=0}^{N_1-1} \sum_{j=0}^{N_2-1} x(i, j) \cos \frac{\pi}{2N_1} m(2i+1) \cos \frac{\pi}{2N_2} n(2j+1) \\ \text{for } 0 \leq m \leq N_1 - 1, \quad 0 \leq n \leq N_2 - 1 \\ 0, & \text{otherwise} \end{cases} \quad (12)$$

식 (12)은 2-D DCT의 정의이다.

역 DCT는  $Y(m, N)$ 로부터  $D(m, n)$ 를 유도하고 역 DFT를 통하여  $x_e(i, j)$  구한 다음  $x(i, j)$ 을 얻을 수 있다.



### III. 고속 4×4 DCT 알고리즘

이 장에서는 고속 4×4 DCT 알고리즘을 제안한다. 여기서 유도된 4×4 DCT 알고리즘은 재래적 알고리즘에 비해 계산상 필요한 곱의 수가 반으로 줄어든다.

이 알고리즘은  $2^m \times 2^m$  DCT를 계산하는 데 있어 기본 모듈로 사용되어진다.

#### 1. 4×4 DCT의 분할

2차 데이터 열  $\{ X_{ij} : i, j = 0, 1, \dots, N-1 \}$ 이 주어졌을 때, 그에 대한 2차 DCT열  $\{ Y_{mn} : m, n = 0, 1, \dots, N-1 \}$ 은 다음과 같이 주어진다.

$$Y_{mn} = \frac{4}{N^2} U(m)U(n) \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} x_{ij} \cdot \cos \frac{(2i+1)m}{2N} \pi \cos \frac{(2j+1)n}{2N} \pi \quad (13)$$

$$\text{여기서 } U(m) = \begin{cases} \frac{1}{\sqrt{2}}, & \text{if } m = 0 \\ 1, & \text{그외} \end{cases}$$

편의를 위해,  $Y_{mn}$ 의 비 정규화된 식을 다음과 같이 정의하자.

$$y_{mn} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} x_{ij} \cdot \cos \frac{(2i+1)m}{2N} \pi \cos \frac{(2j+1)n}{2N} \pi \quad (14)$$

제안된 알고리즘의 주된 개념은 다음과 같은 관계식을 사용함으로써 4개의 분리된 4-point DCT로부터 식 (14)에서 정의된 4×4 DCT를 얻을 수 있다는 것이다.

$$\begin{aligned} & \cos \frac{(2i+1)m}{2N} \pi \cos \frac{(2j+1)n}{2N} \pi \\ &= \frac{1}{2} \left( \cos \frac{(2i+1)m + (2j+1)n}{2N} \pi + \cos \frac{(2i+1)m - (2j+1)n}{2N} \pi \right) \end{aligned} \quad (15)$$

$N = 4$ 인 경우, 식 (15)을 사용하면 식 (14)에 있는 2차 DCT는 다음과 같이 2개의 변환식으로 분리되어질 수 있다.

$$y_{mn} = \frac{1}{2} \left( \sum_{i=0}^3 \sum_{j=0}^3 x_{ij} \cos \frac{(2i+1)m+(2j+1)n}{8} \pi + \sum_{i=0}^3 \sum_{j=0}^3 x_{ij} \cos \frac{(2i+1)m-(2j+1)n}{8} \pi \right), \quad (16)$$

for  $m, n = 0, 1, 2, 3$

편의를 위해  $A_{mn}, B_{mn}$ 을 다음과 같이 나타낸다.

$$A_{mn} = \sum_{i=0}^3 \sum_{j=0}^3 x_{ij} \cos \frac{(2i+1)m+(2j+1)n}{8} \pi \quad (17.a)$$

$$B_{mn} = \sum_{i=0}^3 \sum_{j=0}^3 x_{ij} \cos \frac{(2i+1)m-(2j+1)n}{8} \pi \quad (17.b)$$

따라서,  $y_{mn}$ 은 다음과 같다.

$$y_{mn} = \frac{1}{2}(A_{mn} + B_{mn}) \quad (18)$$


## 2. 4 point DCT

이제 몇 개의 데이터를 정렬하고 조작함으로써  $A_{mn}$ 과  $B_{mn}$ 은 사실상 4-point DCT와 동일하다는 사실을 보일 것이다. 따라서 2차 DCT는 4개의 분리된 1차 DCT로부터 얻어질 수가 있다. 1차 DCT의 변환식과 같아지기 위해서 식 (17)에 있는 변환식의 Kernel 조건은  $\{(2i+1)m \pm (2j+1)n\}$ 이  $(2i+1)$ 의 정수 곱으로 표현되어야 한다는 것이다. 그러한 조건을 만족하기 위해서,  $(2j+1)$ 은  $(2i+1) \pmod{2N}$ 의 배수가 되어지던가 혹은  $(2i+1)$ 의 배수에  $N \pmod{2N}$ 이 더해져야 한다는 것이다. 다시 말해서,

$$(2j+1) = p(2i+1) \pmod{2N} \quad (19.a)$$

$$(2j+1) = p(2i+1) + N \bmod 2N \quad (19. b)$$

여기서  $p$ 는 홀수 정수이다. 한편, 식 (19)에서의 조건은 다음과 동가이다.

$$j = pi + \frac{p-1}{2} \bmod N \quad (20. a)$$

$$j = pi + \frac{p-1}{2} + \frac{N}{2} \bmod N \quad (20. b)$$

for  $p = 1, 3, \dots, N-1$

이들 관계식이 의미하는 것은  $N \times N$  입력 데이터 집합을  $N$ 개의 항목으로 그룹화함으로써 각각의 항목은 식 (20)에 따라  $N$ 개의 데이터를 갖으며 식 (17)의 변환식은  $N$ 개의 분리된 1차 DCT로 나타낼 수 있다.

$N = 4$ 일 때 식 (20)을 만족하는  $p$ 의 값은 1 혹은 3이며,  $4 \times 4$  입력 데이터 집합은 식 (20)의 관계식에 따라 4개의 항목으로 그룹지어질 수 있다. 더욱 세분화해서 식 (20)은 다음과 같이 4가지의 경우로 다시 쓸 수 있다.

$$j = i \bmod 4 \quad (21. a)$$

$$j = i + 2 \bmod 4 \quad (21. b)$$

$$j = 1 - i \bmod 4 \quad (21. c)$$

$$j = 3 - i \bmod 4 \quad (21. d)$$

따라서, 입력 데이터 수열  $\{x_{ij} : i, j = 0, 1, 2, 3\}$ 은 다음 4개의 항목으로 그룹지을 수 있다.

$$R_0 = \{x_{ij} : j = i\} \quad (22. a)$$

$$R_1 = \{x_{ij} : j = 3 - i\} \quad (22. b)$$

$$R_2 = \{x_{ij} : j = i + 2 - 4q_1, q_0 = q_1 = 0, q_2 = q_3 = 1\} \quad (22. c)$$

$$R_3 = \{x_{ij} : j = 1 - i + 4q_1, q_0 = q_1 = 0, q_2 = q_3 = 1\} \quad (22. d)$$

이들은 다음과 동가이다.

$$R_0 = \{x_{00}, x_{11}, x_{22}, x_{33}\} \quad (23. a)$$

$$R_1 = \{x_{03}, x_{12}, x_{21}, x_{30}\} \quad (23. b)$$

$$R_2 = \{x_{02}, x_{13}, x_{20}, x_{31}\} \quad (23. c)$$

$$R_3 = \{x_{01}, x_{10}, x_{23}, x_{32}\} \quad (23. d)$$

따라서,  $A_{mn}$ 과  $B_{mn}$ 은 4개의 분리된 1차 변환식의 항으로 표현될 수 있다.

$$A_{mn} = \sum_{k=0}^3 T_k(m, n) \quad (24. a)$$

$$B_{mn} = \sum_{k=0}^3 S_k(m, n) \quad (24. b)$$

여기서

$$T_k(m, n) = \sum_{x_v \in R_k} x_v \cos \frac{(2i+1)m + (2j+1)n}{8} \pi \quad (24. c)$$

$$S_k(m, n) = \sum_{x_v \in R_k} x_v \cos \frac{(2i+1)m - (2j+1)n}{8} \pi \quad (24. d)$$

이와 같은 그룹핑과 정의로  $T_k(m, n)$ 과  $S_k(m, n)$ 은 데이터 수열  $R_k$ 에 대한 1차 DCT 변환과 동가이다.

$k = 0, 1, 2, 3$ 에 대해 식 (22)의  $i$ 와  $j$ 를 식 (24.c)에 대입함으로써  $T_k(m, n)$ 과  $S_k(m, n)$ 을 원래 1차 변환으로 표현해 보자. 첫째, 식 (22.a)를 식 (24.c)에 대입하면  $T_0(m, n)$ 은

$$T_0(m, n) = \sum_{i=0}^3 x_{ii} \cos \frac{(2i+1)(m+n)}{8} \pi \quad (25. a)$$

둘째, 식 (22.b)에서  $j = 3-i$ 를 식 (24.c)에 대입하면  $T_1(m, n)$ 은 다음과 같다.

$$T_1(m, n) = (-1)^n \sum_{i=0}^3 x_{i(3-i)} \cos \frac{(2i+1)(m-n)}{8} \pi \quad (25. b)$$

또한, 식 (22. c)를 식 (24. c)로 대입하면 다음과 같이 전개된다.

$$T_2(m, n) = \sum_{i=0}^3 (-1)^{q_m} x_{i(i+2-4q)} \cos \frac{(2i+1)(m+n)+4n}{8} \pi$$

이 경우에  $n$ 이 짝수 혹은 홀수냐에 따라 두 가지 경우로 나뉘어질 수 있다. 즉,

$$T_2(m, n) = \begin{cases} (-1)^u \sum_{i=0}^3 x_{i(i+2-4q)} \cos \frac{(2i+1)(m+2u)}{8} \pi & n = 2u, u = 0, 1 \quad (25. c) \\ (-1)^u \sum_{i=0}^3 (-1)^{q_i+1} x_{i(i+2-4q)} \cos \frac{(2i+1)(m+2u-3)}{8} \pi & n = 2u+1, u = 0, 1 \quad (25. d) \end{cases}$$

$T_3(m, n)$ 의 표현식은 식 (22. d)의 관계식을 식 (24. c)에 대입하면 다음과 같이 얻을 수 있다.

$$T_3(m, n) = \sum_{i=0}^3 (-1)^{q_m} x_{i(i+4q)} \cos \frac{(2i+1)(m-n)+4n}{8} \pi$$

마찬가지로 이 표현식의  $n$ 의 값이 짝수 혹은 홀수냐에 따라 다음과 같이 두 가지 경우로 나뉘어질 수 있다.

$$T_3(m, n) = \begin{cases} (-1)^u \sum_{i=0}^3 x_{i(1-i+4q)} \cos \frac{(2i+1)(m-2u)}{8} \pi & n = 2u, u = 0, 1 \quad (25. e) \\ (-1)^u \sum_{i=0}^3 (-1)^{q_i+1} x_{i(1-i+4q)} \cos \frac{(2i+1)(m-2u-5)}{8} \pi & n = 2u+1, u = 0, 1 \quad (25. f) \end{cases}$$

같은 방법으로 식 (22)을 식 (24. d)에 대입하면 다음과 같이  $S_k(m, n)$ 식을 얻을 수 있다.

$$S_0(m, n) = \sum_{i=0}^3 x_{ii} \cos \frac{(2i+1)(m-n)}{8} \pi \quad (26. a)$$

$$S_1(m, n) = (-1)^n \sum_{i=0}^3 x_{i(3-i)} \cos \frac{(2i+1)(m+n)}{8} \pi \quad (26. b)$$

$$S_2(m, n) = \begin{cases} (-1)^u \sum_{i=0}^3 x_{i(i+2-4q_i)} \cos \frac{(2i+1)(m-2u)}{8} \pi & (26. c) \\ n = 2u, u = 0, 1 \end{cases}$$

$$\begin{cases} (-1)^u \sum_{i=0}^3 (-1)^{q_i+i+1} x_{i(i+2-4q_i)} \cos \frac{(2i+1)(m-2u+3)}{8} \pi & (26. d) \\ n = 2u+1, u = 0, 1 \end{cases}$$

$$S_3(m, n) = \begin{cases} (-1)^u \sum_{i=0}^3 x_{i(1-i+4q_i)} \cos \frac{(2i+1)(m+2u)}{8} \pi & (26. e) \\ n = 2u, u = 0, 1 \end{cases}$$

$$\begin{cases} (-1)^u \sum_{i=0}^3 (-1)^{q_i+i+1} x_{i(1-i+4q_i)} \cos \frac{(2i+1)(m+2u+5)}{8} \pi & (26. f) \\ n = 2u+1, u = 0, 1 \end{cases}$$

따라서, 식 (25)과 식 (26)으로부터  $T_k(m, n)$ 과  $S_k(m, n)$  각각은 식 (23)의 데이터  $R_k$ 에 대한 DCT 수열 중 하나라는 것을 알 수 있다. 그러나 식 (25. d), (25. f), (26. d), (26. f)인 경우는  $(-1)^{q_i+i+1}$ 항으로 인해 약간의 수식 변경이 필요하다.

식 (22)으로부터  $i=1, 2$ 인 경우에  $q_i+i+1$ 은 짝수이고,  $i=0, 3$ 인 경우는  $q_i+i+1$  값이 홀수라는 점을 고려해 보자. 그러면 식 (25. d)와 식 (26. d)에 대한 입력 수열은  $\{-x_{02}, x_{13}, x_{20}, -x_{31}\}$ 이고, 식 (25. f)와 식 (26. f)에 대한 입력 수열은  $\{-x_{01}, x_{10}, x_{23}, -x_{32}\}$ 이다.

이 수열들은  $R_2^*$ 와  $R_3^*$ 로 정의해보자.

$$R_2^* = \{-x_{02}, x_{13}, x_{20}, -x_{31}\} \quad (27. a)$$

$$R_3^* = \{-x_{01}, x_{10}, x_{23}, -x_{32}\} \quad (27. b)$$

또한  $R_k$ 의 1차 DCT 수열을  $\{h_k^l : l = 0, 1, 2, 3\}$ 로,  $R_k^*$ 의 1차 DCT 수열을  $\{h_k^{*l} : l = 0, 1, 2, 3\}$ 으로 각각 정의해보자. 그러면 다음과 같은 식을 얻는다.

$$h_k^l = \sum_{i=0}^3 x_{i_v} \cos \frac{(2i+1)l}{8} \pi, \quad x_{i_v} \in R_k \quad (28. a)$$

$$h_k^{*l} = \sum_{i=0}^3 x_{i_v} \cos \frac{(2i+1)l}{8} \pi, \quad x_{i_v} \in R_{k^*} \quad (28. b)$$

$h'_k$  혹은  $h_k^*$  중 하나가  $T_k(m, n)$  혹은  $S_k(m, n)$ 와 등가인지를 알아보기 위해서 식 (25)와 식 (26)에  $m$ 과  $n$ 을 대입하는 것이 필요하다. 이해를 돕기 위해서 임의로 선택된  $(m, n)$ 에 대해  $T_k(m, n)$ 을 계산한 몇 가지 예를 제시한다.  $(m, n) = (1, 1)$ ,  $k=2$  인 경우,  $n$ 은 홀수이므로  $m$ 과  $n$ 의 값을 식 (25.d)에 대입해야 한다.

$$T_2(1, 1) = (-1)^0 \sum_{i=0}^3 (-1)^{q_i+i+1} x_{i(i+2-4q_i)} \cos \frac{(2i+1)(1+0-3)}{8} \pi$$

따라서, 다음과 같은 사실을 알 수 있다.

$$T_2(1, 1) = h_2^{2*} \quad (29. a)$$

$(m, n) = (3, 2)$ ,  $k=2$ 인 경우,  $n$ 은 짝수이므로,  $m$ 과  $n$ 의 값은 (25.c)식에 대입된다. 그러면 다음의 식을 얻을 수 있다.

$$T_2(3, 2) = (-1)^1 \sum_{i=0}^3 x_{i(i+2-4q_i)} \cos \frac{(2i+1)5}{8} \pi$$

이는 다음과 등가이다

$$T_2(3, 2) = \sum_{i=0}^3 x_{i(i+2-4q_i)} \cos \frac{(2i+1)3}{8} \pi$$

따라서,

$$T_2(3, 2) = h_2^3 \quad (29. b)$$

이와 같은 방법으로  $m, n = 0, 1, 2, 3$ 에 대해  $T_k(m, n)$ 와  $S_k(m, n)$  모두를 계산할 수 있으며, 결과는 그림 9에 나타내었다.

이러한 결과로부터 DCT수열  $\{y_{mn} : m, n = 0, 1, 2, 3\}$ 의 각각을 얻을 수 있다. 식 (18)과 식 (24)로부터,  $y_{mn}$ 을 다음과 같이 표현할 수 가 있다.

$$y_{mn} = \frac{1}{2} \left( \sum_{k=0}^3 T_k(m, n) + \sum_{k=0}^3 S_k(m, n) \right), \quad m, n = 0, 1, 2, 3 \quad (30)$$

그림 9로부터 다음의 결과를 얻을 수 있다.

$$y_{00} = h_0^0 + h_1^0 + h_2^0 + h_3^0 \quad (31. a)$$

$$y_{01} = h_0^1 - h_1^1 + h_2^{3*} - h_3^{3*} \quad (31. b)$$

$$y_{02} = h_0^2 + h_1^2 - h_2^2 - h_3^2 \quad (31. c)$$

$$y_{03} = h_0^3 - h_1^3 - h_2^{1*} + h_3^{1*} \quad (31. d)$$

$$y_{10} = h_0^1 + h_1^1 + h_2^1 + h_3^1 \quad (31. e)$$

$$y_{11} = \frac{1}{2}(h_0^2 - h_1^0 + h_2^{2*} + h_0^0 - h_1^2 - h_3^{2*}) \quad (31. f)$$

$$y_{12} = \frac{1}{2}(h_0^3 + h_1^1 - h_2^3 - h_3^1 + h_0^1 + h_1^3 - h_2^1 - h_3^3) \quad (31. g)$$

$$y_{13} = \frac{1}{2}(-h_1^2 - h_2^{0*} + h_3^{2*} + h_0^2 - h_2^{2*} + h_3^{0*}) \quad (31. h)$$

$$y_{20} = h_0^2 + h_1^2 + h_2^2 + h_3^2 \quad (31. i)$$

$$y_{21} = \frac{1}{2}(h_0^3 - h_1^1 + h_2^{1*} + h_3^{3*} + h_0^1 - h_1^3 - h_2^{3*} - h_3^{1*}) \quad (31. j)$$

$$y_{22} = \frac{1}{2}(h_1^0 - h_3^0 + h_0^0 - h_2^0) \quad (31. k)$$

$$y_{23} = \frac{1}{2}(-h_0^3 - h_1^1 - h_2^{1*} + h_3^{3*} + h_0^1 + h_1^3 - h_2^{3*} + h_3^{1*}) \quad (31. l)$$

$$y_{30} = h_0^3 + h_1^3 + h_2^3 + h_3^3 \quad (31. m)$$

$$y_{31} = \frac{1}{2}(-h_1^2 + h_2^{0*} + h_3^{2*} + h_0^2 - h_2^{2*} - h_3^{0*}) \quad (31. n)$$

$$y_{32} = \frac{1}{2}(-h_0^3 + h_1^1 + h_2^3 - h_3^1 + h_0^1 - h_1^3 - h_2^1 + h_3^3) \quad (31. o)$$

$$y_{33} = \frac{1}{2}(-h_0^2 - h_1^0 - h_2^{2*} + h_0^0 + h_1^2 + h_3^{2*}) \quad (31. p)$$

2차 DCT수열  $\{y_{mn} : m, n = 0, 1, 2, 3\}$ 이  $\{h_k^l, h_k^{l*} : l = 0, 1, 2, 3\}$ 항으로 표현되어지므로  $(h_k^l$ 과  $h_k^{l*}$ 항은 식 (24)와 식 (28)에 정의되어진 데로  $R_k$ 혹은  $R_k^*$ 에 대

한 1차 DCT수열) 4 point DCT로부터 4×4 DCT를 얻을 수 있다. 그림 10의 블록도는 4×4 DCT 계산을 위한 제안된 알고리즘을 나타낸다. 제안된 알고리즘은 4-point DCT 6개를 필요로 한다. 그러나, 제안된 알고리즘이 필요로 하는 1차 DCT의 수는  $T_k(m, n)$ 와  $S_k(m, n)$ 에 대한 Kernel을 조사함으로써 해서, 그리고 식 (31)을 체계적으로 계산함으로써 해서 4개로 감소된다. 식 (25)와 식 (26)에서, 주어진  $m$ 과  $n$ 에 대해  $T_0(m, n)$ 에 대한 Kernel은  $S_1(m, n)$ 에 대한 Kernel과 동가이고,  $T_1(m, n)$ 에 대한 Kernel은  $S_0(m, n)$ 에 대한 Kernel과 동가라는 사실을 알 수 있다. 따라서, 식 (31)에서  $h'_0$ 이 나타날 때는 언제든지  $h'_1$  혹은  $-h'_1$  또한 나타난다는 사실을 알 수 있다. 같은 방법으로  $T_2(m, n)$ ,  $T_3(m, n)$ ,  $S_2(m, n)$ ,  $S_3(m, n)$ 의 Kernel을 관찰해보면  $h'_2$ 은  $h'_3$ 과 항상 함께 나타나고,  $h'_2$ 은  $-h'_3$ 과 항상 함께 나타난다는 사실을 알 수 있다.

$T_0(m, n)$

$n \rightarrow$	$h_0^0$	$h_0^1$	$h_0^2$	$h_0^3$
$\downarrow$	$h_0^1$	$h_0^2$	$h_0^3$	0
$\downarrow$	$h_0^2$	$h_0^3$	0	$-h_0^3$
$\downarrow$	$h_0^3$	0	$-h_0^3$	$-h_0^2$

$T_1(m, n)$

$n \rightarrow$	$h_1^0$	$-h_1^1$	$h_1^2$	$-h_1^3$
$\downarrow$	$h_1^1$	$-h_1^0$	$h_1^1$	$-h_1^2$
$\downarrow$	$h_1^2$	$-h_1^1$	$h_1^0$	$-h_1^1$
$\downarrow$	$h_1^3$	$-h_1^2$	$h_1^1$	$-h_1^0$

$T_2(m, n)$

$n \rightarrow$	$h_2^2$	$h_2^{3*}$	$-h_2^2$	$-h_2^{1*}$
$\downarrow$	$-h_2^1$	$h_2^{2*}$	$-h_2^3$	$-h_2^{0*}$
$\downarrow$	$h_2^2$	$h_2^{1*}$	0	$-h_2^{1*}$
$\downarrow$	$h_2^3$	$h_2^{0*}$	$h_2^3$	$-h_2^{2*}$

$T_3(m, n)$

$n \rightarrow$	$h_3^0$	$-h_3^{3*}$	$-h_3^2$	$h_3^{1*}$
$\downarrow$	$h_3^1$	0	$-h_3^1$	$h_3^{2*}$
$\downarrow$	$h_3^2$	$h_3^{3*}$	$-h_3^0$	$h_3^{3*}$
$\downarrow$	$h_3^3$	$h_3^{2*}$	$-h_3^1$	0

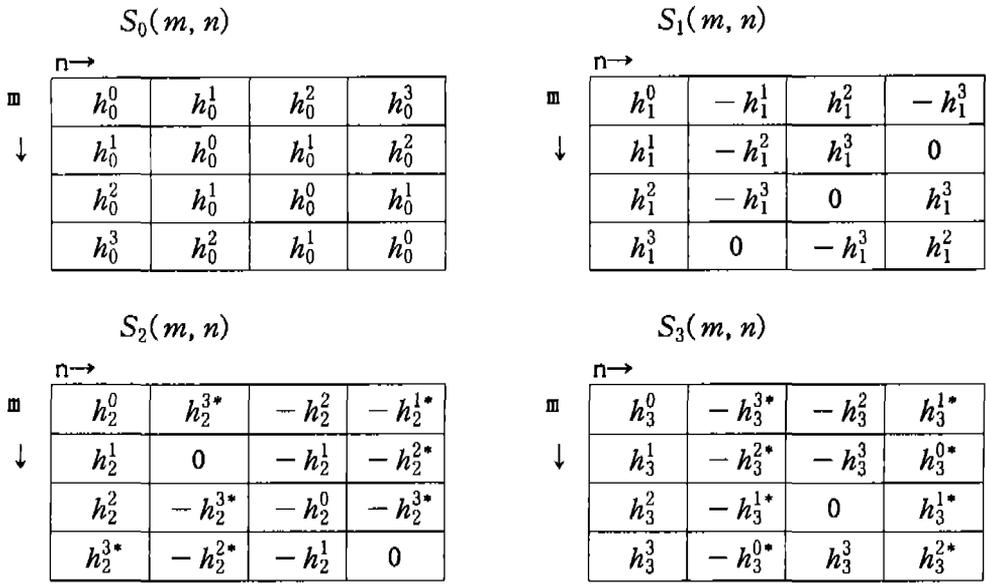


Fig 9.  $T_k(m, n)$  and  $S_k(m, n)$  for each  $m$  and  $n$

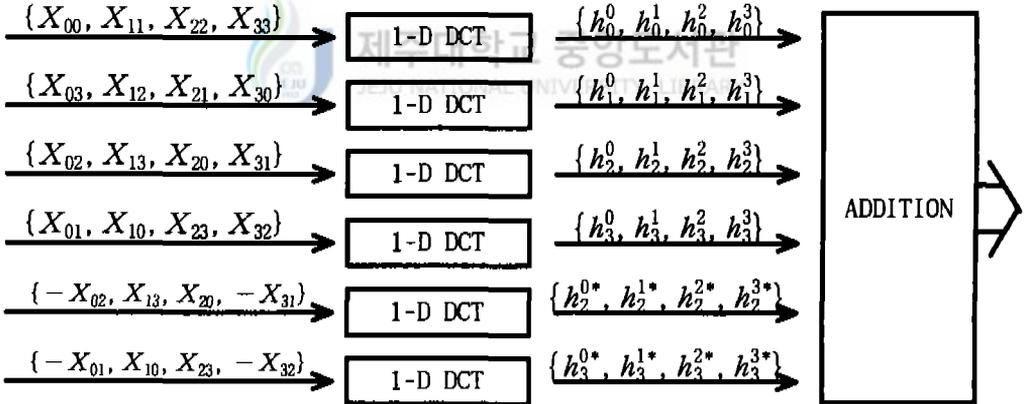


Fig 10. The block diagram of the proposed  $4 \times 4$  DCT

### 3. Butterfly 변환

식 (31)에서 합은 그림 11에서 보여지는 흐름도에 의해 수행될 수 있는데, 이는

원래의 고속 변환 butterfly 계산과 유사하다. 다음은  $4 \times 4$  DCT 알고리즘에 대한 완전한 신호 흐름도를 유도하고자 한다.

그림 11에서 다음을 알 수 있다.

$$\{a_8, a_9, a_{10}, a_{11}\} = \{h_2^0 + h_3^0, h_2^1 + h_3^1, h_2^2 + h_3^2, h_2^3 + h_3^3\} \quad (32. a)$$

$$\{a_{12}, a_{13}, a_{14}, a_{15}\} = \{h_1^{0*} - h_3^{0*}, h_2^{1*} - h_3^{1*}, h_2^{2*} - h_3^{2*}, h_2^{3*} - h_3^{3*}\} \quad (32. b)$$

그러나, 식 (32)는 다음과 동가이다.

$$\{a_8, a_9, a_{10}, a_{11}\} = DCT\{x_{02} + x_{01}, x_{13} + x_{10}, x_{20} + x_{23}, x_{31} + x_{32}\} \quad (33. a)$$

$$\{a_{12}, a_{13}, a_{14}, a_{15}\} = DCT\{-x_{02} + x_{01}, x_{13} - x_{10}, x_{20} - x_{23}, -x_{31} + x_{32}\} \quad (33. b)$$

여기서  $DCT\{\cdot\}$ 는 중괄호에 있는 데이터들의 4 point DCT를 의미한다. 그러므로 식 (33)은 그림 11에서  $a_8$ 에서  $a_{15}$ 까지 계산을 위해  $R_2$ 와  $R_3$ 의 사전 계산이 필요하며, 두 개의 1차 DCT를 요한다. 그림 11에 나타난 계산을 그림 10의 블록도에 삽입하고 식 (33)에 있는 관계식을 이용하면  $4 \times 4$  DCT의 계산을 하기 위한 완성된 신호 흐름도를 그림 12와 같이 나타낼 수 있다. Lee 혹은 Hou와 같은 고속 1차 DCT 알고리즘들은 그림 12에서 보여지는 4 point DCT를 계산하기 위해 사용될 수 있다. Lee와 Hou의 알고리즘이 이용될 때, 1차 DCT 알고리즘과 관련하여 특별히 데이터 치환이 필요하다. 그러나 여기서는 설명을 수월하게 하기 위해 생략한다. 또한 이 알고리즘이 더 큰 길이의 2차 DCT를 수행해야 할 때면, 입력 데이터와 출력 데이터의 재 정렬이 고려되어야 한다.  $4 \times 4$  DCT를 구하기 위해 Lee와 Hou의 알고리즘을 이용하면 제안된  $4 \times 4$  DCT알고리즘은 오로지 16개의 곱셈만을 필요로 한다는 사실을 알 수 있다.

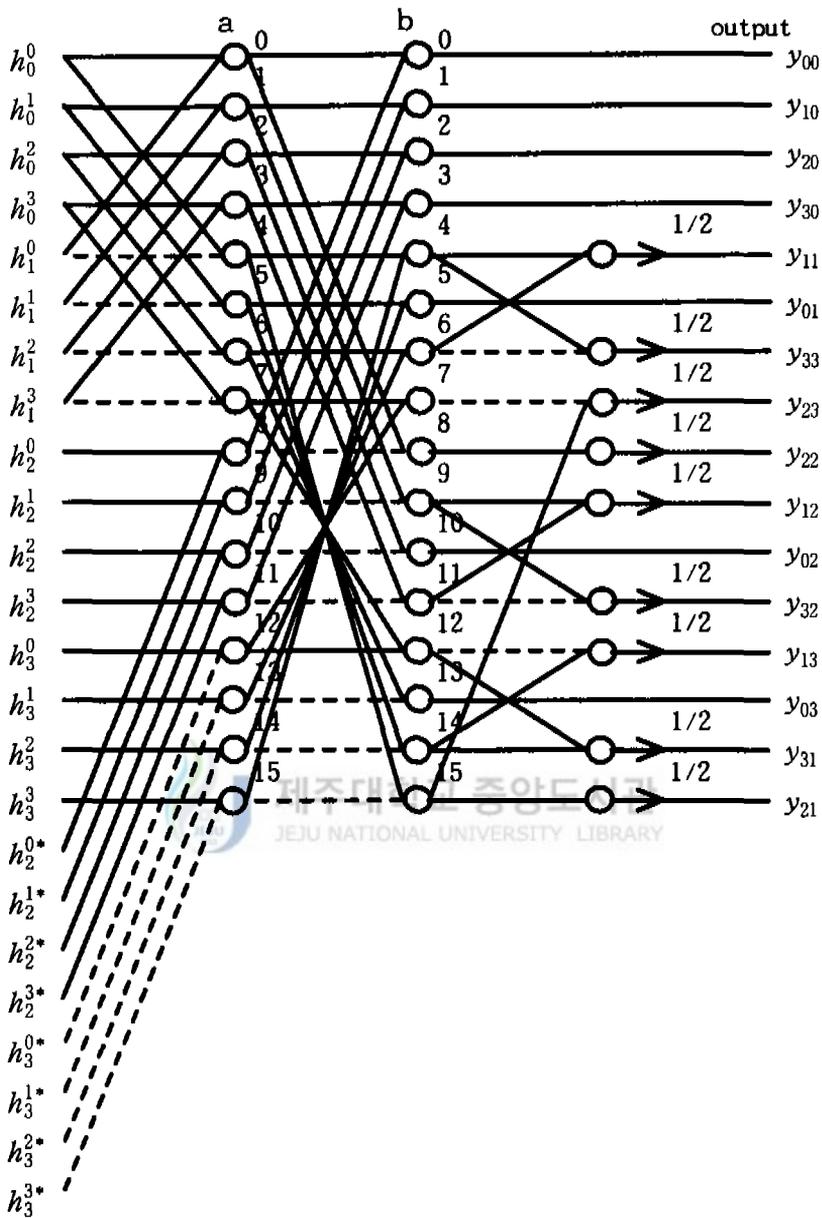


Fig 11. Signal flow graph. Broken lines represents transfer factors -1, full lines represents unity transfer factor,  $\circ$  represents adders and  $\rightarrow$  with  $1/2$  represents multiplication by  $1/2$ .

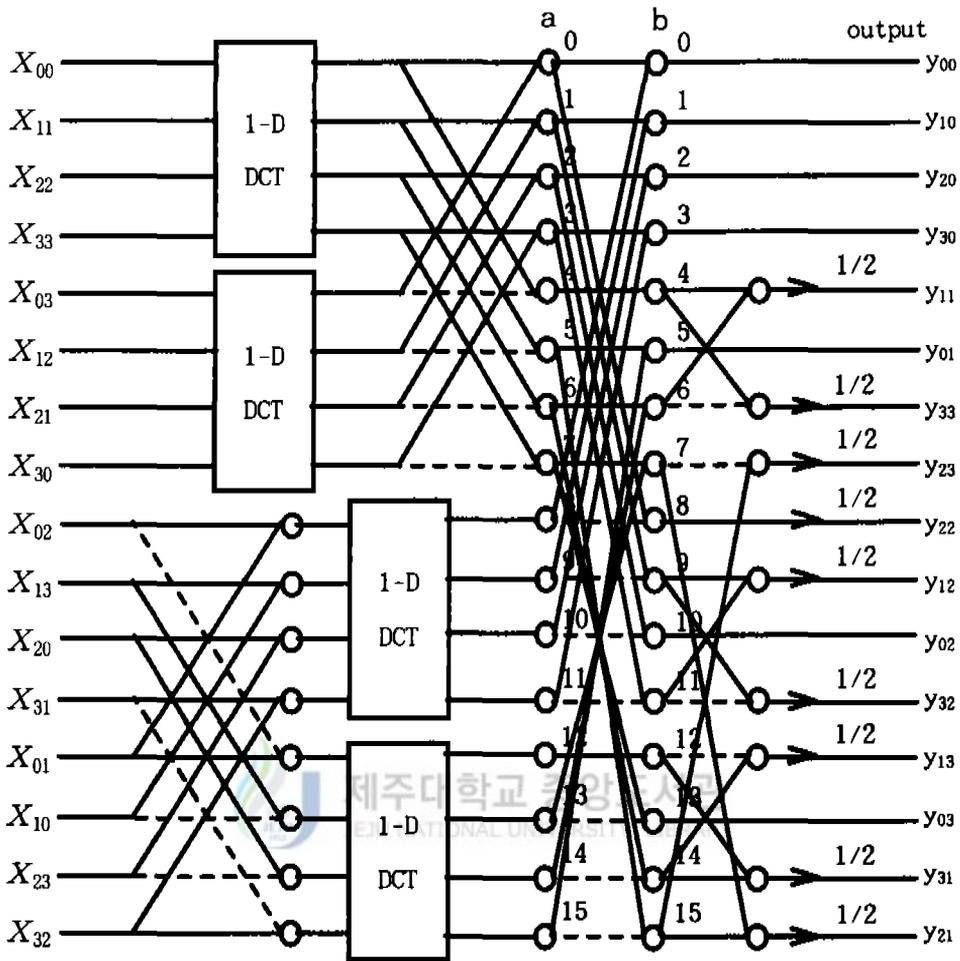


Fig 12. Signal flow graph for the proposed  $4 \times 4$  DCT

덧셈의 수는 74라는 것 또한 쉽게 알 수 있다. 이제 간략하게 DCT 역변환(IDCT)을 설명해보자. DCT 행렬의 직교성 때문에 IDCT는 scaling 요소가 고려되지 않는다면 전 방향 변환을 구하는 신호 흐름도를 인버팅함으로써 얻어진다. 따라서, IDCT 알고리즘의 신호 흐름도는 그림 12의 인버스(역)이며, 그 흐름도에서 1차 DCT(1-D DCT)가 1-D IDCT로 바뀌면 된다.

## IV. 실험 결과 및 비교 검토

이 장에서는, 반복적인 2차 DCT 알고리즘을 사용하여  $2^m \times 2^m$  DCT를 계산하기 위해 필요한 곱셈과 덧셈의 수를 기본적 모듈로 제안한 고속  $4 \times 4$  DCT 알고리즘과 관련지어 검토하였다. 뿐만 아니라, 크기가 qcif(176×144)인 실제 영상 데이터를 이용하여 직접 DCT한 결과와 제안한 알고리즘을 이용해 DCT한 결과를 비교 분석하였다. 그림 14는 시뮬레이션에서 이용한 실제 영상 데이터이다.

$2^m \times 2^m$  DCT를 계산하기 위해 필요한 덧셈의 수를  $A_m$ 이라 하고 곱셈의 수는  $M_m$ 이라 할 때  $M_m$ 가 제안한 반복적 2차 DCT 알고리즘으로부터 다음을 알 수 있다.

$$M_m = 3 \times \left(\frac{N}{2}\right)^2 + 4 \times M_{m-1} \quad (34. a)$$

$$A_m = 2 \times N^2 + 4 \times \left[ A_{m-1} + \frac{N}{2} \times \left(\frac{N}{2} - 1\right) \right], \quad m \geq 3 \quad (34. b)$$

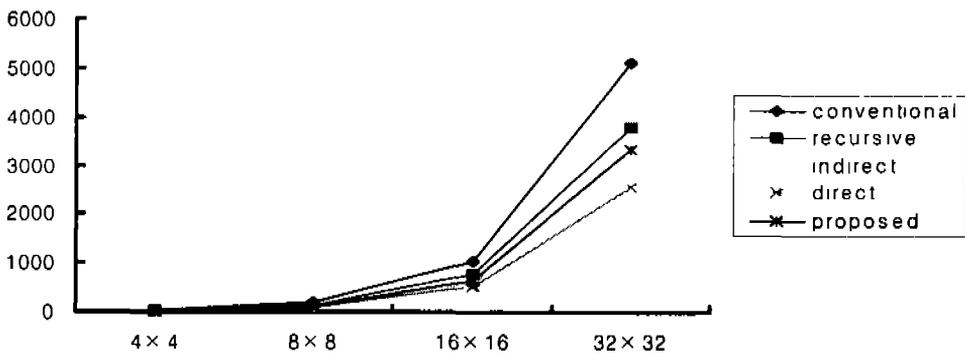
$m=2$ 인 경우, 제안된 알고리즘인 경우는  $M_2=16$ 인 데, 반복적 2차 DCT 알고리즘은  $M_2=24$ 이다. 또한 제안된 알고리즘인 경우  $A_2=74$ 이고 반복적 2차 DCT 알고리즘은  $A_2=74$ 이다. 따라서, 식 (34)에 따른 계산으로 다른 고속 2차 알고리즘인 반복적 2차 DCT 알고리즘과 간접 다항 변환 알고리즘 그리고 직접 다항 변환 알고리즘에서 필요로 하는 덧셈과 곱셈의 수와 비교할 수 있다. 그 결과를 요약하면 표 1과 같다.

Table 1. Comparison of the number of multiplications and additions

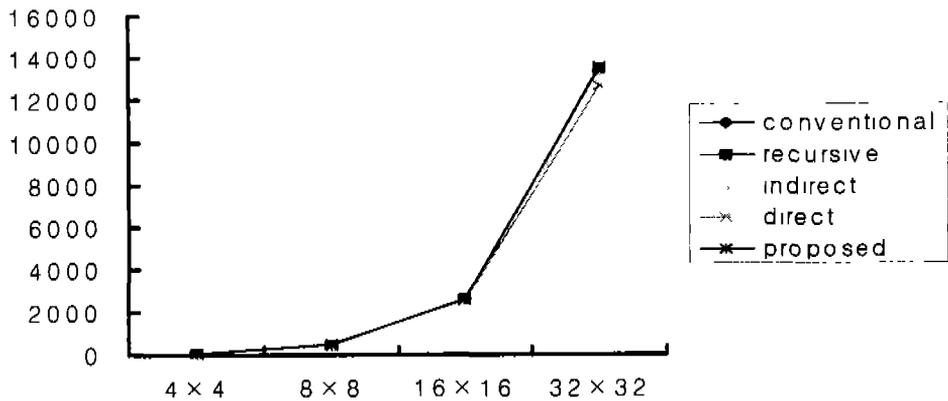
		Conventional Algorithm	Other Fast Algorithms			Proposed Algorithm
			Recursive 2-D DCT	Indirect Polynomial Transform	Direct Polynomial Transform	
Number of Multiplications	4×4	32	24	16	16	16
	8×8	192	144	104	96	112
	16×16	1024	768	568	512	640
	32×32	5120	3804	2840	2560	3328
Number of Additions	4×4	72	72	70	68	74
	8×8	464	464	462	484	472
	16×16	2592	2592	2558	2531	2624
	32×32	13376	13376	12950	12578	13504

표 1에서 알 수 있는 바와 같이 제안된 4×4 DCT 알고리즘에 기초를 둔 반복적인 접근은 원래의 반복적 2차 DCT 알고리즘보다 곱의 수를 더 많이 줄일 수 있었다.

그림 13은 표 1의 데이터를 기초로 해서 각 알고리즘에서 필요로 하는 곱셈의 수 및 덧셈의 수를 그래프로 표현하여 비교한 것이다. 그래프로부터 덧셈의 수에는 영향을 미치지 않으면서 곱셈의 수에 있어서 만큼은 대폭 개선되었음을 재확인할 수가 있다. 비록 제안된 알고리즘이 간접 다항 변환 알고리즘과 직접 다항 변환 알고리즘보다는 곱의 수를 더 많이 필요로 하고 있지만 구조가 더 규칙적이고 실수 연산만을 한다는 장점이 있다. 간접 다항 변환 알고리즘과 직접 다항 변환 알고리즘은 복소수 계산 구조를 필요로 한다.



(a)



(b)

Fig 13. Graph of the number of (a) multiplication and (b) additions



(a) carphone



(b) claire



(c) container

Fig 14. Original image

Table 2. (a) The sampled  $4 \times 4$  block data of Fig 14 (b) the result value of direct  $4 \times 4$  DCT (c) the result value of proposed fast DCT

	123	123	121	122
	120	121	121	122
	118	120	121	121
	120	121	122	119
	(a)			
1935.000000	-2.771639	-3.535534	-1.148050	
7.997890	1.560660	3.807604	-2.767767	
4.949747	5.114166	-0.500000	0.587623	
-1.016734	-0.767767	0.046427	-0.560660	
	(b)			
1935.000000	-2.771639	-3.535534	-1.148050	
7.997890	1.560660	3.807604	-2.767767	
4.949747	5.114166	-0.500000	0.587623	
-1.016734	-0.767767	0.046427	-0.560660	
	(c)			

표 2는 그림 14 (a)의 실 영상 데이터로부터 임의로 추출한  $4 \times 4$  블록을 직접 DCT한 결과와 제안된 고속 DCT 알고리즘을 이용해서 구한 DCT 결과를 나타낸 것이다. 표 2의 (b)와 (c)를 살펴보면 DCT의 결과가 일치함을 확인할 수가 있는데, 이는 기존의 압축률을 그대로 유지하면서 곱셈의 수를 줄인 것이다. 한편, 그림 14의 데이터를 DCT 하는데 걸리는 시간을 체크한 결과 직접 DCT한 경우는 50m sec의

시간이 걸렸고 본 알고리즘을 적용한 경우는 20m sec의 시간이 걸렸다. 따라서 곱의 수가 줄어들어서 처리 속도가 기존 알고리즘보다 개선되었음을 확인할 수가 있었다.

그림 15는 그림 14의 원 영상을 기존의 알고리즘인 직접 DCT를 이용하여 복원시킨 영상이고 그림 16은 그림 14의 원 영상을 제안한 알고리즘을 이용하여 DCT한 후 복원시킨 영상이다.



(a) carphone (b) claire (c) container

Fig 15. Reconstructed image of direct  $4 \times 4$  DCT



(a) carphone (b) claire (c) container

Fig 16. Reconstructed image of proposed  $4 \times 4$  DCT

표 2에서도 알 수 있었듯이 그림 15와 그림 16을 보면 직접 DCT를 수행했을 때와 제안된 알고리즘으로 DCT를 수행했을 때의 복원된 영상에는 차이가 없음을 확인할 수 있다.



## V. 결론

본 논문에서는  $4 \times 4$  DCT를  $4 \times 1$  DCT 4개로 분할해서 효율적인  $4 \times 4$  DCT 알고리즘을 제안하였다. 제안된 알고리즘을 이용하여  $4 \times 4$  DCT 계산을 하는 데 필요한 곱의 수는 16개로 기존의 알고리즘에 비해 40%의 감소를 가져왔으며, CPU 처리 시간을 체크한 결과 50m sec에서 20m sec로 줄이는 효과를 가져왔다. 따라서 제안된 알고리즘은 기존 알고리즘보다 곱의 수를 줄임으로써 처리 속도를 대폭 개선하였다. 그러므로 제안된  $4 \times 4$  DCT 알고리즘을 반복적 2차 DCT 알고리즘과 결합하면  $2^n \times 2^n$  DCT를 계산하는 효율적인 알고리즘이 될 수 있다. 본 알고리즘의 구조가 다항 변환에 바탕을 둔 접근법에 비해 대단히 규칙적이다. 그 알고리즘은  $4 \times 4$  DCT로 제한되지 않고, 약간의 수정을 거치면  $2^n \times 2^n$  DCT로 확장되어질 수 있다. 그래서 곱의 수는 재래적인 알고리즘의 50%가 되며, 이는 Duhamel의 직접 다항 변환 알고리즘의 곱의 수와 같다. 또한, 일반화된 알고리즘은 규칙적인 구조를 갖는다는 사실을 알 수가 있으며, 이는 VLSI 수행에 적절하다.

추후 연구 방향으로서는 본 알고리즘의 병렬성을 활용하여 이를 DSP상에서 파이프라인 구조를 이용하여 계산 시간의 단축을 도모하는 것이다.

## 참고문헌

- B. G. Lee, Dec. 1984, "A new algorithm to compute the discrete cosine transform", *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp.1243-1245.
- C. Ma, 1988, "A fast recursive two-dimensional cosine transform," *Proc. SPIE Int. Soc. Opt. Eng.*, pp.541-548.
- F. A. Kamagar and K. R. Rao, Sept. 1982, "Fast algorithms for the 2-D discrete cosine transform," *IEEE Trans. Comput.*, vol.C-31, pp.899-906.
- H. S. Hou, Oct. 1987, "A fast recursive algorithm for computing the discrete cosine transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, pp. 1455-1461.
- I-Ming Pao, Ming-Ting Sun, June 1999, "Modeling DCT coefficients for fast video encoding," *IEEE Trans.*, vol 9, pp.608-616.
- H. R. Wu and F. J. Paoloni, 1989, "A two-dimensional fast cosine transform algorithm-a structural approach," in *Proc. IEEE ICIP'89.*, pp.50-54.
- J. Makhoul, Feb.1980 "A fast cosine transform in one and two dimensions," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-28, pp.27-34.
- M. A. Haque, Dec. 1985, "A two-dimensional fast cosine transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-33, pp.1532-1539.
- M. D. Wagh and H. Ganesh, Apr. 1980, "A new algorithm for the discrete cosine transform of arbitrary number of points," *IEEE Trans. Comput.*, vol. C-29, pp. 269-277.
- M. Vetterli, Mar. 1985, "Fast 2-D discrete cosine transform," in *Proc. ICASSP-85*, pp.1538-1541.
- N. Ahmed and K.R.Rao., 1975 *Orthogonal Transforms for Digital Signal Processing*. Springer-Verlag.

N. I. Cho and S. U. Lee, Jan. 1990, "DCT algorithms for VLSI parallel implementation," *IEEE Trans. Acoust., Speech, signal Processing*, vol.38, pp.121-127.

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

W. H. Chen, C. H. Smith, and S. c. Fralick, Nov.1977, "A fast computational algorithm for discrete cosine transform," *IEEE Trans. Commun.*, vol. COM-25, pp. 1004-1009.



## 본 논문과 관련된 저자의 발표 논문

### 1. 학술대회 논문집

- 함영희, 임재운, “DCT 영상 코딩에서 블록화 현상 감소를 위한 레벨 조정 알고리즘에 관한 연구”, 대한전자공학회 한국통신학회 제주대학교공과대학 산업기술연구회 합동학술발표 논문집, pp.13-17, 1996.



## 감사의 글

본 논문이 완성되어 결실을 이룰 수 있도록 세심한 배려와 관심으로 지도해 주신 임재운 교수님께 진심으로 감사를 드리며, 논문을 잘 마무리할 수 있도록 조언과 격려를 아끼지 않으신 이용학 교수님, 문건 교수님, 김홍수 교수님, 양두영 교수님, 강진식 교수님께도 감사를 드립니다. 그리고 언제나 능력 이상으로 나를 인정해 주시고 말없이 지켜봐 주신 한국통신의 오영웅 부장님께 감사를 드리며, 인터넷으로 연을 맺게된 김순국 선배님과 이 기쁨을 함께 하고자 합니다.

학부시절서부터 직장생활에 이르기까지 매사에 힘이 되어 주신 신상훈 선배님과 나를 무척이나 아껴주시는 한국 통신의 김동식 대리님은 물론, 우리 연구실을 말없이 곳곳하게 지켜나가는 그러면서도 힘들다는 소리 한번 한 적 없는 후배 향진과 형준에게 고마움을 전하며, 이제 졸업이 얼마 남지 않은 지연과 연구실의 모든 후배들에게도 고마운 마음을 전합니다.

오늘의 내가 있기까지 헌신적인 사랑으로 보살펴 주시고 본 논문을 쓰는 동안 줄곧 따뜻한 격려와 지원을 서슴지 않으신 아버지, 어머니 감사합니다. 그리고 사랑합니다. 내가 힘들어 할 때면 언제나 따뜻한 위안을 주었던 오빠 내외분, 경덕, 정희와 이 기쁨을 함께 하렵니다.

끝으로 학위 과정 내내 힘들고 어려울 때 줄곧 옆에서 나를 지켜주고 격려를 아끼지 않았던 나의 사랑하는 친구이면서 평생 동지인 동현씨와 이 기쁨을 함께 하고 싶습니다.