DCT 기반 H.264 transcoder를 위한 half-pixel 보정 기법

권순영*O 임성렬** 정기동*
부산대학교 컴퓨터공학과*, 부산대학교 멀티미디어 협동과정**
{ksy2020*O, kdchung* }@melon.cs.pusan.ac.kr syim**@wsu.ac.kr

Half-Pixel Correction for H.264 Transcoding in the DCT domain

Soonyoung Kwon*O , Sungyeal Im** , Kidong Chung*
Dept. of Computer Engineering, Pusan National University*
Dept. of Multimedia, Pusan National University**

요 약

최신 동영상 압축 표준인 H.264는 압축 효율을 높이기 위해 기존의 표준과는 다른 1/2 화소 생성 방법을 사용한다. 그러므로 기존의 동영상 압축표준으로 압축된 비트열을 DCT 상에서 H.264로 트랜스코딩(transcoding)하기 위해서는 추가적인 보정 작업이 필요하다. 본 논문에서는 MPEG-2로 압축된 비트열을 DCT 상에서 H.264로 트랜스코딩 할 때 두 표준 간 1/2 화소 값의 차이를 보정하는 기법을 제안한다. 제안된 1/2 화소 보정 기법에서는 DCT 상태의 참조 프레임을 이용하여 두 표준 간의 차이 값을 구하여 입력으로 들어온 블록의 값에 더하여 보정한다. 픽셀 기반에서 보정하는 기법과 성능을 비교한 결과 제안하는 기법이 화질 면에서 우수하며 움직임이 빠른 비디오의 경우 계산량이 높아지는 것으로 나타났다.

1. 서 론

최신의 동영상 압축 기법인 H.264/AVC는 ISO와 ITU에서 2003년 표준으로 승인되었다. 이 표준은 MPEG-2 압축 방식의 화질을 유지하면서 압축률을 50%로 낮추기 위해 가변 블록 움직임 보상, 복수 참조 영상, 1/4 화소 움직임 벡터와 같은 다양한 기법을 추가하였다[1]. 우수한 압축 성능으로 H.264는 다양한 사용자 환경에 사용될 것으로 보이며 기존의 압축 표준을 대체할 것으로 예상된다. 따라서 MPEG-2로 압축된 데이터를 H.264로 변환할 가능성이 매우 높으며 효율적인 트랜스코더의 필요성이 대두되고 있다[2].

최근에는 픽셀기반 트랜스코딩 과정에서 발생할 수 있는 DCT와 IDCT 과정의 불일치 문제를 피하고 계산량도 줄일 수 있는 DCT 기반 트랜스코딩 기법 연구가 활발히 진행되고 있다[3]. 하지만 MPEG-2에서 H.264로의 트랜스코딩에 대한 연구에서는 기존 표준들과의 차이 때문에 픽셀 상에서 트랜스코딩을 수행하였다. 특히 H.264는 MPEG-2와는 다른 1/2 화소 연산 식을 사용한다. 본 논문에서는 앞에서 지적한 차이점을 보정하여 DCT상에서 트랜스코딩이 가능하도록 한다. 제안하는 기법은 DCT 상태의 참조 블록을 이용해서 두 표준간의 1/2 화소 차이 값을 구하고 이를 입력 데이터에 더해줌으로써 보정이 이루어진다.

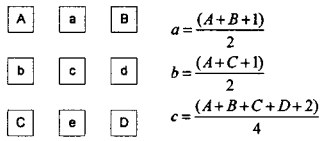
실험에서는 다양한 비디오에 대해서 픽셀 기반 트랜스코더와 화질, 계산량 비교를 수행한다. 실험 결과 픽셀 기반 트랜스코더 보다 더 좋은 화질을 생성하는 것을 확인하였다.

본 논문의 구성은 다음과 같다. 2장에서 기존의 압축 표준과 H.264에서 1/2화소 값을 생성하는 과정을 살펴보고, 제안하는 1/2화소 보정 기법에 대해서 3장에서 알아본다. 제안된 기법의 비교 분석을 4장에서 기술하고, 마지막으로 5장에서 결론을 맺는다.

2. 동영상 압축 표준에서의 1/2 화소

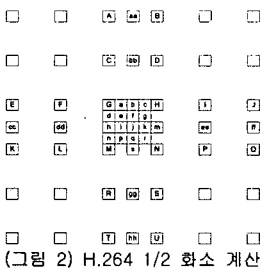
1/2 화소의 경우 MPEG-2, H.263, MPEG-4에서는 생성식이 동일한 반면 H.264는 다른 생성 식을 사용한다. 그러므로 MPEG-2에서 H.264로의 트랜스코딩을 할 때, 움직임 벡터를 그대로 사용하려면 1/2 화소 보정을 수행해야 한다.

MPEG-2, H.263, MPEG-4 등의 표준에서는 비교적 간단하게 이웃하는 2개 또는 4개 화소의 평균값을 1/2 화소로 정한다[4]. (그림 1)에서 MPEG-2 방식의 1/2 화소 계산의 예를 보여주고 있다.



(그림 1) MPEG-2의 1/2 화소 계산

H.264에서는 기존의 표준과는 달리 주위 6개의 화소들을 이용하여 1/2 화소를 계산한다[1][5]. (그림 2)는 H.264에서 1/2 화소의 예를 보여주고 있다.



(그림 2) H.264 1/2 화소 계산

수평 1/2 화소인 b와 수직 1/2 화소 h를 구하기 위해서 아래와 같이 6-taps 필터를 적용한다.

$$b_1 = (E - 5F + 20G + 20H - 5I + J) \quad (2)$$

$$h_1 = (A - 5C + 20G + 20M - 5R + T) \quad (3)$$

계산된 값을 0-255 사이의 값이 되도록 아래의 수식을 이용하여 조정한다.

$$b=(b_1+16)\gg 5 \quad (4)$$

$$h=(h_1+16)\gg 5 \quad (5)$$

수직수평 1/2화소인 j의 경우는 아래와 같은 6-taps 필터를 거치게 되며 cc,dd,h_1,m_1,ee,ff들은 b_1과 비슷한 방법으로 얻어진다.

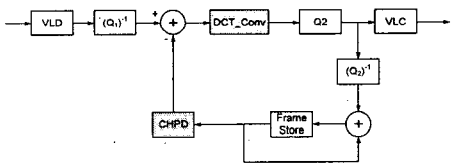
$$j_1=cc-5dd+20h_1+20m_1-5ee+ff \quad (6)$$

$$j=(j_1+512)\gg 10 \quad (7)$$

이와 같은 두 표준간의 1/2화소 계산 방식의 차이 때문에 트랜스코더에서의 보정이 필요하게 된다.

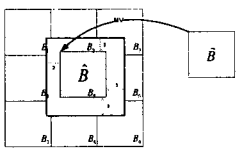
3. DCT 상에서의 1/2 화소 보정기법

(그림 3)는 기존 표준에서 H.264로 변환하기 위한 본 논문에서 제안하는 DCT 기반 트랜스코더 구조를 블록 다이어그램으로 나타낸 것이다. 그림에서 Q_1은 MPEG-2의 양자화, Q_2는 H.264의 양자화 단계를 나타내고 DCT_Conv (DCT Conversion)은 두 표준간의 DCT 변환(Conversion)을 수행한다. 좀 더 자세한 내용은 [6]을 참조하라. 본 논문의 초점이 DCT 상에서의 1/2 화소의 보정이므로 1/2 화소를 보정하는 CHPD(Compensation for Half-Pixel Difference)블록에 대하여 자세히 살펴보기로 한다.



(그림 3) 기존 표준에서 H.264로의 DCT기반 트랜스코더 구조

CHPD 블록에서 참조 블록을 얻어 오기 위해 Frame Store 버퍼에 저장되어 있는 DCT 상태의 이전 프레임을 이용해서 역 움직임 추정을 수행한다[7]. 기존의 연구에서 MPEG-2의 1/2 화소를 위한 역 움직임 추정은 8x8 블록을 기본단위로 사용하였다. 하지만 본 논문에서 제안하는 방법을 적용하기 위해서 8x8 블록이 아닌 최대 왼쪽 2픽셀, 오른쪽 3픽셀, 위쪽 2픽셀, 그리고 아래쪽 3픽셀이 더 필요하다. 즉 13x13 블록($\tilde{B}$)이 필요하다. 왜냐하면 H.264에서 1/2 화소를 구하기 위해선 자신을 포함한 주위 6개의 픽셀 값이 더 필요하기 때문이다. 수직, 수평 1/2 화소를 구하기 위하여 8x8 블록의 경계선에 있는 픽셀들은 추가적으로 2 또는 3개의 픽셀 정보가 더 필요하다. 13x13의 경우는 최대 9개의 8x8 블록과 겹칠 수 있다는 것을 (그림 4)에서 알 수 있다. 여기서 구한 $\tilde{B}$블록을 이용해서 보정 블록을 구한다. 수식에서 소문자는 픽셀상의 블록을 나타내며 대문자는 DCT 상의 블록임을 나타낸다.



(그림 4) 역 움직임 추정.
(입력 블록이 참조 프레임의 9개 블록과 겹친다.)

(그림 4)와 같은 경우에 $\hat{b}$을 구하기 위해서는 참조 프레임의 8x8 블록 9개의 정보를 이용을 하여야한다. 시프트 연산 행렬을 이용해서 겹치는 부분의 정보만을 $\hat{b}$의 해당 위치에 옮겨 놓을 수 있다. 이것은 수식 (8)과 같이 표현된다[8].

$$\hat{b}=\sum_{i=1}^{n} e_l\cdot b_i\cdot e_r \quad n\in S,\ S=\{2,3,4,6,9\} \quad (8)$$

$$=\underbrace{\begin{pmatrix} b_1 & 0 & \cdots & 0\\ 0 & 0 & \cdots & 0\\ 0 & 0 & \cdots & 0\end{pmatrix}}_{\hat{b}_1}+\underbrace{\begin{pmatrix} 0 & b_2 & \cdots & 0\\ 0 & 0 & \cdots & 0\\ 0 & 0 & \cdots & 0\end{pmatrix}}_{\hat{b}_2}+\cdots+\underbrace{\begin{pmatrix} 0 & 0 & \cdots & 0\\ 0 & 0 & \cdots & 0\\ 0 & 0 & \cdots & b_n\end{pmatrix}}_{\hat{b}_n}$$

n은 겹쳐지는 블록의 수를 나타낸다. b_i는 8x8 크기의 블록이고, e_l, e_r과 $\hat{b}$은 1/2 화소의 방향에 따라 변한다. e_l과 e_r 은 시프트 연산 행렬로 아래와 같이 나타낼 수 있다.

$$e_l=\begin{pmatrix} 0 & I_{l\times l}\\ 0 & 0\end{pmatrix}_{m'\times n'} \quad or \quad \begin{pmatrix} 0 & 0\\ I_{l\times l} & 0\end{pmatrix}_{m'\times n'} \quad (9)$$

$$e_r=\begin{pmatrix} I_{l\times l} & 0\\ 0 & 0\end{pmatrix}_{m''\times n''} \quad or \quad \begin{pmatrix} 0 & 0\\ 0 & I_{l\times l}\end{pmatrix}_{m''\times n''} \quad m',n',m'',n''\in\{8,13\}$$

수직 1/2 화소인 경우에 $\hat{b}$은 13x8의 크기를 가진다. e_l은 13x8 크기의 행렬을 e_r은 8x8크기의 행렬을 가지므로 m'은 13을 n',m''과n''은 8의 값을 가진다.

DCT는 분배법칙이 성립하므로 수식(8)은 수식(10)와 같이 나타낼 수 있다.

$$\hat{B}=\sum_{i=1}^{n} E_L\cdot B_i\cdot E_R \quad n\in S,\ S=\{2,3,4,6,9\} \quad (10)$$

수식(10)에서 구해진 $\hat{B}$와 수식(11)을 이용하여 MPEG-2 와 H.264에서의 1/2화소 차이 값을 찾는다.

$$\begin{cases} C=F_L^S\cdot\hat{B}\gg 5\\ C=\hat{B}\cdot F_R^S\gg 5\\ C=F_{HL}\cdot\hat{B}\cdot F_{HR}\gg 10-F_{ML}\cdot\hat{B}\cdot F_{MR}\gg 2\end{cases} \quad (11)$$

1/2 화소 차이 값은 1/2화소가 생기는 방향에 따라 수식(11)과 같이 달라진다. 첫 번째 수식은 수직 방향, 두 번째 수식은 수평 방향 그리고 세 번째는 수직,수평 1/2화소인 경우이다. f는 고정 된 상수 행렬 값이며 $f_{hr}=f_{hl}^T$, $f_{mr}=f_{ml}^T$이고, 그 값은 아래와 같다.

$$f_{hr}=\begin{pmatrix} 1 & -5 & 20 & 20 & -5 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 1 & -5 & 20 & 20 & -5 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & -5 & 20 & 20 & -5 & 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & -5 & 20 & 20 & -5 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & -5 & 20 & 20 & -5 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1 & -5 & 20 & 20 & -5 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -5 & 20 & 20 & -5 & 1 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -5 & 20 & 20 & -5 & 1\end{pmatrix} \quad f_{mr}=\begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0\end{pmatrix}$$

f_l^s와f_r^s은 아래와 같이 두 표준 간의 차이를 하나의 행렬로 나타내어 계산량을 줄였다.

$$f_l^s=f_{hl}-16\cdot f_{ml} \quad (12)$$

$$f_r^s=f_{hr}-16\cdot f_{mr}$$

이렇게 구해진 보정 블록(C)은 입력 블록($\tilde{B}$)에 더해 주면 보정이 완료된 새로운 residual 값을 구할 수 있다.

$$N=\tilde{B}-C \quad (13)$$

수식 (10), (11)에서 E_L , E_R , F_L^S , F_R^S은 고정된 행렬이기 때문에 수식(14)과 같이 나타낼 수 있으며 계산량을 줄 일수 있다.

$$C=\sum_{i=1}^{n} W_L\cdot B_i \quad n\in S,\ S=\{2,3,4,6,9\}$$

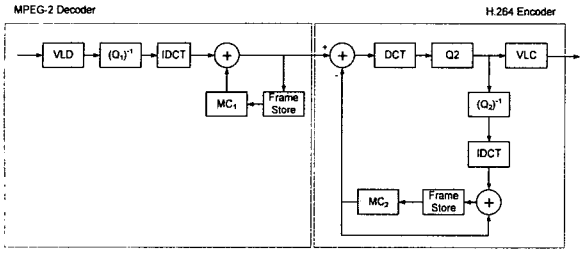
$$C=\sum_{i=1}^{n} B_i\cdot W_R \quad n\in S,\ S=\{2,3,4,6,9\}$$

$$W_L = F_L \cdot E_L \square 5 \qquad (14)$$
$$W_R = E_R \cdot F_R \square 5$$

새롭게 구한 residual 블록(N)은 두 표준간의 transform 차이를 수정하기 위해서 DCT 변환 단계의 입력으로 들어가게 된다.

4. 실험 결과

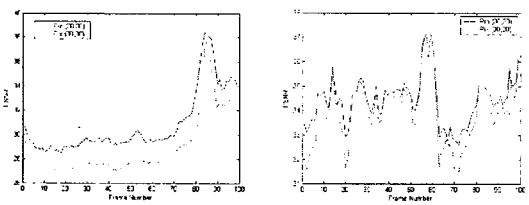
제안된 기법의 성능을 평가하기 위해 (그림 3)와 같은 구조의 트랜스코더를 구현하고 (그림 5)의 픽셀 기반 트랜스코더[3]와 화질 및 계산량을 비교하였다. 트랜스코더에 사용된 표준은 MPEG-2의 TM5[9]와 H.264의 JM8.2[10]이며 트랜스코더 구조에 맞게 수정하였다. 또한 실험의 편리를 위해 B 프레임은 사용하지 않았으며 IPPP의 순으로 부호화하였다.



(그림 5) 비교대상인 픽셀 기반 트랜스코더 구조

(그림 6)는 프레임 변화에 따른 PSNR 변화이다. 움직임이 비교적 많은 Football과 Foreman에 대해서 실험을 실시하였다. 실험 그래프에서도 볼 수 있듯이 움직임이 클수록 PSNR 차이가 커지는 것을 볼 수 있다. 평균 PSNR이 약 1.12dB 정도 더 높은 것을 확인하였다.

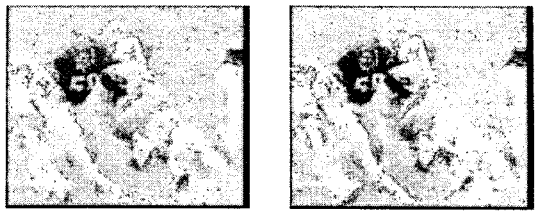
실험을 통하여 논문에서 제안하는 기법이 더 좋은 화질을 보이는 것을 확인하였다. 그 이유는 (그림 5)에서 볼 수 있듯이 픽셀 기반 트랜스코더는 두 번의 MC 과정을 거치게 되며 복호화를 위해 MC_1에서 MPEG-2 방식의 1/2화소를 계산을 한다. 이 과정에서 반올림 에러 값이 생긴다. 새로운 residual 값을 구하기 위해 MC_2에서 H.264 방식의 1/2 화소를 계산을 하게 되며 이 과정에도 반올림 값 에러가 생긴다. 즉 두 곳에서 반올림 에러 값이 생기게 된다. 하지만 제안 하는 트랜스코더는 행렬 연산으로 두 표준 사이의 차이 값을 바로 구하게 되므로 중간 과정에서 생기는 반올림 에러 값을 한번으로 줄였다. 또한 DCT 상에서 트랜스코딩이 이루어지므로 DCT, IDCT에서 생기는 오류 값 또한 제거 하였다.



(a) (b)

(그림 6) 프레임 변화에 따른 PSNR 변화
(a)Football Q1=30, Q2=30. (b)Foreman Q1=30, Q2=30

동일한 결과를 (그림 7)을 통해서도 확인 할 수 있다. (그림 7)의 경우 실제 이미지에서도 차이를 볼 수 있다. 예를 들어 (a)와 (b)의 경우 오른쪽의 사람 다리를 보면 주관적인 측면에서도 화질의 향상을 볼 수 있다.



(a) (b)

(그림 7) Q_1 과 Q_2 가 30일 때의 이미지 비교.
(a) 픽셀기반 트랜스코더 (b) 제안하는 트랜스코더

5. 결론

본 논문에서는 MPEG-2에서 H.264로의 트랜스코딩에서 DCT 상에서 1/2 화소 보정 방법을 제안하였다. 제안하는 기법에서는 DCT 상태의 참조 프레임에 특정 행렬 연산을 거쳐 두 표준 사이의 1/2 화소 차이를 구한 뒤 입력 블록에 더하여 보정작업을 수행하였다. 이 과정 에서 특정 행렬을 사용하며 미리 정해지는 상수 값이므로 메모리에 저장을 해 둘 수 있으므로 계산량이 감소한다. 제안하는 기법은 MC 과정에서 생기는 오류 값을 줄임으로써 화질에서 픽셀 기반 트랜스코더보다 나은 성능을 보이며 실제 실험에서도 그와 같은 사실을 확인할 수 있었다. 그러나 움직임이 많은 동영상은 1/2 화소가 많고 그 계산량이 픽셀기반 트랜스코딩보다 높아 전체 계산량이 증가함을 알 수 있었다. 따라서 실시간 변환보다는 오프라인에서 변환된 비디오의 화질 개선을 위한 목적으로 사용할 때 제안된 기법이 유용할 것으로 보이며 하드웨어의 발전에 따른 실시간 사용도 가능할 것으로 판단된다.

[1] T.Wiegand, G.J.Sullivan, G. Bjontegaard, and A.Luthra, "Overview of the H.264/AVC video coding standard,"IEEE Trans. Circuits Syst. Video Technol.,vol. 13, pp. 560-576, July 2003
[2] Hari Kalva. "Issues in H.264/MPEG-2 Video Transcoding". Consumer Communications and Networking Conference, 2004. CCNC 2004. First IEEE, 5-8 Jan. 2004
[3] A. Vetro, C. Christopoulos, and H. Sun, "Video Transcoding Architectures and Techniques: An Overview," IEEE Signal Processing Magazin, vol. 20,no.2,p.18-29, March2003
[4] ISO/IEC 13818-2:1995(E) pp.83~100
[5] Iain E.G. Richardson, "H.264 and MPEG-4 Video Compression" WILEY , 2003
[6] 강진미, "MPEG-2에서 H.264/AVC로의 변환을 위한 DCT 기반 트랜스코더 구조" 공학석사 학위논문 , 2005년 02월
[7] T. Shanableh and M. Ghanbari,"Hybrid DCT/pixel domain architecture for heterogeneous video transcoding" Signal processing: Image Communication, vol.18, pp.601-620, 2003
[8] S. F. Chang and D. G. Messerschmitt, "Manipulation and composing of MC-DCT compressed video," IEEE JNL. Select. Areas Commun,. Vol. 13, pp. 1-11, Jan. 1995.
[9] http://diml.yonsei.ac.kr/~wizard97/mpeg2/mpeg2v12.zip
[10] http://bs.hhi.de/~suehring/tml/download/old_jm/jm82.zip