MPEG-2/H.264 변환을 위한 1/2 화소 보정

권순영° , 이주경 , 정기동
부산대학교 컴퓨터공학과
{ksy2020°, jklee , kdchung }@melon.cs.pusan.ac.kr

Half Pixel Correction for MPEG-2/H.264 Transcoding

Soonyoung Kwon° , Jookyong Lee , Kidong Chung
Dept. of Computer Engineering, Pusan University

요 약

다양한 동영상 압축표준에서 압축효율을 높이기 위해 1/2 화소를 이용한다. 1/2 화소는 프레임 간 참조 시 압축 효율을 높이기 위해 프레임 내 화소를 연산하여 생성되는 가상의 값이며 이 연산식은 표준에 따라 다르다. MPEG-2에서 H.264로의 포맷 변환시 이 1/2 화소값의 차이로 인해 MPEG-2의 모션벡터와 움직임 보상된 값을 그대로 사용할 수 없게 된다. 본 논문에서는 MPEG-2의 모션벡터를 그대로 사용하고 DCT(Discrete Cosine Transform) 도메인에서 두 표준의 화소값의 차이를 보정하는 기법을 제안한다. 제안된 기법은 픽셀 도메인의 참조 블록을 이용하여 보정 할 위치를 찾고 두 표준의 1/2 화소 계산식의 차이를 이용하여 보정 할 값을 구하게 된다. 구해진 보정 값을 DCT하여 DCT 도메인의 현재 블록에 더하여 보정하게 된다. 이 기법은 모든 블록의 값을 완벽하게 보정할 수는 없지만 두 표준 간 차이값이 큰 1/2 화소를 보정할 수 있으며 IDCT와 DCT로 인한 화질 열화도 감소된다. 또한, DCT 상태에서 보정을 수행하므로 픽셀 도메인에서 보다 약 7%의 계산복잡도도 낮출 수 있다.

1. 서 론

최근 표준화가 완성된 새로운 동영상 압축 방식인 H.264는 ISO(International Organization for Standardization)와 ITU(International Telecommunication Union)전문가 그룹에 의해 개발되었다. 이 표준은 MPEG-2 압축 방식의 화질을 유지하면서 압축률을 50%로 낮추기 위해 기존의 압축 방식 외에도 가변 블록 움직임 보상, 복수 참조 영상, 그리고 1/4 화소 움직임 벡터 등을 추가하였다[1].

H.264는 우수한 압축 성능으로 다양한 사용자 환경에 사용될 것으로 보이며 기존의 압축 표준을 대체할 것으로 예상된다. 특히, HDTV(High Definition Television), DVD(Digital Versatile Disc)에 적합한 MPEG-2 데이터의 대용량으로 인해 H.264로의 급격한 전환이 이루어질 것으로 전망된다. 그러므로 기존의 MPEG-2 표준으로 압축된 데이터를 H.264로 변환할 필요성이 매우 높다.

기존의 MPEG-2 데이터를 H.264로의 변환할 때 모션벡터를 수정 없이 사용하려면 MPEG-2 데이터를 픽셀 도메인으로 완전히 복호화하고 H.264 방식의 1/2 화소를 생성한 후 움직임 보상을 수행해야한다. 1/2 화소는 여러 표준에서 프레임 간 참조의 압축 효율을 높이기 위해 프레임 내의 화소를 이용하여 만든 가상의 화소이며 표준마다 생성하는 연산식이 다르기 때문이다.

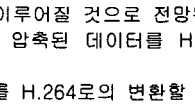
본 논문에서는 1/2 화소 보정을 위한 계산량을 줄이기 위해 DCT 도메인에서 두 표준 간 1/2 화소값의 차이를 보정하는 기법을 제안한다. 제안하는 기법에서는 1/2 화소를 생성하는 두 연산식의 차이를 이용하여 픽셀 도메인의 참조 프레임에서 오류가 큰 부분을 찾고 그 값에 DCT를 적용한다. DCT가 적용된 오류값과 현재 DCT 블록을 더하여 보정을 수행한다. 이 과정을 통하여 특정 값 이상으로 오류가 큰 값을 보정하게 된다. 보정된 블록은 H.264방식으로 DCT 변환된다.

본 논문의 구성은 다음과 같다. 먼저 2장에서 1/2화소 움직임 벡터에 대한 관련 연구를 한다. 이 연구를 통해 차이 값을 알아보고 이에 대한 보정을 3장에서 제안한다. 제안된 방법의 비교 분석을 4장에서 기술하고, 마지막으로 5장에서 결론을 맺는다.

2. 관련연구

2.1 MPEG-2와 H.264에서의 1/2화소

MPEG-2의 경우는 이웃하는 2개 또는 4개 화소 값의 평균값을 1/2 화소로 정한다[2][3]. (그림 1)에서 MPEG-2 방식의 1/2화소 계산의 예를 보여주고 있다.

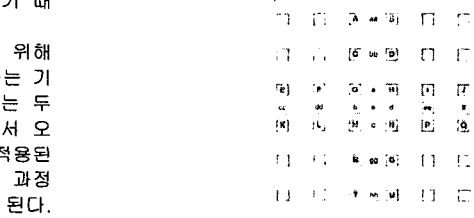


$$c = \frac{(C+D)}{2}$$

$$e = \frac{(A+B+C+D)}{4}$$

(그림 1) MPEG-2 1/2 화소 계산

H.264에서는 기존의 표준과는 달리 주위 6개의 화소들을 이용하여 1/2 화소를 계산한다[1]. (그림 2)는 H.264에서 1/2화소의 예를 보여주고 있다. 1/2화소인 b와 c를 구하기 위해서 (1)~(4)와 같이 6-taps 필터를 적용한다.



(그림 2) H.264 1/2 화소 계산

$$b_1 = (A-5C+20G+20M-5R+T) \quad (1)$$
$$c_1 = (K-5L+20M+20N-5P+Q) \quad (2)$$
$$b = (b_1+16) \gg 5 \quad (3)$$
$$c = (c_1+16) \gg 5 \quad (4)$$

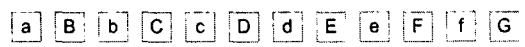
계산된 값을 0-255 사이가 되도록 자른다. 비슷한 방법으로 이웃한 모든 값이 1/2 화소인 e 경우는 (5), (6) 과정을 거치게 된다. 여기서 cc, dd, ee, d_1 은 b_1과 비슷한 방법으로 얻어진다.

$$e_1 = (cc-5dd+20b_1+d_1-5ee+ff) \quad (5)$$
$$e = (e_1+512) \gg 10 \quad (6)$$

2.2 MPEG-2와 H.264의 1/2 화소 값의 차이

위에서 보듯이 서로 다른 연산식으로 두 표준 간의 1/2화소 값의 차이가 발생하게 된다. 수평방향으로 계산되는 1/2 화소의 경우 (그림 3)과 같이 표시 할 수 있다. 그림에서 대문자는 원래 화소이며, 소문자는 1/2 화소를 의미한다. 이때 1/2 화소인 c의 경우 두 표준에서의 1/2 화소 값 차이는 (7)과 같이 나타낼 수 있다.

$$\frac{(A-5B+20C+20D-5E+F+16)}{32} - \frac{(C+D+1)}{2} = \Delta_h \quad (7)$$



(그림 3) 수평 1/2 화소 움직임 벡터에서 화소와 1/2화소

2.3 1/2 화소 모션벡터의 빈도

MPEG-2 표준으로 부호화되어 변환기에 입력되는 데이터 중에서 1/2 화소 모션벡터를 사용하는 매크로 블록의 빈도를 (표 1)에서 볼 수 있다. 비디오의 움직임 정도에 따라 다르지만 평균 38.7%의 발생 빈도를 보이고 있다. 프레임에서 38.7%의 매크로블록이 차이 값을 가진다면 1/2 화소 보정은 필수적이다.

[표 1] MPEG-2 매크로블록의 1/2 화소 모션벡터의 비율

비디오	Foreman	Akiyo	Carphone	Football	평균
발생빈도	46.8%	2.5%	38.7%	66.7%	38.7%

3. 1/2 화소 차이 값 보정 기법

1/2 화소 모션벡터를 이용하여 부호화된 MPEG-2 블록은 블록 내의 전 화소에서 보정이 필요하다. 그러나 미세한 차이 값은 DCT와 양자화 과정에서 무시될 수 있으므로 본 논문에서는 블록 내 모든 화소를 보정하지 않고 두 표준 간 화소 값의 차이가 주어진 값 이상인 경우에만 보정하는 기법을 제안한다.

다양한 실험에 의하면 두 표준에서 이웃하는 정수형 화소 값의 차이가 큰 부분에서 1/2 화소 값의 차이가 큰 것을 알 수 있었다. 이러한 특징을 이용하여 현재 DCT되어 있는 블록에서 이웃하는 화소간의 값의 차이가 큰 부분을 찾아야 한다. 그러나 이 경우 현재 블록을 픽셀도메인으로 IDCT하지 않고서는 계산이 거의 불가능 하므로 참조블록의 정보를 이용하기로 한다. 즉, 현재 블록이 참조하는 블록은 현재 블록과 특징이 비슷할 것이라고 가정하고 참조블록에서 이웃하는 화소의 값의 차이가 주어진 임계치보다 크다면 이 차이로 인하여 영향을 받을 1/2 화소의 위치를 찾고 값을 보정하게 된다.

예를 들어 (그림 3)에서 C, D 화소의 값의 차이가 주어진 임계 값 이상이라고 하자. 이때, C의 값을 x, D의 값을 y라고 하면 A, B는 C와 유사한 값을 가지며 E, F, G는 D와 유사한 값을 가진다고 가정한다. 유사한 값이지만 화소 간의 차이는 존재하므로 이 차이 값을 α_n이라고 한다. 1/2 화소인 a의 경우 두 표준에 의해서 구해진 1/2 화소의 차이는 (8)과 같다.

$$\frac{(x+\alpha_1)-5(x+\alpha_2)+20(x+\alpha_3)+20(x+\alpha_4)-5x+y+16}{32} - \frac{(x+\alpha_3)+(x+\alpha_3)+1}{2} = \left(\frac{-x+y}{32}\right) + C_1 \quad (8)$$

b의 경우는 (9)와 같다.

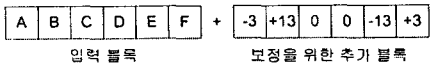
$$\frac{(x+\alpha_1)-5(x+\alpha_2)+20(x+\alpha_3)+20x-5y+(y+\alpha_4)+16}{32} - \frac{(x+\alpha_3)+y+1}{2} = \left(\frac{x-y}{8}\right) + C_2 \quad (9)$$

c의 경우는 수식 (10)과 같다.

$$\frac{(x+\alpha_1)-5(x+\alpha_2)+20x+20y-5(x+\alpha_3)+(x+\alpha_4)+16}{32} - \frac{(x-y+1)}{2} = C_3 \quad (10)$$

위의 식에서 볼 수 있듯이 C, D 화소에서 차이 값을 가지면 주위 1/2 화소에도 영향을 미친다는 것을 알 수 있다. 또한 수식 값을 이용해서 이웃하는 1/2 화소의 보정을 위한 값도 알 수 가 있다. 즉 C, D 값에 따라 1/2 화소인 b와 d, a와 e에 값을 더하거나 빼주어야 된다.

다음으로 해당 부분에 보정을 위한 값을 계산한다. 만약 1/2 화소인 b에 대해 두 표준간의 1/2화소 차이 값(Δ_h)이 10에서 15사이라고 하면 화소 간(C, D)의 차이($x-y$)는 수식(9)에 의해 $80+C$ 에서 $120+C$ 가 된다. 즉 두 화소 간의 차이 값을 구할 때 그 차이 값이 이 범위 안에 있다면 보정이 필요하다고 판단이 되어지고 C, D 화소 주위의 1/2 화소에 보정을 해주어야 된다. 이때 두 표준간의 차이 값이 10에서 15 사이이므로 중간 값인 13을 b에 더해주고 e에는 13을 빼준다. 즉 화소 값의 차이가 큰 화소의 위치에서 한 화소 건너 이웃하는 화소에는 13을 더하거나 빼주고 두 화소 건너 이웃하는 화소에는 3을 더하거나 빼주므로 해서 보정을 할 수 있다. 이때 C 와 D 중 어느 값이 큰 가에 따라서 더하는 값의 부호가 결정된다. (그림 4)에서 보정하기 위한 값의 예를 보여주고 있다. 여기서는 상수 C 값에 대한 고려는 하지 않는다. 다만 좀 더 정확한 실험을 위해서 4장에서 적절한 상수 C 의 값을 정하게 된다.



(그림 4) 입력 블록에 대한 보정 값의 예

보정이 필요한 위치와 보정 값을 구했다면 실질적인 보정이 이루어져야 한다. 입력으로 들어온 프레임을 블록 순서대로 변환하는 과정에서 해당 블록이 1/2 화소 움직임 벡터일 경우 그 블록(b)은 보정된다. 보정을 위해 더하거나 뺄 값을 임의의 블록(s)에 저장하게 된다. 저장 되어진 블록(s)은 원래 블록에 더해져서 보정이 된 블록(b')을 구하게 된다.

이 과정을 (11)으로 표현 할 수 있다.

$$b' = b + s = \underbrace{\begin{pmatrix} b_{11} b_{12} \cdots b_{18} \\ b_{21} b_{22} \cdots b_{28} \\ \vdots \quad \vdots \quad \ddots \quad \vdots \\ b_{81} b_{82} \cdots b_{88} \end{pmatrix}}_{pixel\ domain} + \underbrace{\begin{pmatrix} s_{11} s_{12} \cdots s_{18} \\ s_{21} s_{22} \cdots s_{28} \\ \vdots \quad \vdots \quad \ddots \quad \vdots \\ s_{81} s_{82} \cdots s_{88} \end{pmatrix}}_{\text{보정을 위한 추가적인 값}} \quad (11)$$

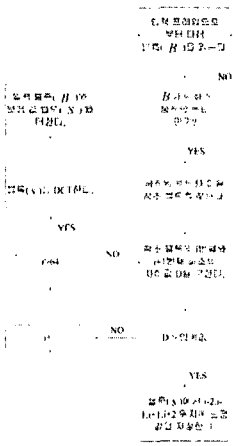
(11) 대해 양변에 DCT를 해주면 수식(12)과 같다.

$$DCT(b') = DCT(b+s) \quad (12)$$

DCT는 분배 법칙이 성립하므로 우리는 수식(13)을 얻을 수 있다. DCT가 수행 되이진 블록을 알파벳 대문자로 표현하면 수식(14)을 얻을 수 있다.

$$DCT(b') = DCT(b) + DCT(s) \quad (13)$$
$$B' = B + S \quad (14)$$

입력으로 들어오는 B값에 보정 값을 저장하고 있는 블록(s)를 DCT 하여 구해진 블록(S)를 더해서 보정이 이루어진다.
(그림 5)에서 1/2 화소 보정의 전체적인 순서도를 보여주고 있다.

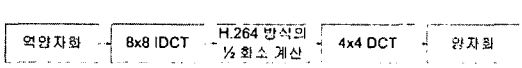


(그림 5) 1/2 화소 보정 알고리즘 순서도

4. 실험 결과

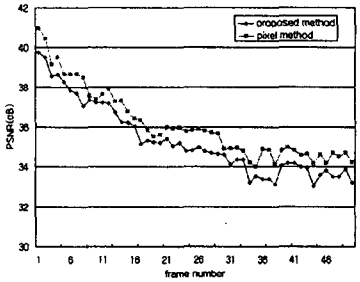
MPEG-2 실험에 사용된 부호화기는 TM5이고 H.264 실험에 사용된 부호화기는 JM8.2이다[4][5]. 실험영상은 Foreman 영상 50장으로 프레임율은 30Hz이다. B 프레임은 사용하지 않았고, 실험은 영상에 대하여 IPPP의 순으로 부호화하였다.

제안된 알고리즘의 비교대상은 (그림 6)의 픽셀 도메인에서의 보정 방식이다. 객관적인 성능비교를 위하여 복호 화된 데이터의 PSNR을 측정하였으며 MPEG-2와 H.264로의 양자화 값은 동일하게 수행하였다.



(그림 6)픽셀 도메인에서의 보정 단계

(그림 7)은 두 가지 알고리즘에 대하여 모의실험 결과를 그래프로 표현한 것이다. 큰 부호화 손실 없다는 결과를 보여주고 있다.



(그림 7) Foreman 에서의 두 알고리즘 PSNR 분석

시간 복잡도를 살펴보기 위해 두 방법(픽셀 도메인에서의 보정과 제안 기법)에 대한 연산 횟수를 계산 하였다. 우선, 픽셀 도메인에서 보정을 하는 알고리즘의 경우 연산 횟수는 (15), (16)과 같다.

$$1792Ml + 1600Add + 192 + 320Shi \quad (15)$$
$$1812Ml + 1625Add + 202 + 345Shi \quad (16)$$

(15)는 움직임 벡터의 x 또는 y 좌표중 하나만 1/2 화소 움직임벡터를 가진 경우이고 (16)는 움직임 벡터 x, y 좌표 둘 다 1/2 화소 움직임 벡터를 가진 경우이다. 후자의 경우 6 taps 필터를 적용한 중간 값이 계산 과정에 필요하므로 (15)보다 더 많은 연산이 필요하다.
본 논문에서 제안하는 과정의 경우의 연산 횟수는 (17)와 같다.

$$1532Ml + 1280Add + 240 + 240Com \quad (17)$$

제안하는 알고리즘은 방향에 상관없이 연산 횟수가 동일하다. 여기서 Mul 은 곱셈 연산을 , Add는 덧셈 연산을 , Sub는 뺄셈 연산을, Shi는 오른쪽 시프트 연산을 om 는 비교 연산을 나타낸다.
제안하는 기법의 계산 복잡 도를 곱셈 연산을 기준으로 보면 약 7%의 연산이 줄어든 것을 알 수 있다.

5. 결론

H.264에서의 1/2 화소 계산 방법은 기존 표준과는 달리 6-taps 연산을 수행한다. 본 논문에서는 MPEG-2와 H.264 포맷 변환 시 1/2 화소값 차이를 보정하기 위한 기법을 제안했다. 본 논문에서는 IDCT와 DCT 수행으로 인한 화질 열화를 줄이고 계산 복잡도를 줄이기 위해 DCT 도메인에서 차이 값을 보정하는 방법을 제안하였다. 실험 결과 큰 부호화 손실 없이 약 7% 정도의 계산량 감소를 얻을 수 있다. 추후 더 많은 실험을 통하여 상수 C 값에 대한 연구와 양자화 과정까지 고려하여 계산 량을 더 많이 줄이고 화질의 열화가 없는 알고리즘에 대한 연구가 진행되어야 한다.

참고 문헌

[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]ISO/IEC 13818-2:1995(E) pp.83~100
[3]Iain E.G. Richardson, "H.264 and MPEG-4 Video Compression" WILEY , 2003
[4]http://diml.yonsei.ac.kr/~wizard97/mpeg2/mpeg2v12.zip
[5]http://bs.hhi.de/~suehring/tml/download/old_jm/jm82.zip