2006년 2월
석사학위논문

Chen-DCT 알고리즘을 이용한 H.264 Encoder 성능 개선에 관한 연구

조 선 대 학 교 대 학 원

컴 퓨 터 공 학 과

고 은 혜

Chen-DCT 알고리즘을 이용한 H.264 Encoder 성능 개선에 관한 연구

A Study of Enhanced H.264 Encoder using Chen-DCT Algorithm

2006년 2월 일

조 선 대 학 교 대 학 원

컴 퓨 터 공 학 과

고 은 혜

Chen-DCT 알고리즘을 이용한 H.264 Encoder 성능 개선에 관한 연구

지도교수 김 충 원

이 논문을 공학석사학위신청 논문으로 제출함.

2006년 10월 일

조 선 대 학 교 대 학 원

컴 퓨 터 공 학 과

고 은 혜

고은혜의 공학석사 학위논문을 인준함

위 원 장 **조선대학교** 교수 _____ 印

위 원 **조선대학교** 교수 _____ 印

위 원 **조선대학교** 교수 _____ 印

*2005*年 *11*月

조선대학교 대학원

목 차

List of Titles

표 목 차

도 목 차

ABSTRACT

A Study of Enhanced H.264 Encoder using Chen-DCT Algorithm

Ko, Eun-Hye
Advisor : Prof. Kim, Choongwon, Ph. D.
Dept. of Computer Engineering
Graduate School of Chosun University

H.264 is standards for the coded representation of visual information. H.264/AVC is newest video coding standard of the ITU-T Video Coding Experts Group and the ISO/IEC Moving Picture Experts Group. Most important feature of H.264 standard include compression efficiency, transmission efficiency and a focus on popular applications of video compression. Only three profiles are currently supported, each targeted at a class of popular video communication applications. H.264 standard assume a CODEC 'model' that uses block-based motion compensation, transform, quantization and entropy coding.

The purpose of the transform stage in video CODEC is to convert image or motion-compensated residual data into another domain. H.264 standard use DCT algorithm for transform stage. DCT is one of the most computational intensive parts of recent high quality digital Video coding techniques.

In this paper, we propose an efficient DCT calculation algorithm for fast video encoding. In the proposes an implementation of DCT using a fast DCT algorithm with shift and addition operations instead of multiplications. The Chen-DCT which is the most commonly used among fast DCT algorithm has less multiplication and additions than standard DCT algorithm.

And a method for reduce process-time has been an architecture of fast 2-D DCT processor using only 1-D DCT module. In thesis, video compression process time be reduced and unnatural distortion cannot be recognized.

제 *1* 장 서 론

디지털 영상 압축은 많은 양의 디지털 데이터로 이루어진 멀티미디어 정보의 전송 또는 저장 시 전송시간과 기억 용량의 제한을 극복하기 위해 화질을 용도에 따라 요구되는 수준으로 유지하면서, 가능한 적은 비트 수로 정보를 기록·재생하는데 그 주안점을 둔 기술이다 [1].

디지털 영상 압축 기술은 디지털 신호처리, 디지털 통신, 반도체, 컴퓨터 등의 기술 발전에 힘입어 멀티미디어 TV, 주문형 비디오(VOD), HDTV, 디지털 방송, DVD 등 여러 가지 상품의 형태로 소비자들에게 선보이고 있다. 이와 같은 대용량 디지털 멀티미디어는 통신 및 네트워크 기술의 발전으로 멀티미디어 데이터의 전송 및 저장에 대한 수요가 크게 증가하고 있다. 그런데 사용자들은 전송 대역폭이 불충분한 환경에서 서비스를 받거나 가능한 화질을 유지하면서 적은 용량으로 비디오를 저장하기를 원하므로 주어진 환경에 맞추어 멀티미디어데이터의 크기를 줄여야 할 필요가 생긴다. 멀티미디어 데이터의 크기를 줄이는 방법은 여러 가지가 있지만, 변환계수의 재양자화가 일반적이다. 그렇지만 대부분의 고부가가치 컨텐츠가 디지털TV나 DVD로 제작되기 때문에 H.264/AVC와 같은 저용량에 초점을 맞춘 압축방식을 사용할 필요가 있다 [1], [2], [3].

H.264/AVC(이하 H.264)는 국제통신연맹(ITU)과 국제 표준화 기구(ISO)·국제전자기술위원회(IEC)가 공동 결성한 JVT(Joint Video Team)에 의해 개발되어 최근 표준권고안이 확정되었다[8], [9]. H.264는 기존의 동영상 압축 국제 표준으로 널리 사용되는 H.261/H.263과 MPEG-1/2/4에 비하여 새로운 요소들과 방식들을 채택하였고, 그 결과 파일 압축 기능이 기존에 비하여 매우 우수하면서, 제한된 회선으로 보다 많은 데이터를 고속으로 전송할 수 있다.

기존의 동영상 압축 과정에서 소요되는 계산량을 감소시키기 위한 다양한 연

구들이 국내·외적으로 시도되어 왔다. 지금까지 발표된 연구들은 움직임 추정(Motion Estimation) 과정에서의 속도 개선과 DCT(Discrete Cosine Transform) 및 양자화 과정에서의 계산량 감소로 크게 나눌 수 있다 [4], [5], [6], [7]. 일반적으로 DCT를 구현하는 방법은 FFT처럼 successive dicimation 기법을 사용하는 FDCT(Fast DCT), FFT를 이용하는 FFT에 기반한 DCT, 그리고 계수별 순환 연산식을 사용하는 순환연산 DCT로 분류할 수 있다 [19].

본 논문에서는 부호화 과정의 DCT의 속도 향상을 위하여 FDCT변환 방식 중 Chen-DCT알고리즘을 이용한 DCT방식을 제안한다. 본 논문의 구성은 서론에 이어 2장에서는 H.264 동영상 압축과 압축의 과정을 설명하였고, 3장에서는 신호변환 방식인 DCT방식에 대하여 설명하고, 고속 DCT알고리즘 중 Lee's DCT와 Chen-DCT 방식에 대하여 설명하였다. 4장에서는 본 논문에서 제안하는 알고리즘 방식인 Chen-DCT 방식으로 실험하여 성능을 평가하였고, 5장에서 결론을 맺는다.

제 *2* 장 *H.264* 동영상 압축의 개요

압축(compression)이란 데이터를 보다 적은 수의 비트로 만드는 것이다. 압축은 중복요소를 제거함으로써 행해지는데 많은 유형의 데이터가 통계적으로 중복요소를 가지고 있으면 무손실 압축이나 손실압축 방식을 이용하여 압축될 수 있다. 무손실 압축의 경우 디코더에서 출력되는 복원 데이터가 원본 데이터와 정확히 일치한다. 하지만, 무손실 방법으로 이미지나 비디오를 압축 할 경우 보통 수준의 압축결과밖에 얻을 수가 없다. 손실 압축 시스템에서는 복원된 데이터가 원본 데이터와 동일하지는 않지만, 영상의 화질을 저하시킴으로써 훨씬 높은 압축효율을 얻을 수 있다. 손실 비디오 압축 시스템은 주관적인 중복요소를 제거하는 원리를 기본으로 한다. 주관적인 중복요소란, 보는 사람이 직관적으로 느낄 수 있는 화질에 큰 영향을 주지 않으면서 제거할 수 있는 요소를 말한다.

비디오 코딩이란 디지털 비디오 신호를 압축하고 복원하는 것을 의미하고, 시간적(temporal) 영역, 공간적(spatial) 영역내의 중복성(redundancy) 을 제거하는 원리를 기본으로 한다. [1]

본 장의 1절에서는 H.264 동영상 압축에 대하여 설명하고, 2절에서는 H.264 동영상의 부호화 과정에 대하여 상세하게 알아보고자 한다.

제 *1*절. *H.264* 동영상 압축

MPEG(Moving Picture Experts Group)과 ITU-T(International Telecommunication Union)이 연구 그룹인 VCEG(Video Coding Experts Group)는 초기 MPEG-4와

H.263 표준안보다 우수하고 뛰어난 비디오 이미지 압축 성능을 가진 표준안을 개발하였다. 새로운 표준안은 “Advanced Video Coding(AVC)″로 이름이 붙여졌고, MPEG-4 part 10 과 ITU-T Recommendation H.264로 공동 발표 되었다 [8], [9], [27].

H.264는 기본적으로 직사각형 비디오 프레임의 효과적이고 강력한 압축 및 전송을 지원하기 위해 설계되었다. H.264의 응용분야에는 양방향 비디오 통신, 방송 또는 고화질 비디오를 위한 압축 그리고 패킷 네트워크를 통한 비디오 스트리밍이 포함된다. 표준안에는 네트워크를 통한 강력한 전송을 지원하는 기능이 내장되었고, 가능한 한 넓은 범위의 프로세서 플랫폼에 구현할 수 있도록 설계되었다.

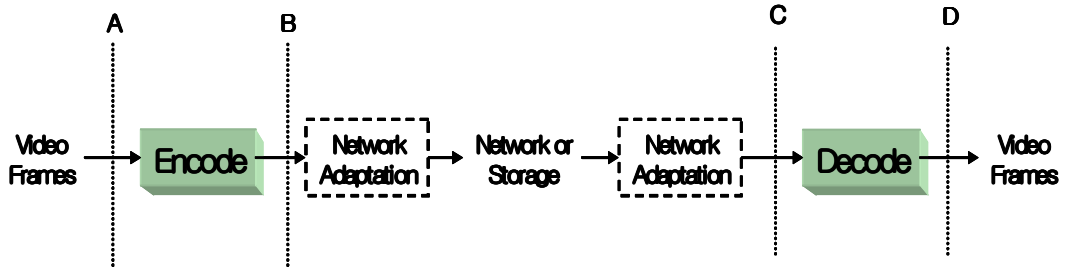


그림 *2-1. H.264* 동영상 압축 인터페이스

H.264는 특정한 기능을 지원하는 세 개의 프로파일이 정의되어 있는데, 각 프로파일은 비디오 통신 응용분야를 목표로 하고 있다. Baseline 프로파일은 화상회의와 같은 '대화를 위한' 응용분야에 특히 유용하고, Extended 프로파일은 '네트워크를 통한 비디오 스트리밍에 유용한 추가적인 도구'들을 포함하고 있으며, Main 프로파일은 비디오 방송 및 저장과 같은 '소비자 응용 제품에 적합한 도구'들을 포함하고 있다 [1].

H.264의 비디오 포맷은 4:2:0 순차주사 또는 비월주사 비디오의 인코딩과 디코딩을 지원한다. 즉, 기본 샘플링 포맷에서 색차샘플은 수평으로 2번째 휘도 샘플마다 위치하며, 수직으로 2개의 휘도 샘플 사이에 위치한다. 4:2:0 샘플링은 화상회의, 디지털 TV, DVD 등의 응용제품에 널리 이용된다. 4:2:0 비월 주사 비디오 영상에서 전체 비디오 프레임에 해당하는 Y, C_b, C_r 샘플은 두 개의 필드에 분배된다.

H.264의 기본적인 개념 자체는 H.263과 MPEG-4와 유사하나, 세부적인 내부 구현에 있어 상당 부분 변경된 방식을 채택하고 있다.

움직임 추정/보상(Motion Estimation/Compensation)의 경우를 살펴보면 H.264는 H.263이 모든 블록의 크기를 동일하게 하여 제공했던 것과는 달리 더 작은 블록사이즈, 1/4화소의 섬세한 화소 정밀도로 움직임 보상을 한다. 또 복수의 참조 프레임 중에서 최적한 것을 선택하여 움직임 보상에 사용한다.

H.264 비디오 압축은 동일한 대역폭에서도 향상된 화질을 보장하며 종전 대역폭의 절반 속도에서도 동일한 화질을 제공한다. H.264 압축 알고리즘의 또 다른 이점은 네트워크 에러 시 향상된 성능을 볼 수 있다는 점이다. 즉, 네트워크 에러 때문에 화상 데이터에 손실이 있을 때에도 화상이 완전히 끊어지는 대신 화질의 약화만 가져온다는 것이다.

DCT를 이용하는 화상부호화 방식의 결점이었던 블록 경계의 왜곡을 억제하는 필터를 사용하여 시각적인 왜곡이 쉽게 눈에 띄지 않도록 시각적인 화질 열화를 억제한다 [10], [11].

제 2절. H.264 동영상 압축의 부호화 과정

비디오 인코더는 시간적 모델, 공간적 모델, 엔트로피 인코더의 세 개의 주요 기능 부분으로 구성되어 있다. 시간적 모델은 인접하는 비디오 프레임 사이의 유사성을 이용하여 시간적 중복 요소를 제거하는데, 일반적으로 현재 비디오 프레임을 예측하는 방법을 사용한다. 공간적 모델은 오차 프레임 내의 인접 샘플 사이의 유사성을 이용하여 공간적 중복요소를 제거한다. H.264에서는 오차 샘플에 대해 변환을 수행하고, 그 결과를 양자화 함으로써 공간적 중복 요소를 제거한다. 시간적 모델의 파라미터 즉, 움직임 벡터와 공간적 모델의 파라미터인 계수는 엔트로피 인코더에 의해 압축된다. 엔트로피 인코더는 데이터에 존재하는 통계적인 중복 요소를 제거하며, 전송하거나 저장할 수 있는 압축된 비트스트림 또는 압축된 파일을 생성한다. 압축된 데이터는 코딩된 움직임 벡터 파라미터, 코딩된 오차계수 그리고 헤더 정보로 구성되어 있다 [1].

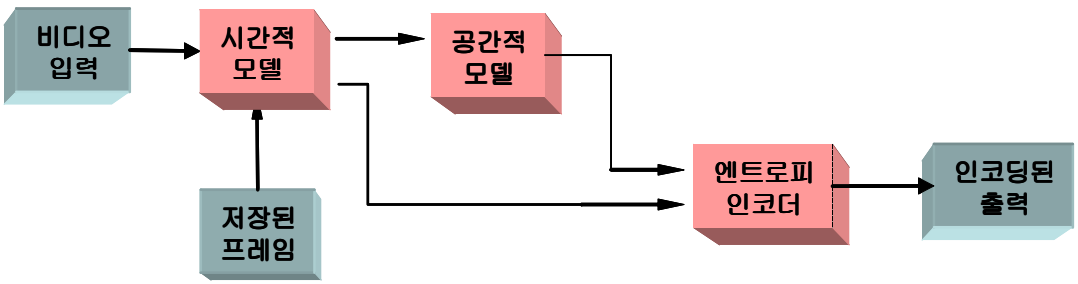


그림 2-2. 비디오 인코더 블록 다이어그램

일반적으로 표준안에 호환되는 인코더는 그림 2-3에 나타난 기능 요소들을 포함할 것이다. 블록현상 제거 필터를 제외하면, 대부분의 기본적인 기능 요소들은 이전의 표준안들에서도 존재하지만, H.264의 중요한 변화는 각 기능 블록의 세부적인 부분에서 일어난다.

입력영상 F_n은 현재 입력되는 영상으로 매크로블록 단위로 분할되고, 각 매크로 블록은 현재 프레임의 정보를 이용하는 화면 내 부호화 또는 참조 영상의 정보를 이용하는 화면 간 부호화 방식으로 처리된다. 기존에는 보통 참조영상으로 이 전에 재생된 영상의 정보를 이용하였지만, H.264에서는 최대 5개의 참조 영상 중 최적의 것을 참조한다 [1], [28].

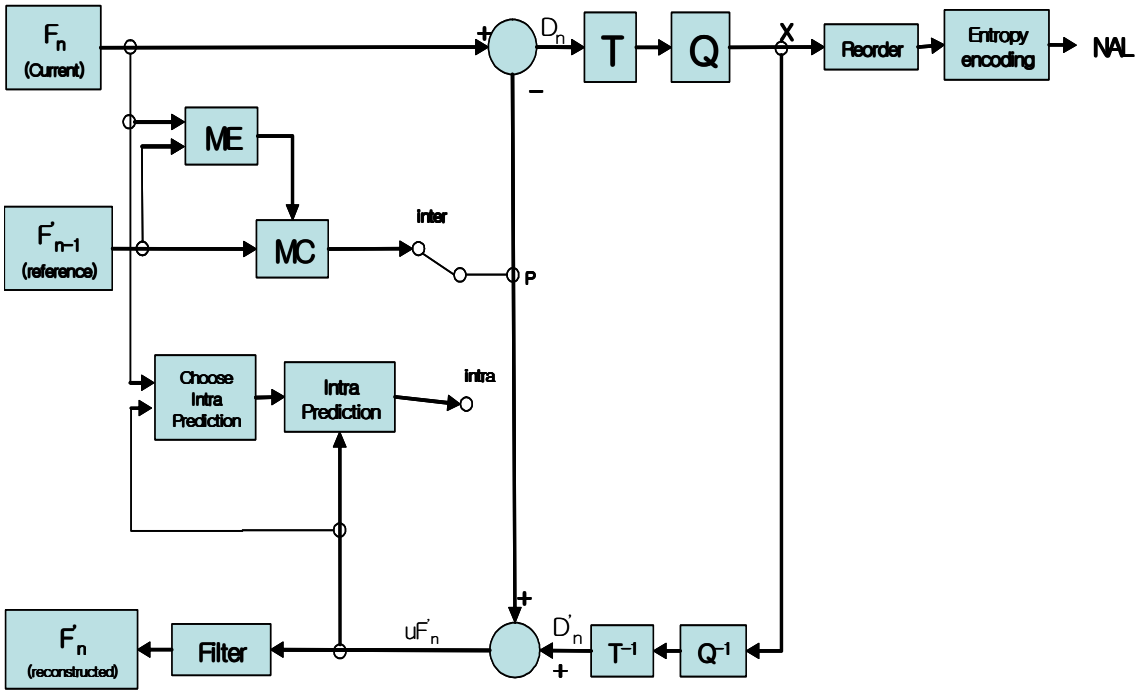


그림 *2-3. H.264* 인코더

1. 블록기반 움직임 추정 및 보상

일반적으로 널리 사용되는 움직임 보상의 방법은 현재 프레임의 사각형 구간 또는 블록의 움직임을 보상하는 것이다. 현재 프레임에서 M×N 샘플을 갖는 블록 각각에 대하여 다음의 과정이 수행된다.

1. 현재 프레임의 M×N 샘플 블록과 일치하는 M×N 샘플 영역을 찾기 위해 참조 프레임의 영역을 탐색한다. 현재의 M×N 블록에서 후보영역을 뺌으로써 구해지는 오차 에너지가 최소가 되는 후보 영역을 가장 일치하는 영역으로 선택하는 방법이 널리 사용된다. 가장 일치하는 영역을 찾아내는 이러한 과정을 움직임 추정이라고 한다.

2. 선택된 후보 영역은 현재의 M×N 블록을 위한 예측블록이 되고, 현재 블록에서 예측 블록을 뺌으로써 M×N 오차블록이 만들어지는데, 이러한 과정을 움직임 보상이라고 한다.

3. 오차 블록은 인코딩되어 전송되며, 현재 블록의 위치와 후보 영역의 위치 사이의 차이 값인 움직임 벡터 또한 전송된다.

H.264의 이전 표준안들과 중요한 차이점은 16 × 16 부터 4 × 4까지의 다양한 블록 사이즈를 지원하고, 1/4 화소 휘도의 세밀한 서브 샘플의 해상도를 지원하는 것이다 [27]. 기존의 표준에서는 16×16 또는 8×8 단위로 움직임 추정이 이루어졌지만, H.264에서는 매크로 블록의 움직임 추정을 총 7개의 모드로 정의하고, 각 모드에 따른 가변 블록 단위의 움직임 추정이 이루어진다. 다시 말해서 움직임 추정 및 보상이 규칙적이지 않고, 다양한 블록 크기로 이루어진다. 움직임 추정 및 보상의 정밀도에 있어서는 과거 1/2 화소 단위까지 움직임 보상이 실시되었지만, H.264에서는 1/4 화소 단위까지 움직임 보상이 실시된다 [1], [12].

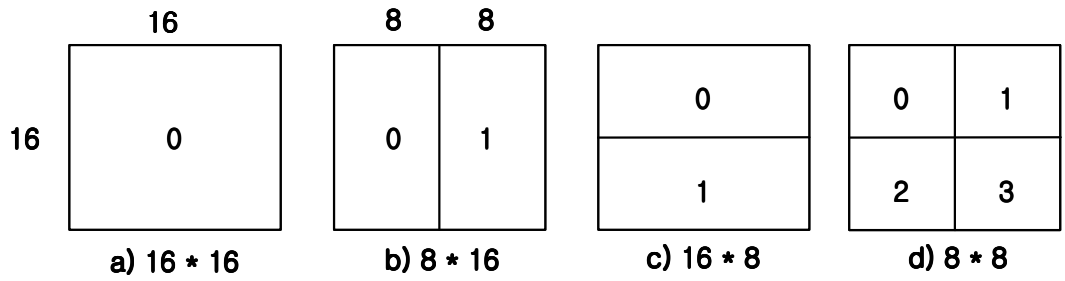


그림 *2-4.* 매크로블록 파티션

*a) 16 * 16, b) 8 * 16, c) 16 * 8, d) 8 * 8*

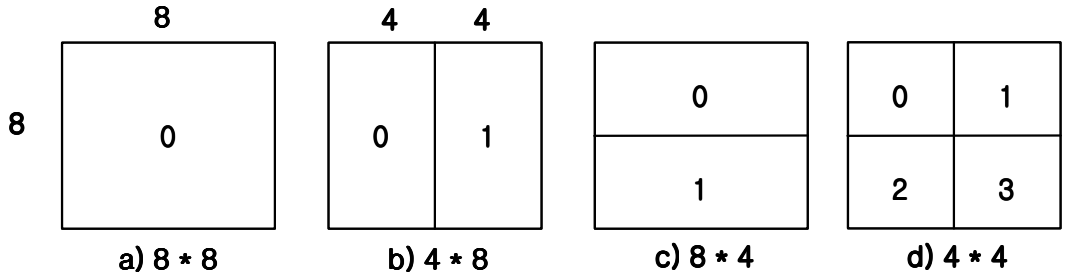


그림 *2-5.* 서브 매크로블록 파티션

*a) 8 * 8, b) 4 * 8, c) 8 * 4, d) 4 * 4*

2. 변환 및 양자화

H.264는 코딩될 오차 데이터의 종류에 따라 3가지 변환을 사용하는데, 16×16 모드로 예측된 인트라 매크로블록내의 휘도 DC 계수들의 4×4 배열을 위한 Hadamard 변환, 모든 매크로블록 내의 색차 DCT 계수들의 2×2 배열을 위한 Hadamard 변환 그리고 다른 모든 4×4 블록의 오차 데이터를 위한 DCT 기반 변환이 사용된다.

양자화란 DCT 등의 변환과정을 수행한 이후에 0에 근접한 크지 않은 DCT 계수를 제거함으로써 이미지 데이터의 정확도를 감소시키는 데 사용된다. H.264의 양자화는 스칼라 양자화를 사용한다. 스칼라 양자화란 하나의 입력 샘플을 하나의 양자화된 출력값으로 대체하는 방식이다 [1].

3. 스캔

인코더에서 각각의 양자화된 변환 계수들의 4×4 블록은 16개의 요소를 갖는 배열에 지그재그 순서로 배치된다.

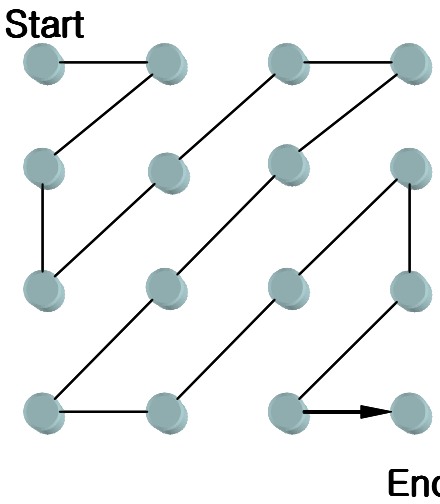


그림 2-6. 4×4 휘도 블록을 위한 지그재그 스캔 (프레임모드)

4. 엔트로피 코딩

엔트로피 인코더는 비디오 영상의 요소를 나타내는 연속적인 부호를 전송하거나 저장하기 용이한 압축된 비트스트림으로 변환한다. 입력부호는 양자화된 변환계수, 움직임벡터, 영상의 동기를 맞추는 지점을 나타내는 코드인 마커, 헤더 그리고 부가적인 정보를 포함 할 수 있다. 가장 널리 사용되는 두 가지 엔트로피 코딩이법에는 '변형 허프만 가변길이 코드'와 '산술코딩' 이 있다.

5. 디블록킹 필터

블록 왜곡현상을 감소시키기 위해 각각의 디코딩된 매크로블록에 필터가 적용된다. 디블록킹 필터는 인코더와 디코더에서 역변환 후에 적용된다. 필터는 블록의 가장자리를 부드럽게 하여 디코딩된 프레임의 화질을 향상시킨다. 필터링된 이미지는 블록 현상이 있는 필터링되지 않은 이미지보다 원본 프레임에 더 충실하게 복원된 것이기 때문에 압축 성능을 향상시킬 수 있다. 필터링 과정의 선택은 경계세기(boundary strength)와 경계 주위의 이미지 샘플의 변화에 의해 좌우된다.

H.264의 인코딩과정에서의 출력은 VCL(video coding layer) 이고, 전송하거나 저장하기 전에 NAL(network abstraction layer) 단위로 맵핑된다. 각 NAL 단위는 압축된 비디오 데이터 또는 헤더 정보에 해당하는 데이터인 RBSP(raw byte sequence payload)를 포함한다. [1]

그림 2-7. NAL 단위의 순서

제 3 장 신호 변환방식

이미지 또는 비디오 코덱에서 변환 과정의 목적은 이미지 또는 움직임 보상된 오차 데이터를 다른 영역 즉, 변환영역으로 변환하기 위한 것이다.

변환 알고리즘은 다음의 기준에 의해 선택된다.

1. 변환 영역의 데이터는 최소의 상호 의존성을 갖는 요소들로 분리되어야 하고 데이터가 밀집되어 있어야 한다. 변환된 데이터의 에너지 대부분은 적은 개수의 값들에 집중되어야 한다.

2. 역변환이 가능해야 한다.

3. 적은 메모리 용량을 사용하고, 적은 횟수의 연산으로 계산이 용이해야 한다.

비디오 압축에 사용되는 변환 알고리즘은 블록 기반 변환과 이미지 기반 변환의 두 가지 종류로 나누어진다. 블록 기반 변환에는 K-L변환(karhunen-loeve transform) [17], [18], SVD(singular value decomposition) 그리고 DCT 등이 있다. 블록 기반 변환은 N×N 이미지 또는 오차 샘플 블록에 대해 수행되므로 이미지는 블록 단위로 처리된다. 반면, 이미지 기반 변환은 이미지 또는 프레임 전체에 대해 수행된다. 가장 널리 알려진 이미지 기반 변환으로는 DWT(discrete wavelet transform 또는 '웨이블릿')이 있다. DWT와 같은 이미지 변환은 정지 이미지 압축에 있어서 블록 기반의 변환보다 성능이 뛰어나다는 것이 입증되었으나 필요로 하는 메모리가 크고 블록 기반의 움직임 보상과 잘 조화되지 않는다 [14].

블록 기반 변환 방법 중 K-L 변환은 상수계수가 높은 영상 신호에 대하여 데이터간의 상관관계를 줄이는 데 있어 이론적으로는 최적의 변환이다. 그러나 K-L 변환은 커널이 영상에 따라 바뀌므로 커널 산출을 위하여 많은 계산이 필요하며, 전송 시 각 커널을 부가정보로 보내주어야 하므로 실제적인 적용이 어

려운 단점이 있으므로 실제 영상 부호화에는 DCT를 사용한다. [13], [23].

이 장에서는 블록 기반 변환 방법 중 DCT 변환에 대하여 설명하고, 이를 더 세분화한 FDCT(fast discrete cosine transform)에 대하여 설명한다.

제 1절. DCT

DCT은 이미지와 비디오 압축 알고리즘에서 양자화 및 압축 이전에 이미지 또는 오차 데이터의 상관관계를 제거하기 위해 사용하는 방식으로서 N×N 영상 신호들을 적은 수의 영상 신호에 정보를 집중시키고 나머지 신호들은 '0' 또는 작은 값을 갖도록 처리한다. 이러한 특성은 통신 선로의 대역폭 감소 문제 및 정보 저장에 필요한 기억용량의 감축 문제를 동시에 해결하기 때문에 디지털 영상 신호처리 사용 시 발생하는 경제성 문제를 해결하는 핵심 기술의 하나로 연구되어지고 있으며, JPEG, MPEG 등에서 국제 표준 규격으로 권고한 압축기법이다 [15].

DCT는 N×N 샘플의 블록 X에 대해서 수행되면 N×N 계수 블록 Y를 생성한다. DCT의 동작은 변환행렬 C로 설명될 수 있다. N×N 샘플 블록의 순방향 DCT(forward DCT)와 역방향 DCT(inverse DCT)는 식 3.1과 같이 정의된다.

$$Y = CXC^T \qquad \text{(식 3.1)}$$
$$X = C^T YC$$

H.264 변환은 DCT와 거의 동일한 압축 성능을 나타내며, 많은 중요한 장점을 가진다. 변환의 핵심 부분인 CXC^T 는 덧셈, 뺄셈 그리고 쉬프트만을 사용하여 정수 연산을 수행할 수 있다. 여기서 X는 샘플의 행렬, Y는 계수의 행렬이고, C는 N×N 변환 행렬이다. C의 요소는 식 3.2와 같다.

$$C_{ij} = C_i \cos\frac{(2j+1)i\pi}{2N} \text{ 여기서 } C_i = \sqrt{\frac{1}{N}}(i=0),\ C_i = \sqrt{\frac{2}{N}}(i>0) \quad \text{(식 3.2)}$$

식 3.1은 식 3.3과 같은 덧셈의 형태로 나타낼 수 있다.

$$Y_{xy} = C_x C_y \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} X_{ij} \cos\frac{(2j+1)y\pi}{2N} \cos\frac{(2i+1)x\pi}{2N} \quad \text{(식 3.3)}$$

$$X_{ij} = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} C_x C_y Y_{xy} \cos\frac{(2j+1)y\pi}{2N} \cos\frac{(2i+1)x\pi}{2N}$$

H.264 변환은 DCT에 기반을 둔 변형된 DCT 기법을 사용하고, 다음과 같은 몇 가지 근본적인 차이점이 있다. 첫째, 모든 연산은 정수 연산을 사용하여 수행되므로 디코딩이 정확도가 손실되지 않는다. 둘째, 정수 연산이므로 인코더와 디코더의 역변환 결과가 정확히 일치하도록 하는 것이 가능하다. 셋째, 변환의 핵심부분을 덧셈과 쉬프트 연산만으로 구현할 수 있다. 곱셈부분이 양자화에 통합되어 전체 곱셈의 횟수를 감소시킨다. 이러한 이점들이 기존의 표준화

보다도 정확한 비디오 압축을 가능하게 한다 [1], [15].

DCT를 구현하는 방법은 FFT처럼 successive decimation 기법을 사용하는 FDCT, FFT를 이용하는 FFT에 기반한 DCT, 그리고 계수별 순환 연산식을 사용하는 순환연산 DCT로 분류할 수 있다 [16], [19], [20], [21].

제 2절. 고속 DCT 알고리즘

많은 고속 DCT 알고리즘[29]들은 계산량을 줄이기 위해 변환 행렬을 변환하여 상당수의 계수를 1또는 0으로 대치시켰다. 따라서 이들 알고리즘들을 적용하는 경우, 곱셈 횟수는 감소하고 코사인 항의 종류도 줄어든다. FDCT는 빠른 연산 구조를 가지지만, 계수의 안정성에 문제가 있어 전처리과정과 후처리과정과 같은 추가적인 연산이 필요하다.

그림 3-1. FDCT 알고리즘 개념도

FDCT 알고리즘들은 대부분 1차원 DCT를 위한 것인데 일반적으로 2차원 신호인 영상신호에 대한 2차원 DCT의 계산에는 N×N 입력신호에 대하여 행 방향으로 1차원 DCT를 수행한 후에 다시 열 방향으로 1차원 DCT를 수행하는 행렬분해기법(Row-Column Method)이 많이 사용된다. 즉, N×N 개의 2차원 DCT변환을 위하여 2N개의 1차원 DCT를 순차적으로 수행하는 것이다 [15], [19], [25].

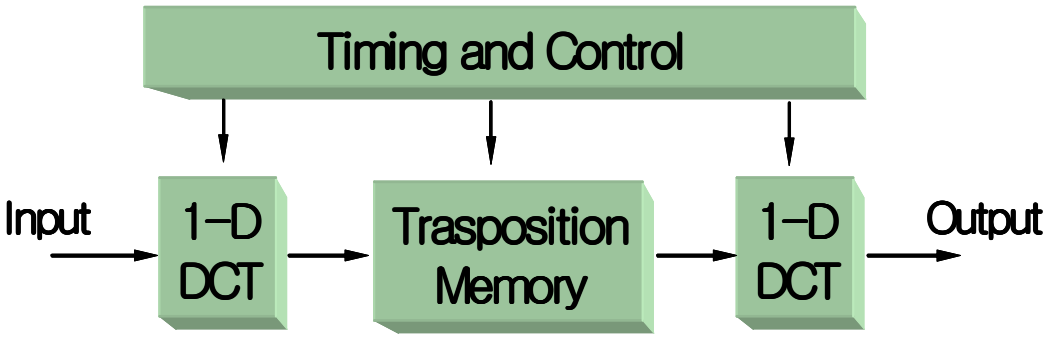


그림 3-2. 2차원 DCT 블럭도

영상에 적용하는 일반적인 2차원 DCT와 IDCT의 정의식은 식 3.3 과 식 3.4 과 같다.

$$F(u,v)=\frac{4}{N^2}C(u)C(v)\sum_{x=0}^{N}\sum_{y=0}^{N}f(x,y)\cos\left[\frac{(2x+1)u\pi}{2N}\right]\cos\left[\frac{(2y+1)v\pi}{2N}\right] \quad (식\ 3.3)$$

$$f(x,y)=\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}C(u)C(v)F(u,v)\cos\left[\frac{(2x+1)u\pi}{2N}\right]\cos\left[\frac{(2y+1)v\pi}{2N}\right] \quad (식\ 3.4)$$

(단, N×N은 블록크기, u,v=0,1,2,3...N-1, C(w)는 w=0인 경우에만 $\frac{1}{\sqrt{2}}$ 이고 나머지는 0이다.)

1974년 DCT가 소개된 이후 초창기에는 입력 시퀀스를 재배열하여 얻는 FFT를 수학적으로 변환하여 FDCT를 구현하였다. 그러나 FFT 알고리즘을 통한 FDCT를 얻는 방법들은 복잡도가 아주 좋은 FFT알고리즘이 필요할 뿐만 아니라 입력 데이터 샘플을 재배열하여 결합하는 과정에서 복잡도가 증가하며 FFT를 통한 간접적인 DCT의 구현이기 때문에 속도 개선의 한계가 있었다.

1. Lee's DCT

1984년 Lee는 수학적으로 inversion과 division을 행하고 블록 크기에 따른 반복성을 이용함으로써 FDCT를 구현하였다. Lee's DCT 알고리즘[16]은 곱셈 연산의 수를 줄이고 단순한 구조를 갖는 고속 알고리즘으로써 FFT(Fast Fourier Transform) 와 유사한 전개방식을 사용했다.

Lee's DCT를 위해서는 먼저 입력을 bit-reverse 순서로 재정렬이 필요하다. 재정렬된 입력은 2-point DCT 형태까지 단계적으로 나뉘어 계산되고, 최종 출

력 또한 재정렬이 필요하다. Lee's DCT 방식은 블록 다이어그램으로 표현하면 그림 3-3과 같다.

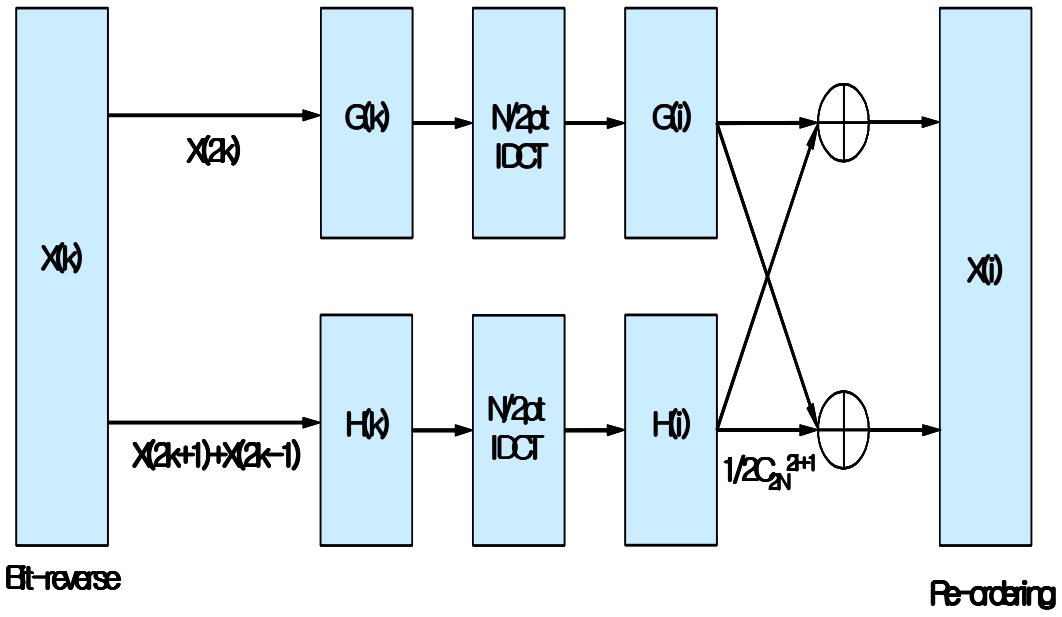


그림 *3-3. Lee's Fast DCT* 블록 다이어그램

Lee's DCT 정수연산을 수행하는 구조로 구현 시 최종단의 변환 계수를 구하기 위해 세단계의 곱셈을 수행하므로 곱셈의 단수가 증가하여 연산의 오차가 누적된다. 즉, 곱셈 연산의 단수를 줄이는 것이 정수연산에서 오차문제를 해결하는 요소가 된다 [20], [23], [24].

2. Chen-DCT

Chen-DCT 알고리즘은 1977년 W. H Chen, C. Harrison Smith, S. C. Fralick에 의해 제안된 고속 DCT 알고리즘으로써 행렬로 표현된 DCT kenel을 0과 1인 성분을 많이 가지고 있는 행렬의 곱으로 인수분해 함으로서 곱셈수를 줄이는 방식이다. 즉, 코사인의 주기적 특성을 이용하여 불필요한 곱셈수를 줄임으로써 계산의 속도를 향상시키는 방식이다.

Chen-DCT 알고리즘의 DCT, IDCT에 대한 수식은 식 3.5와 같이 정의된다 [21], [22].

$$X(u,v)=\frac{2}{N}C(u)C(v)\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}x(i,j)\cos\frac{(2i+1)u\pi}{2N}\cos\frac{(2j+1)v\pi}{2N} \quad \text{(식 3.5)}$$

$$x(i,j)=\frac{2}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}C(u)C(v)X(u,v)\cos\frac{(2i+1)u\pi}{2N}\cos\frac{(2j+1)v\pi}{2N}$$

(단, u,v는 변환영역의 좌표이며, x,y는 화소 영역이 공간 좌표)

Chen-DCT방식은 기존의 고속 DCT 알고리즘과는 달리 DCT 및 IDCT의 연산 과정에서 처리 속도에 직접적으로 영향을 미치는 곱셈 연산을 쉬프트 및 덧셈 연산으로 대체함으로써 곱셈 연산으로 인한 문제를 근본적으로 해결하였다. 그 방법은 입력된 영상 데이터를 하나의 레지스터에 저장하고, 이 데이터를 1비트씩 오른쪽으로 수차례에 걸쳐 쉬프트하여 각각의 레지스터에 저장한다. 이 입력된 영상 데이터와 곱셈 연산을 2진 코드로 표시할 경우 이 코드의 '1'인 비트에 대응되는 레지스터 내에 저장된 값만을 선택하여 더해줌으로써 영상 데이터와 코사인 계수와의 곱셈 연산이 쉬프트와 덧셈 연산으로 대치되는 것이다. Chen-DCT 알고리즘의 구조는 Row-Column 접근 방식으로 구현하며 Chen-DCT의 데이터 흐름도는 그림 3-4와 같다 [15], [25].

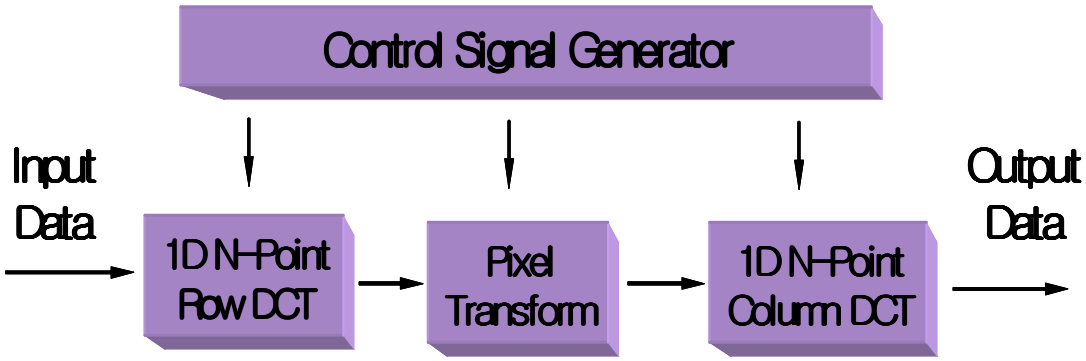


그림 *3-4. Chen-DCT* 알고리즘의 데이터 흐름도

제 4 장 실험 결과

본 논문에서는 H.264 동영상 압축 과정 중 부호화 과정에서 DCT 방식을 고속 DCT 방식 중 제안하는 알고리즘인 Chen-DCT 알고리즘을 사용하여 부호화 과정의 속도 변화 및 화질 변화를 측정하였다.

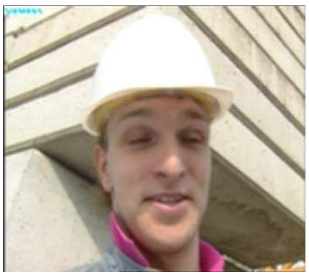
제 1절. 실험 환경 및 데이터

이 실험은 Intel Pentium4 2GHz Processor, 512MB RAM 환경에서 진행되었으며, 컴파일을 위하여 Microsoft Visual C++ 6.0 SP5를 사용하였다. JM10.1[26] 코덱의 기본 결과 값을 기준으로 하였고, 8×8 기본 DCT 코드를 Chen-DCT로 수정한 후 그 결과 값과 비교하였다. 실험에 사용한 양자화계수 값은 28이며, 프레임 비율은 30Hz로 설정하였다. 실험에 사용된 비디오 시퀀스는 표 4-1과 같고, 각 시퀀스는 그림 4-1과 같다.

표 4-1. 실험에 사용한 QCIF 형식의 Sequence

Sequence	Total Frame	Frame Format
Foreman	300 frame	QCIF
Akiyo	300 frame	QCIF
Salesman	449 frame	QCIF
Claire	494 frame	QCIF
Container	300 frame	QCIF
News	449 frame	QCIF

표 4-1에 정리되어 있는 영상 중에서 Container와 Foreman는 배경이나 움직임의 변화가 많은 영상이고, Akiyo, Claire, News, Salesman 는 비교적 배경의 변화와 움직임이 적은 영상이다.



Foreman

Akiyo

Salesman

Claire

Container

News

그림 4-1. 실험에 사용된 비디오 시퀀스

제 2절. 실험 결과

표 4-1과 같이 5개의 qcif 형식의 영상을 먼저 JM10.1에서 실험하였고, 같은 영상으로 본 논문에서 제안하는 Chen-DCT 알고리즘으로 수정하여 실험하였다.

PSNR(Peak Signal to Noise Raio)은 쉽고 빠르게 계산할 수 있는 매우 대중적인 화질 측정방법이다. PSNR의 단위는 dB(데시벨)이다. 영상의 PSNR(dB)이 약 1.0dB정도 변화하면 사람이 그 영상의 화질변화를 판별할 수 있다. PSNR은 로그(log)단위로 측정되며, 이미지에 존재할 수 있는 최대 샘플 개수의 제곱과 원본 이미지와 손상된 이미지 또는 비디오 프레임 사이의 평균제곱오차(MSE:Mean Squared Error)의 비율 에 의해 결정된다. PSNR은 식 4.1과 식 4.2 와 같이 정의한다.

$$MSE = \sum_{m=1}^{M} \sum_{N=1}^{N} (x(m,n) - X^{R}(m,n))^{2} \qquad \text{(식 4.1)}$$

$$PSNR_{dB} = 10 \log_{10} \frac{(2^{n}-1)^{2}}{MSE} \qquad \text{(식 4.2)}$$

식 4.1 에서 M,N은 영상의 가로와 세로의 크기이고, $x(m,n)$ 은 원 영상을 나타내고, $x^{R}(m,n)$ 은 움직임 추정 및 보상된 영상을 나타낸다.

표 4-3과 표4-4는 2D-DCT 형식인 Chen-DCT 알고리즘을 1D N-Point RowDCT와 1D N-Point ClumnDCT 형식으로 작성한 코드이다.

표 *4-2. Chen-DCT*의 *1D Row DCT* 함수

```
for( i=0; i<8; i++)
{
        a[0] = LS((img->m7[i][0] + img->m7[i][7]), 2);
        a[1] = LS((img->m7[i][1] + img->m7[i][6]), 2);
        a[2] = LS((img->m7[i][2] + img->m7[i][5]), 2);
        a[3] = LS((img->m7[i][3] + img->m7[i][4]), 2);

        b[0] = a[0] + a[3];
        b[1] = a[1] + a[2];
        b[2] = a[1] - a[2];
        b[3] = a[0] - a[3];

       a[4] = LS((img->m7[i][3] - img->m7[i][4]), 2); //c0
       a[5] = LS((img->m7[i][2] - img->m7[i][5]), 2); //c1
       a[6] = LS((img->m7[i][1] - img->m7[i][6]), 2); //c2
       a[7] = LS((img->m7[i][0] - img->m7[i][7]), 2); //c3

        m6[i][0] = RS((k0 * ( b[0] + b[1])), 9);
        m6[i][4] = RS((k0 * ( b[0] - b[1])), 9);
        m6[i][2] = RS((k2 * b[2] + k1 * b[3]), 9);
        m6[i][6] = RS((k2 * b[3] - k1 * b[2]), 9);

        b[0]= RS((k0 * (a[6] - a[5])), 9);
        b[1]= RS((k0 * (a[6] + a[5])), 9);
        a[0]= a[4] + b[0];
        a[1]= a[4] - b[0];
        a[2]= a[7] - b[1];
        a[3]= a[7] + b[1];

        m6[i][1] = RS((k6 * a[0] + k3 * a[3]), 9);
        m6[i][3] = RS((k4 * a[2] - k5 * a[1]), 9);
        m6[i][5] = RS((k4 * a[1] + k5 * a[2]), 9);
        m6[i][7] = RS((k6 * a[3] - k3 * a[0]), 9);

}
```

표 *4-3. Chen-DCT*의 *1D ColumnDCT* 함수

```
for( i=0; i<8; i++)
        {
                a[0] = RS((m6[0][i] + m6[7][i]), 1);
                a[1] = RS((m6[1][i] + m6[6][i]), 1);
                a[2] = RS((m6[2][i] + m6[5][i]), 1);
                a[3] = RS((m6[3][i] + m6[4][i]), 1);

                b[0] = a[0] + a[3];
                b[1] = a[1] + a[2];
                b[2] = a[1] - a[2];
                b[3] = a[0] - a[3];

                a[4] = RS((m6[3][i] - m6[4][i]), 1);     //c0
                a[5] = RS((m6[2][i] - m6[5][i]), 1);     //c1
                a[6] = RS((m6[1][i] - m6[6][i]), 1);     //c2
                a[7] = RS((m6[0][i] - m6[7][i]), 1);     //c3

                img->m7[0][i] = RS((k0 * (b[0] + b[1])), 9);
                img->m7[4][i] = RS((k0 * (b[0] - b[1])), 9);
                img->m7[2][i] = RS((k2 * b[2] + k1 * b[3]), 9);
                img->m7[6][i] = RS((k2 * b[2] - k1 * b[3]), 9);

                b[0]= RS((k0 * (a[6] - a[5])), 9);
                b[1]= RS((k0 * (a[6] + a[5])), 9);
                a[0] = a[4] + b[0];
                a[1] = a[4] - b[0];
                a[2] = a[7] - b[1];
                a[3] = a[7] + b[1];

                img->m7[1][i] = RS((k6 * a[0] + k3 * a[3]), 9);
                img->m7[3][i] = RS((k4 * a[2] - k5 * a[1]), 9);
                img->m7[5][i] = RS((k4 * a[1] + k5 * a[2]), 9);
                img->m7[7][i] = RS((k6 * a[3] - k3 * a[0]), 9);
        }
```

5개의 영상에 대한 실험 결과는 표 4-4, 4-5과 같고, JM10.1 Encoder로 실험한 결과와 Chen-DCT를 사용한 Encoder의 결과는 그림 4-2, 그림 4-3, 그림 4-4, 그림 4-5, 그림 4-6, 그림 4-7에서 비교하였다.

그림 4-7에서는 각 영상에 대한 JM10.1 Encoder로 실험한 FPS(Frame per Sec.)와 Chen-DCT를 사용한 Encoder의 FPS를 비교하였고, 그림 4-8에서는 JM10.1 Encoder로 실험한 PSNR와 Chen-DCT를 사용한 Encoder의 PSNR을 비교하였다.

실험 결과를 비교하기 위하여 JM10.1 Encoder의 결과는 표 4-4와 같다.

표 *4-4. JM10.1 Encoder*의 실험 결과

Sequence	*Encoding FPS*	*PSNR(Y)*	*PSNR(U)*	*PSNR(V)*
Foreman	0.62 fps.	36.96	41.05	43.02
Akiyo	0.66 fps.	39.42	41.12	42.02
Salesman	0.61 fps.	36.42	39.83	40.52
Claire	0.68 fps.	40.70	40.56	42.56
Container	0.61 fps.	37.30	41.38	40.87
News	0.60 fps.	37.45	40.20	40.61

본 논문에서 제안하는 DCT 방식을 Chen-DCT 알고리즘을 이용하여 JM10.1 코드를 수정하여 그 결과값은 표 4-5과 같다.

표 *4-5. Chen-DCT* 알고리즘을 이용한 *Encoder*의 실험 결과

Sequence	*Encoding FPS*	*PSNR(Y)*	*PSNR(U)*	*PSNR(V)*
Foreman	1.24 fps.	36.87	41.13	42.87
Akiyo	1.32 fps.	39.35	41.14	42.27
Salesman	1.22 fps.	36.34	39.79	40.59
Claire	1.36 fps.	40.75	40.47	42.46
Container	1.18 fps.	37.31	41.34	40.89
News	1.20 fps.	37.45	40.03	40.63

실험 결과, FPS(Frame Per Sec)는 5개의 영상에서 모두 약 2배정도 향상되었음을 확인할 수 있다. 즉, Chen-DCT알고리즘을 이용함으로써 부호화 시간이 약 2배정도 빨라졌다는 것을 알 수 있다. PSNR은 영상에 따라서 감소한 것도 있지만 약 0.15dB 정도 향상되었음을 확인할 수 있다.

각 영상에 대하여 JM10.1 Encoder의 PSNR과 Chen-DCT를 사용한 Encoder의 PSNR을 비교하였다.

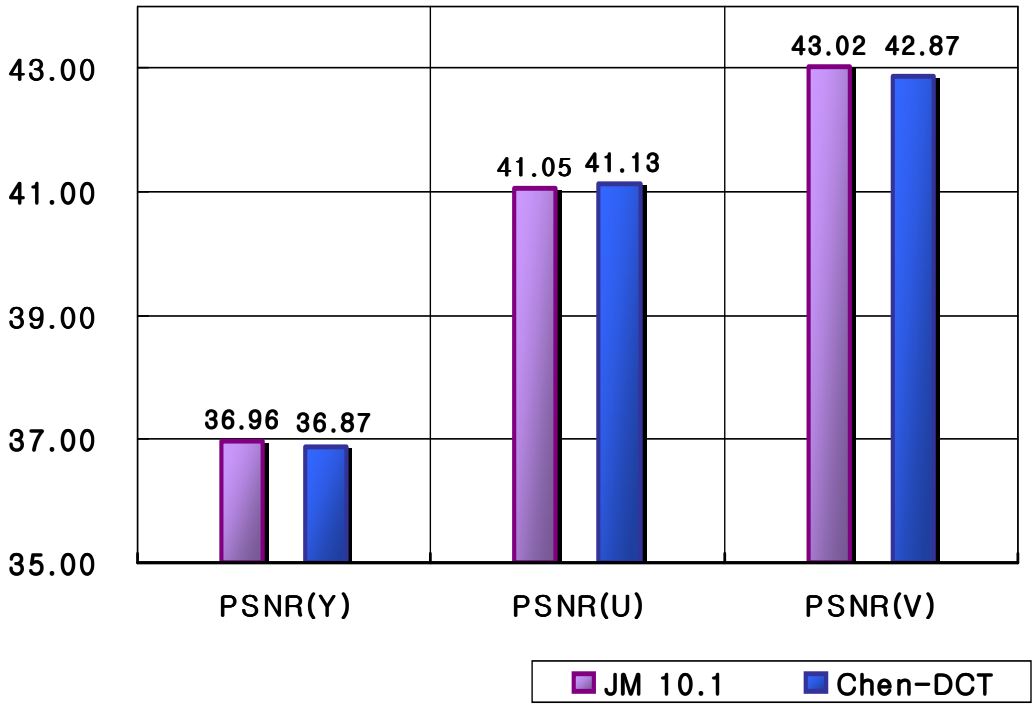


그림 *4-2. Foreman* 영상의 *JM10.1 Encoder*와 *Chen-DCT*를 이용한 *Encoder*의 *PSNR*비교

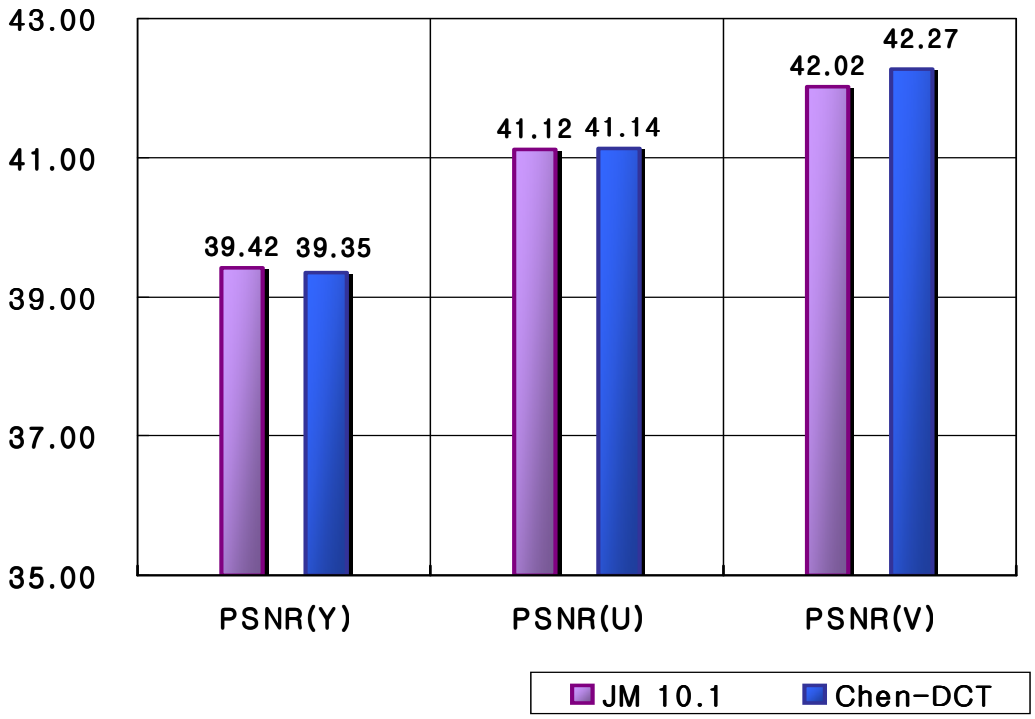


그림 *4-3. Akiyo* 영상의 *JM10.1 Encoder*와
*Chen-DCT*를 이용한 *Encoder*의 *PSNR* 비교

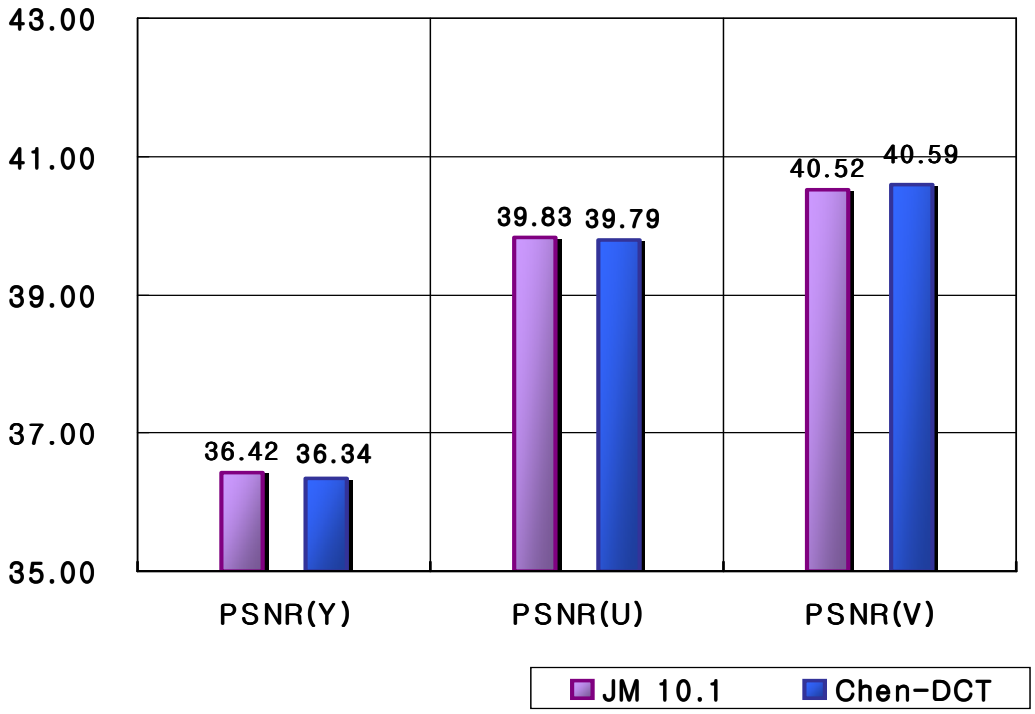


그림 *4-4. Salesman* 영상의 *JM10.1 Encoder*와
*Chen-DCT*를 이용한 *Encoder*의 *PSNR* 비교

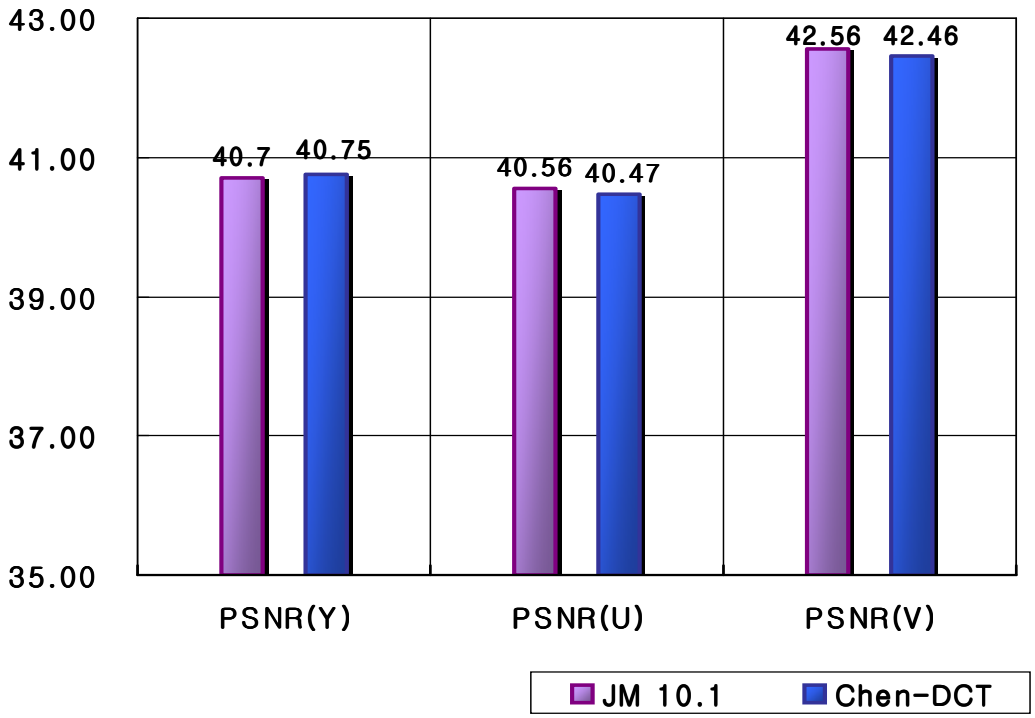


그림 *4-5. Claire* 영상의 *JM10.1 Encoder*와
*Chen-DCT*를 이용한 *Encoder*의 *PSNR* 비교

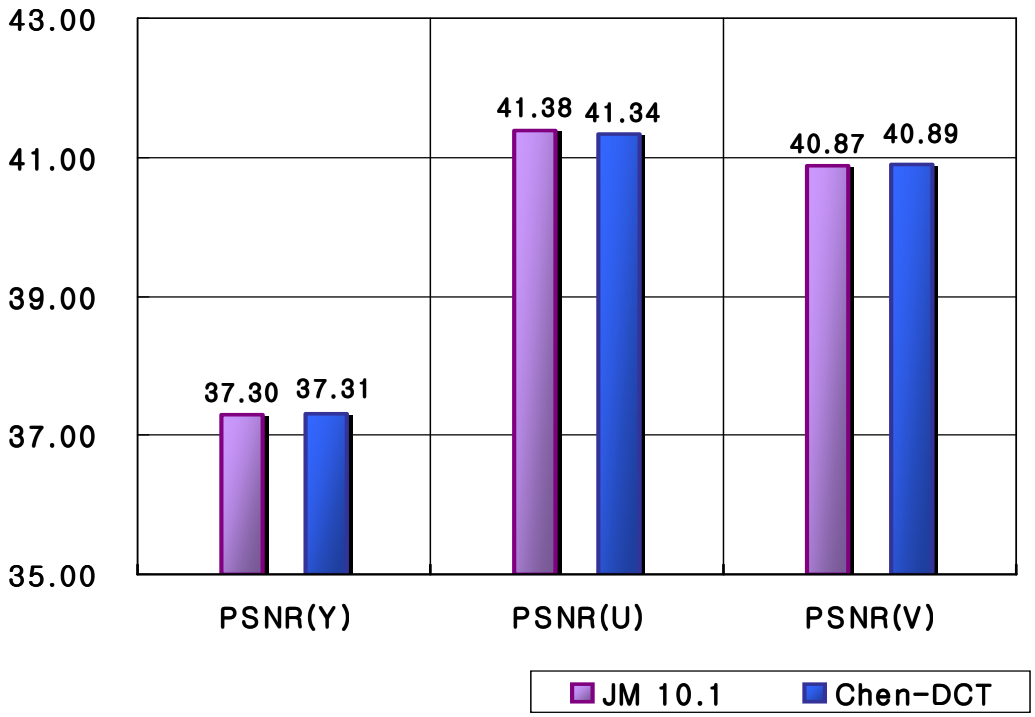


그림 *4-6. Container* 영상의 *JM10.1 Encoder*와
*Chen-DCT*를 이용한 *Encoder*의 *PSNR* 비교

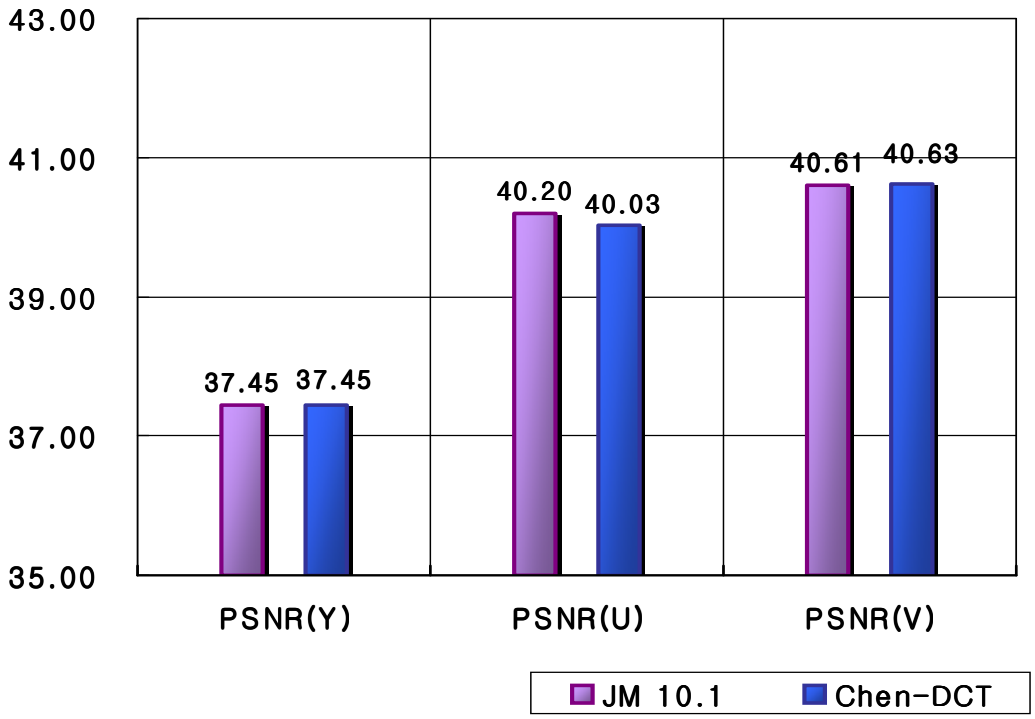


그림 *4-7. News* 영상의 *JM10.1 Encoder*와
*Chen-DCT*를 이용한 *Encoder*의 *PSNR* 비교

각 영상에 대하여 JM10.1 Encoder에서 실험한 FPS, 즉 초당 압축하는 프레임 수와 Chen-DCT 알고리즘을 사용한 Encoder에서 실험한 FPS의 결과를 비교하면 그림 4-8과 같다.

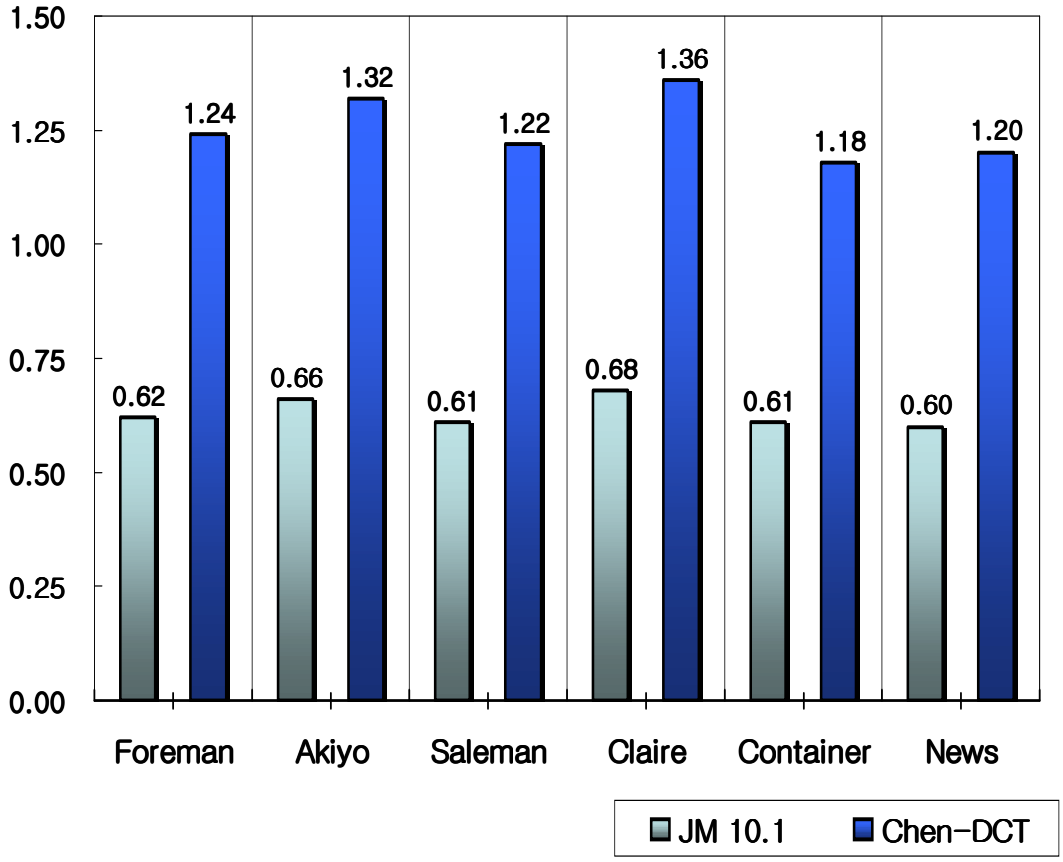


그림 *4-8. JM10.1 Encoder*의 *FPS*와
*Chen-DCT*를 사용한 *Encoder FPS* 비교

각 영상에 대하여 JM10.1 Encoder에서 실험한 PSNR과 Chen-DCT 알고리즘을 사용한 Encoder에서 실험한 PSNR의 결과를 비교하면 그림 4-9 같다.

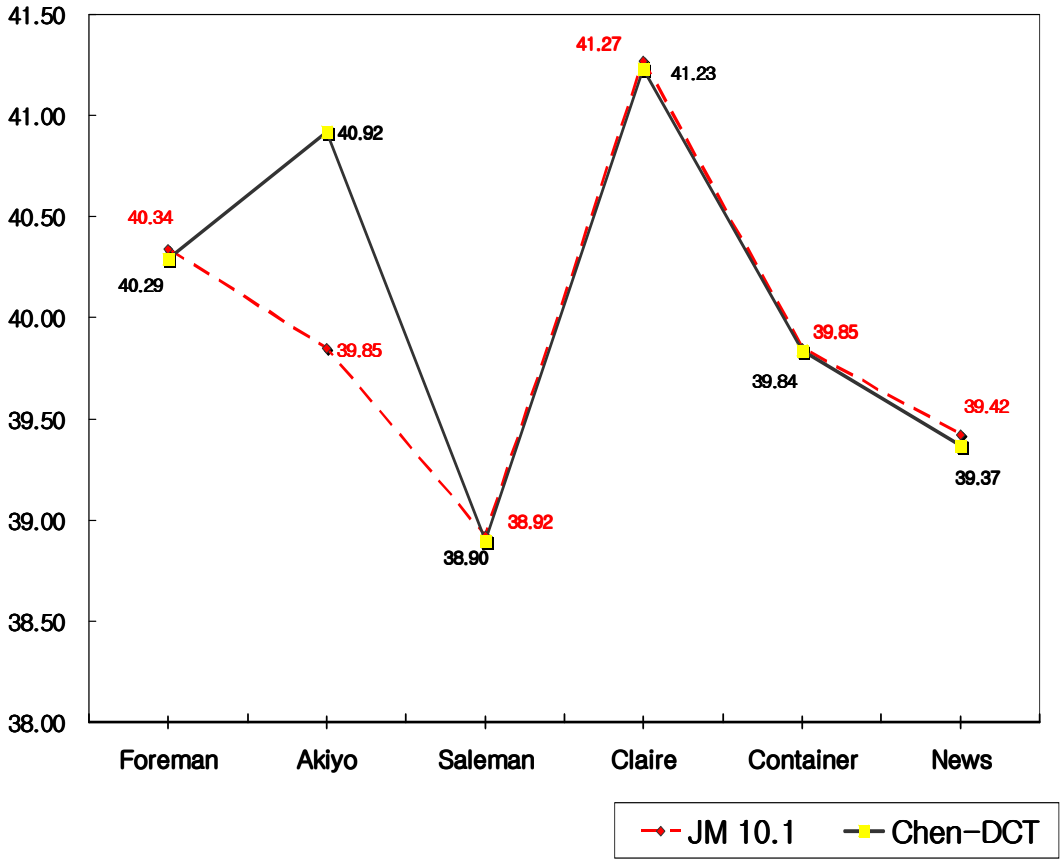


그림 *4-9. JM10.1 Encoder* 와 *Chen-DCT*를 이용한 *Encoder*의 각 영상의 *PSNR* 비교

제 5 장 결 론

통신 및 네트워크 기술의 발전으로 모바일 환경이나 무선 통신 분야에서 멀티미디어 TV, 주문형 비디오(VOD), HDTV, 디지털 방송 DVD 등 과 같은 대용량 디지털 멀티미디어를 실시간으로 제공받기 위하여 H.264/AVC와 같은 동영상 압축 방식을 이용하여 데이터의 크기를 줄어야 한다. H.264/AVC 동영상 압축 방식은 기존의 압축 방식과는 달리 동일한 대역폭의 절반 속도에서도 동일한 화질을 제공하고, 네트워크 에러 시 화상 데이터에 손실이 있을 때에도 화상이 완전히 끊어지는 대신 화질의 약화만 가져온다는 장점이 있다.

H.264/AVC 동영상 압축 과정 중 DCT 변환은 DCT에 기반을 두고 있지만 정수 연산을 사용하여 인코더와 디코더의 결과가 정확하게 일치하고, DCT 변환 과정 중 곱셈 부분을 덧셈과 쉬프트 연산만으로 구현하였고, 곱셈 부분은 양자화에 통합되어 전체 곱셈 부분을 감소시켜 기존의 다른 표준들 보다 정확한 비디오 압축이 가능하다. 이러한 DCT의 특성은 통신 선로의 대역폭 감소 문제 및 정보 저장에 필요한 기억용량의 감축 문제를 해결하는 핵심기술의 하나로 연구되어지고 있고, JPEG, MPEG 등에서 국제 표준 규격으로 권고한 압축기법이다.

본 논문에서는 DCT 과정에서의 속도 개선 및 압축 효율을 높이기 위하여 Chen-DCT 알고리즘을 사용하여 H.264 동영상 압축 방식을 제안한다. 고속 DCT 알고리즘 중에 한 방식으로 제안된 Chen-DCT 알고리즘은 행렬로 표현된 DCT kenel을 0과 1인 성분을 많이 가지고 있는 행렬의 곱으로 인수분해 함으로서 곱셈수를 줄이는 방식이다.

2차원 DCT인 Chen-DCT의 구현은 1차원 DCT를 행 방향으로 수행한 후에 다시 열 방향으로 1차원 DCT를 수행하는 행렬 분해 기법을 사용하였다.

JM10.1 Encoder와 제안하는 Chen-DCT를 이용한 Encoder와의 부호화 속도와 압축 효율을 실험 결과를 살펴보면 JM10.1 Encoder와 비교하여 PSNR의 값은 각 영상마다 조금씩의 열화는 보였으나, 부호화하는 속도는 2배 정도의 증가했음을 알 수 있다..

향후 연구 과제에서는 대용량 멀티미디어 데이터의 압축만을 전담하는 하드웨어와 소프트웨어에 대한 연구가 필요하다. 본 논문의 하드웨어 실험환경은 대용량 멀티미디어 정보 처리를 전담하지 않는 일반 PC 이므로 H.264/AVC 동영상 압축의 큰 장점인 대용량의 멀티미디어 정보를 다른 사용자에게 실시간으로 제공하기는 어려운 문제점을 가지고 있다. 그러므로 대용량의 멀티미디어 데이터를 전담 할 수 있는 하드웨어나 소프트웨어의 발전은 사용자들에게 VOD나 MOD와 같은 멀티미디어 데이터를 다운로드하지 않고, TV 방송처럼 실시간으로 이용할 수 있게 될 것이다.

[참 고 문 헌]

[1] Iain E. G. Richardson, "*H.264 and Mpeg-4 Video Compression : Video Coding for Next-generation multimedia*", John WILEY & Sons Ltd., 2003.

[2] A. Vetro, C. Christopoulos, and Huifang Sun, "Video Transcoding architecture and techniques: an overview", *IEEE Signal Processing Magazine*, Vol. 20, issue. 2, pp. 18-29, March. 2003.

[3] T. Shanableh and M. Ghanbari, "Heterogeous Video Transcoding to Lower Spatio-Temporal Resolutions and Different Encoding Format", *IEEE Trans. Multimedia*, Vol. 2, pp. 101-110, Jun. 2000.

[4] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding", *IEEE Journal on Selected Articles in Communications*, Vol. 29, pp. 1799-1808, 1981.

[5] M. Bierling, "Displacement estimaiton by hierarchical block mathching", *SPIE*, Vol. 1001, pp. 942-951, 1998.

[6] W. Li and E. Salari, "Successive elimination algorithm for motion estimation", *IEEE Trans. Image Processing*, Vol. 4, No. 1, pp. 105-107, 1995.

[7] H. S. Wang and R. M. Mersereau, "Fast algorithm for the estimation of motion vectors", *IEEE Trans. Image Processing*, Vol. 8, No. 3, pp. 435-438, 1999.

[8] Thomas Wiegand, Gary Sullivan, Ajay Luthra, "*Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification(ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC)*", Geneva, May. 2003.

[9] A. Hallapuro, M. Karczewicz and H. Malvar, "*Low Complexity Transform and Quantization - Part I :Basic Implementation*", JVT document JVT-B038, Geneva, February. 2002.

[10] "*차세대 동화상 부호화 방식 MPEG-4 AVC/H.264의 기술동향*", 국제테크노정보연구소.

[11] 진군선, "*개선된 MRME 알고리즘을 이용한 H.264 움직임 추정기 설계 및 FPGA 검증*", 제주대학교 대학원, 2004.

[12] 이규호, "*병합 방법을 이용한 H.264 가변 블록 움직임 추정*", 전남대학교 대학원, 2004.

[13] 안광태, 강경인, 박경배, 김정일, 이광배, 김현욱, 박석천, "*DCT 영역 분류에 의한 고속 알고리즘*", *대한전자공학회, 추계종합학술대회,* 제 18권, 제2호, pp. 856-859, 1995.

[14] 박균재, 이기현, "고속 DCT 구현을 위한 컴퓨터 알고리즘 개선에 관한 연구", *한국통신학회 논문지, 영상처리 및 이해에 관한 워크샵,* 제 4권, 1992.

[15] 이승옥, 정화자, 정기현, 김용득, "고속 DCT 알고리듬을 이용한 IDCT 구조", *대한전자공학회*, 하계종합학술대회 논문지, 제 16권, 제 1호, pp. 314-318, 1993.

[16] 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.

[17] N. Ahmed, T. Natarsjan and K. R. Rao, "Discrete Cosine Trasform," *IEEE Tras. on Comput.*, Vol. com-23, No. 1, pp. 90-93, Jan. 1974.

[18] N. Ahmed and K. R. Rao, "*Orthogonal Transform for Digital Signal Processing*", Springer-Verlag, 1975.

[19] 손영기, "*MPEG 오디오 코더를 이용한 DCT 구현 기법 연구*", 중앙대학교 대학원, 2000.

[20] 민경육, “*효율적인 파이프라인구조를 갖는 1/4 SD Digital VCR용 2-4-8 DCT/IDCT processor의 설계에 관한 연구*”, 한양대학교 산업대학원, 1995.

[21] W. H. Chen, C. H. Smith, and S. Fralick, "A Fast Computational Algorithm for the Discrete Cosine Transform", *IEEE Trans. Commun,* Vol. Com-25, pp. 1004-1009, Sept. 1997.

[22] 안덕철, 김희석, “*고속 DCT/IDCT 설계*”, 청주대학교 산업과학연구, Vol. 17, No. 1, pp. 309-313, 1999.

[23] 이재성, “*프로세서 구조에 따른 DCT 알고리듬의 구현 성능 비교*”, 연세대학교 대학원, 2004.

[24] 정연식, 이임건, 윤택현, 송홍엽, 박규태, “정수곱셈을 위한 고속 이산여현변환(DCT) 알고리즘”, *대한전자공학회*, 제 19권, 제 2호, pp. 1625-1628, 1996.

[25] H. S. Lim, D. S. Kim, N. I. Cho, and S. U. Lee, "Systolic Arrays for 2-D DCT and Other Orthogonal Transforms," *KITE*, Vol. 27, No. 7, July. 1990.

[26] http://iphome.hhi.de/suehring/tml/index.htm

[27] Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, "Study of Final Committee Draft of Joint Video Specification(ITU-T REC. H.264 | EC 14496-10 AVC)," *JVT-100.doc, 6th Meeting* : Awaji, Island, JP, 5-13, December. 2002.

[28] 한기훈, “*고속모드 결정을 통한 H.264 부호기 속도향상*,” 세종대학교 대학원, 2004

[29] H. J. Chung, S. J. Kim, G. H. Jung, Y. D. Kim, " A DCT algorithm using shift and addition," *KITE*, 5, 1993