논문 2004-41SP-5-29

MPEG-4에서 H.264로 트랜스코딩

(MPEG-4 to H.264 Transcoding)

이 성 선*, 이 영 렬**

(Sung-Sun Lee and Yung-Lyul Lee)

요 약

본 논문에서는 30 Hz 프레임 율의 MPEG-4 simple profile 비디오 비트스트림을 15 Hz 프레임 율을 갖는 H.264 baseline profile 비디오 비트스트림으로 변환하는 트랜스코딩을 제안한다. MPEG-4의 블록 모드(block mode)와 움직임 벡터(Motion Vector) 정보를 H.264에서 이용 가능하도록 블록 모드 변환을 수행하고, MPEG-4의 움직임 벡터 보간을 이용하여 H.264에서 움직임 예측(Motion Estimation) 없이 정수 화소 단위로 움직임 벡터를 찾는 3가지 움직임 벡터 보간 (Motion Vector Interpolation) 방법을 실험한다. 이와 같은 방법을 이용해서 움직임 예측 시 소요되는 계산량을 줄이고 낮은 대역폭에서 심각한 화질 열화가 없는 트랜스코더를 제안한다. 실험 결과 제안된 방법은 직렬 화소영역 트랜스코딩에 비해 신호 대 잡음비 (PSNR: peak signal to noise ratio)는 실험 영상에 따라 높은 비트율에서는 0.2dB에서 낮은 비트율에서 0.9dB의 손실이 있으나 전체 수행 시간은 3.2배에서 4배 빨라진다.

Abstract

In this paper, a transcoding method that transforms MPEG-4 video bitstream coded in 30 Hz frame rate into H.264 video bitstream of 15 Hz frame rate is proposed. The block modes and motion vectors in MPEG-4 is utilized in H.264 for block mode conversion and motion vector (MV) interpolation methods. The proposed three types of MV interpolation method can be used without performing full motion estimation in H.264. The proposed transcoder reduces computation amount for full motion estimation in H.264 and provides good quality of H.264 video at low bitrates. In experimental results, the proposed methods achieves 3.2-4 times improvement in computational complexity compared to the cascaded pixel-domain transcoding, while the PSNR (peak signal to noise ratio) is degraded with 0.2-0.9dB depending on video sizes.

Keywords: transcoding, motion estimation, motion vector interpolation, MPEG-4, H.264

I. 서 론

최근 멀티미디어 환경에서의 비디오 전송이 폭넓게 이용되고 있다. 그 중에서 비디오는 멀티미디어 통신에서 많은 대역폭을 차지하고 있다. 따라서 이러한 비디오 전송이 빠르고 더 좋은 화질(Quality)을 가능하게 하기 위한 다양한 멀티미디어 압축 표준이 만들어지고 있으며 현재 기존의 압축 표준[1-5]보다 적은 비트 양으로 더 좋은 화질을 제공하는 H.264 (MPEG-4 AVC (Advanced Video Coding))의 표준화가 완료되었다[6].이러한 영상 압축 표준의 다양화로 이전에 압축된 비디오 비트스트림을 다른 압축 표준의 비디오 비트스트림으로 변환하는 트랜스코딩[7-9]이 연구되고 있다. 트랜스코딩에 많은 연구 분야가 있지만 네트워크의 발전은 대역폭이 서로 다른 네트워크 상황에서도 QoS(Quality of Service)를 만족시킬 수 있는 비트율(bit rate)로 변화시키는 트랜스코딩이 필요하게 되었다. 본 논문에서는 현재 널리 사용되고 있는 MPEG-4 SP (Simple Profile) 비디오 비트트림을 H.264 BP (Baseline Profile) 비디오 비트스트림으로 트랜스코딩할 때, 매크로 블록 (Macro-block, MB) 모드 변환과 프레임 스킵을 통한 비트율

* 학생회원, ** 정회원, 세종대학교 인터넷공학과
(Dept. of Internet Engineering, Sejong University, DMS Lab. yllee@sejong.ac.kr)

※ 본 논문은 Tcom & dtvro 위탁 연구 지원에 의해 수행되었습니다.
접수일자: 2004년2월2일, 수정완료일: 2004년6월10일

변화[10-13]를 이용해서 MPEG-4의 움직임 벡터를 H.264에서 재사용하는 3가지 움직임 벡터 보간 방법을 고찰한다. 실험결과는 직렬 화소영역 트랜스코딩과 제안한 방법의 실험 결과를 비교한다.

본 논문은 다음과 같이 구성된다. Ⅱ장에서는 MPEG-4와 H.264의 블록 모드와 움직임 예측 방법의 차이점, 직렬 화소영역 트랜스코딩, 제안된 블록 모드 변환에 대해 소개하며 Ⅲ장에서는 프레임 율이 1/2로 줄어든 상황에서 움직임 벡터 보간 방식을 사용해 움직임 벡터를 찾는 3가지 방법에 대해 소개한다. Ⅳ장에서 직렬 화소영역 트랜스코딩과 본 논문에서 제안한 방법을 사용한 트랜스코딩 실험 결과를 비교하며, 마지막으로 결론을 맺는다.

Ⅱ. 제안된 블록 모드 변환

MPEG-4와 H.264는 압축 방식에서 많은 차이점을 갖고 있으며 한 개의 16×16 MB내의 블록 모드 종류와 움직임 예측에 있어서도 서로 다른 방법을 사용하고 있다. 그림 1(a)에서와 같이 MPEG-4는 한 개의 MB에 대하여 16×16, 8×8 두 종류의 Inter블록 모드와 Intra, Skip 모드가 있으며 ±16 검색 영역(search range)에서 정수 화소와 1/2 화소 움직임 예측을 이용하지만 H.264는 Intra 16×16, Intra 4×4, Skip 모드와 그림 1(b)에서처럼 하나의 MB를 16×16, 16×8, 8×16, 8×8 블록으로 나누고 다시 8×8 블록을 8×4, 4×8, 4×4 블록으로 나누어 7가지 모드의 가변 블록 단위 움직임 예측을 한다. 그림 2와 같이 각 블록 모드에 대해 정수 화소 단위에서는 ±16 검색 영역, 1/2화소 단위에서는 정수 화소 단위에서 찾은 움직임 벡터 주변의 1/2 화소 단위의 ±1 화소를 검색하여 1/2 화소 단위의 최적 움직임 벡터를 찾고, 1/4 화소 단위에서는 1/2 화소 단위에서 찾은 움직임 벡터 주변의1/4 화소 단위의 ±1 화소를 검색하여 1/4 화소 단위의 최적 움직임 벡터를 찾는다. 또한 Intra 16×16 부호화 모드에는 4가지 방식, Intra 4×4 부호화 모드는 9가지 방식이 있다.

그 후 PSNR 향상을 위하여 각각의 MB에 대하여 가능한 부호화 모드 중 최적의 블록 모드를 결정하기 위해 선택사양(Non-normative)인 율-왜곡 최적화 (RDO: rate distortion optimization)을 통해 각 MB에 대한 최적의 블록 모드를 결정하게 된다. 위에서 설명한 것처럼 H.264 부호기는 모든 블록 모드에 대해 블록 기반 ME를 수행하고 율-왜곡 최적화를 통해 블록 모드를

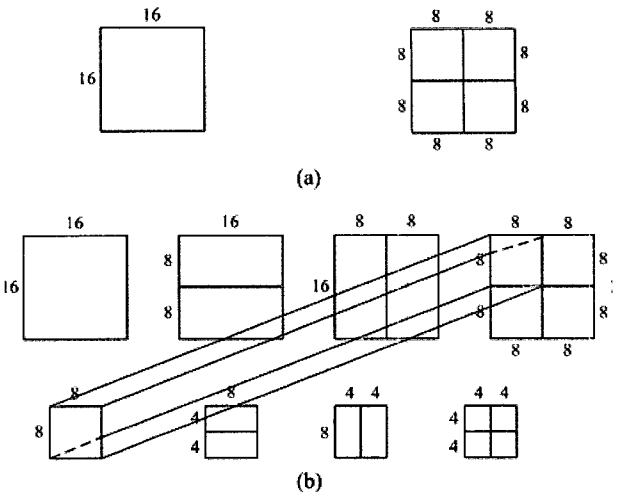


그림 1. 움직임 예측시 사용하는 한 개의 MB내에 가변 블록 모드 (a) MPEG-4의 Inter 블록 모드 (b) H.264의 Inter 블록 모드

Fig. 1. Variable blocks used for motion estimation in one MB; (a) Inter block mode of MPEG-4 (b) Inter block mode of H.264.

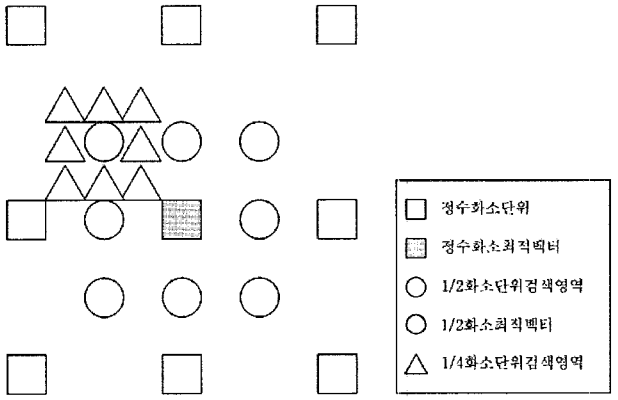


그림 2. H.264의 1/4화소 움직임 예측

Fig. 2. H.264 quarter_pel motion estimation.

결정함으로써 다른 형식의 표준에 비해 화질과 압축 효율 면에서는 좋으나 계산 복잡도가 증가한다.

트랜스코딩의 기본이 되는 직렬 화소영역 트랜스코딩과 제안하는 MPEG-4와 H.264의 트랜스코딩에 있어서 각 MB에 대한 블록 모드 변환 방법을 소개한다. MPEG-4의 비트스트림을 H.264 비트스트림으로 변환할 수 있는 가장 간단한 방법은 그림 3과 같이 직렬 화소영역 트랜스코딩을 수행하는 방법이다. MPEG-4로 압축된 비트스트림을 복원하고 복원된 영상을 다시 H.264의 입력 영상으로 H.264 비트스트림을 구성한다. 그러나 이러한 방법은 MPEG-4에서 수행했던 모든 프레임에 대한 움직임 예측을 H.264에서 또다시 수행해야 하기 때문에 뛰어난 화질 성능을 얻을 수 있는 반면에 많은 계산 복잡 도로 인해 실시간 상황에서 문제점이

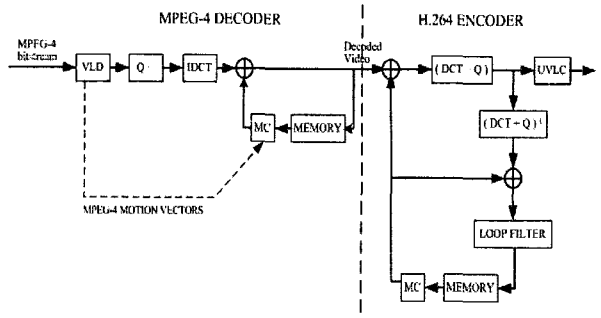


그림 3. 직렬 화소영역 트랜스코딩
Fig. 3. Cascaded pixel-domain transcoding.

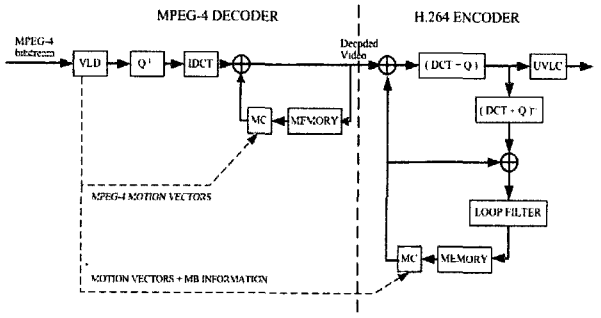


그림 4. 제안된 화소영역 트랜스코딩
Fig. 4. Proposed pixel-domain transcoding.

있을 수 있다. 이러한 문제점을 보완하기 위해서 그림 4에 MPEG-4에서 사용한 각 MB에 대한 블록 모드 정보와 움직임 벡터를 H.264에서 재사용하는 움직임 벡터 보간 방법을 이용하여 계산 복잡도를 줄이고 프레임 스킵을 통해 비트율을 줄이는 제안된 방식의 그림을 보인다. 본 논문에서 제안하는 방법은 제한된 대역폭 및 낮은 처리능력(processing power)를 갖는 모바일 단말과 같은 응용에 사용할 수 있다.

MPEG-4 비트스트림 복호 과정에서 나온 블록 모드 정보를 H.264에서 재사용하기 위하여 블록 모드 변환을 수행한다. H.264는 각 MB내의 가변 블록에 대해 움직임 예측을 수행하고 가장 최적의 MB 모드를 결정하므로 MPEG-4의 MB 정보를 이용하면 움직임 예측을 위한 블록 모드의 수를 줄임으로써 계산 복잡도를 낮출 수 있다. 그림 5에서와 같이 MPEG-4의 Inter 블록 모드가 16×16이면 H.264에서는 16×16, 16×8, 8×16 블록 움직임 예측 및 Intra 16×16만 수행하도록 하고, MPEG-4에서 Inter 블록 모드가 8×8이면 H.264에서는 8×8, 8×4, 4×8, 4×4 블록 움직임 예측과 Intra 4×4만 수행한다. MPEG-4 MB가 Skip 모드인 경우 H.264에서는 Skip 모드와 Intra 16×16, Inter 16×16 움직임 예측을 수행하고, MPEG-4에서 Intra 모드인 경우 H.264에서 Intra 4×4와 intra 16×16 만을 수행한다. 그림 5의 블록 모드 변환은 실험적으로 결정되었다. H.264에서 Intra

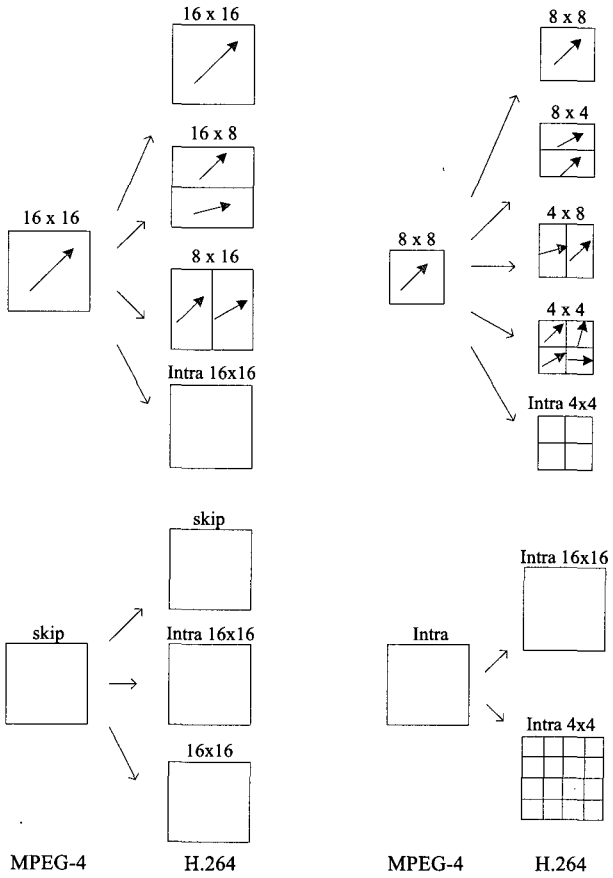


그림 5. MPEG-4 블록 모드의 H.264 블록 모드 변환
Fig. 5. Proposed pixel-domain transcoding.

4×4 블록의 예측 모드는 9가지, Intra 16×16 블록의 예측 모드는 4가지가 존재한다. Intra 4×4 블록과 Intra 16×16 블록 각각의 예측 모드들에 대해 예측 화소들의 차분을 구한 후 (DCT+Q)와 $(DCT+Q)^{-1}$를 수행하여 율-왜곡(rate-distortion)이 가장 낮은 최적의 모드를 결정하는 RDO를 수행할 경우 복잡도가 높기 때문에 복잡도를 낮추기 위해서 RDO 대신 각각의 모드에 대해 평균 제곱 오차 (MSE: mean square error)가 가장 작은 모드를 Intra 4×4 및 Intra 16×16의 모드로 결정한다.

제안한 방법을 이용해서 MPEG-4의 블록 모드 정보를 H.264에 적용, H.264 복잡도의 80% 이상을 차지하는 움직임 예측과 율-왜곡 과정에서 소요되는 시간을 줄일 수 있다.

III. 변화된 프레임 율에서 움직임 벡터 보간

가장 최적의 성능을 발휘하는 트랜스코딩은 입력되는 비트스트림을 복호화 하고 복호화된 영상에 대해 움직임 예측을 다시 수행하여 압축을 하는 직렬 화소영역 트랜스코딩이지만 이런 방법은 움직임 예측을 다시 수

행하게 되어 계산 복잡도가 높아지게 된다. 계산 복잡도를 줄이기 위해 트랜스코딩 시 움직임 벡터를 다시 사용하고 양자화 계수(Quantization parameter)를 크게 하여 네트워크 대역폭(Network bandwidth)에 맞도록 비트율을 조정할 수도 있으나 이 장에서는 프레임 율을 1/2로 낮춤으로써 비트율을 조정하고 MPEG-4의 움직임 벡터를 H.264에서 재사용하는 움직임 벡터 보간 방법을 제안한다.

H.264에서 프레임 율을 1/2로 줄이게 되면 MPEG-4에서 가져온 움직임 벡터를 재사용하기 위한 과정이 필요하다. 그림 6(a)는 H.264에서 스킵된 프레임에 대해 MPEG-4의 움직임 벡터를 이용하여 움직임 벡터 보간 과정을 보여주고 있다. MPEG-4에서 H.264로 트랜스코딩 시 프레임 율을 1/2로 줄이는 경우, H.264의 (n)번 프레임은 MPEG-4 프레임 (2n)번에 해당하고 MPEG-4의 (2n-1)번 프레임은 H.264의 (n)번 프레임과 (n-1)번 프레임의 중간에 위치하게 된다. MPEG-4의 (2n-1)번 프레임에서 한 블록의 움직임 벡터는 (2n-2)번 프레임을 참조해서 나온 것이다. MPEG-4의 (2n)번 프레임에서 한 블록의 움직임 벡터와 스킵된 (2n-1)번 프레임의 한 블록의 움직임 벡터를 더하면 H.264에서 (n-1)번 프레임을 참조한 현재 프레임의 한 블록의 움직임 벡터를 구할 수 있다. 다음은 스킵된 프레임에서 한 블록의 움직임 벡터를 MPEG-4에서의 한 블록의 움직임 벡터를 이용해서 재사용하는 3가지 움직임 벡터 보간 방법을 제안한다.

1. 움직임 벡터들의 양방향 보간 (Bilinear interpolation of motion vectors, BI)

MPEG-4는 움직임 예측 시 각 MB에 대하여 16×16, 8×8 블록 모드가 존재한다. 따라서 16×16 블록 움직임 벡터라 할지라도 8×8 블록 단위 움직임 벡터 4개로 나누어서 표현할 수 있으므로 모든 프레임을 8×8 블록 단위 움직임 벡터로 표현이 가능하다. 그림 6(b)에 MPEG-4 (2n) 프레임에서 움직임 벡터를 추적하여 skipped 프레임에 겹친 8×8 블록을 보인다. MPEG-4에서 움직임 벡터가 8×8 블록 모드로 표현이 가능하기 때문에 각 8×8 블록의 움직임 벡터는 다음과 같이 추적될 수 있다.

$$MV = \frac{\sum_{i=1}^{4}(w_i \times h_i \times MV_i)}{\sum_{i=1}^{4}(w_i \times h_i)} \qquad (1)$$

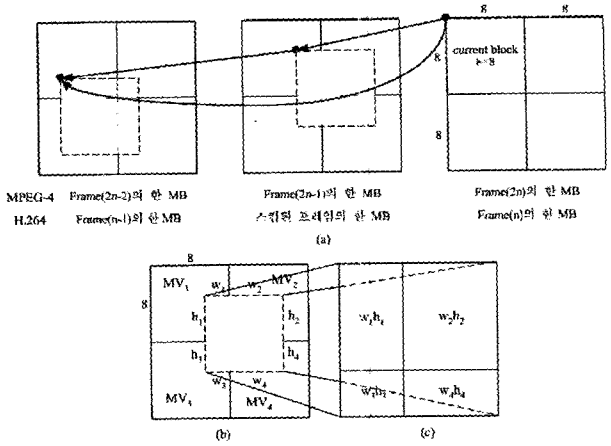


그림 6. MPEG-4에서 프레임 스킵 시 움직임 벡터들의 양방향 보간을 위한 겹친 블록들

Fig. 6. Overlapped blocks for bilinear interpolation of motion vectors when frame-skip happens in the MPEG-4.

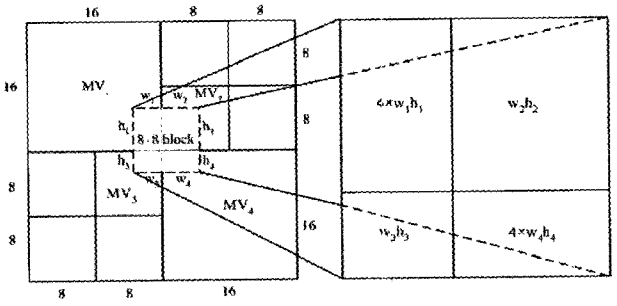


그림 7. MPEG-4 블록 모드에 따른 움직임 벡터들의 가중치 양방향 보간을 위한 겹친 블록들

Fig. 7. Overlapped blocks for weighted bilinear interpo-lation of motion vectors depending on the block modes in MPEG-4.

식 (1)에서 w_i, h_i는 8×8 블록과 i번째 겹친 블록간의 수평, 수직 겹친 길이, MV_i는 i번째 겹친 블록의 움직임 벡터를 나타낸다. 이에 대한 자세한 내용을 그림 6(b)와 6(c)에 보였다. 결국 MPEG-4의 8×8 블록 코드로부터 H.264의 16×16, 16×8, 8×16, 8×8에 대응하는 움직임 벡터를 구할 수 있다.

2. 움직임 벡터들의 가중치 양방향 보간 (Weighted bilinear interpolation of motion vectors, WBI)

MPEG-4의 블록 모드는 16×16과 8×8 두 종류를 가지고 있다. 16×16 블록은 8×8 블록보다 4배 많은 화소에 대한 정보를 가지고 있으므로 그림 7과 같이 Bilinear interpolation을 할 때 8×8 블록 모드보다 4배 더 많은 비중을 주고 계산을 한다. 겹치는 부분은 작더라도 그 부분이 16×16 블록 모드의 움직임 벡터라면 4배를 해줌으로써 최종 계산된 움직임 벡터에서의 비중

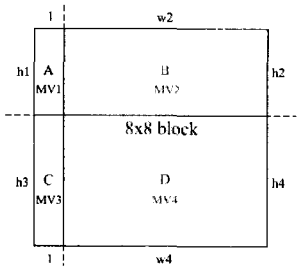


그림 8. 겹친 블록의 상태에 따른 움직임 벡터 계산
Fig. 8. Overlapped blocks for constrained bilinear interpolation of motion vectors.

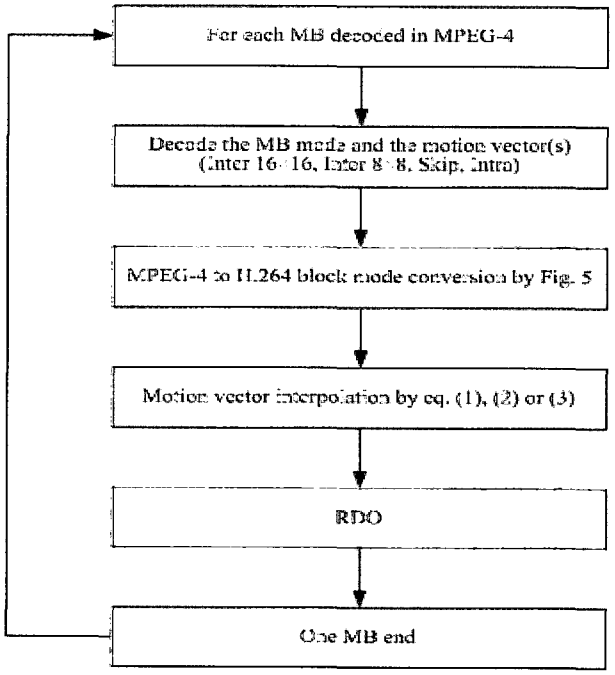


그림 9. 제안된 화소 영역 트랜스코딩 흐름도
Fig. 9. Flow chart of proposed pixel-domain transcoding.

이 같은 면적의 8×8 블록 모드의 움직임 벡터보다 크게 한다.

최종 움직임 벡터는 다음과 같이 계산 될 수 있다.

$$MV = \frac{4w_1h_1MV_1 + w_2h_2MV_2 + w_3h_3MV_3 + 4w_4h_4MV_4}{4w_1h_1 + w_2h_2 + w_3h_3 + 4w_4h_4} \quad (2)$$

3. 움직임 벡터들의 제약적 양방향 보간 (Constrained bilinear interpolation of motion vectors, CBI)

양방향 보간을 할 때 각 블록들이 차지하는 영역은 H.264의 8×8 블록의 위치에 따라 다르다. 그러나 w_i 또는 h_i, (i=1, 2, 3, 4) 중에서 1이라는 값이 존재할 경우 그 블록의 움직임 벡터 영향력은 극히 작으며 인접한 주변 블록의 넓이도 작아지고 그 블록들의 움직임 벡터의 영향력도 적다. 그림 8에서 겹친 블록 A와 C는 움직임 벡터 고려 대상 블록에서 제외되고 B와 D 블록만이 움직임 벡터 고려 대상이 된다. 따라서 w_i=1 또는 h_i=1인 겹친 블록의 경우 움직임 벡터 보간 시 제외한다. 작은 영역 부분을 차지하는 움직임 벡터를 제거함으로써 최종 계산된 움직임 벡터에 미칠 영향을 제거한다.

최종 움직임 벡터는 다음과 같이 계산 될 수 있다.

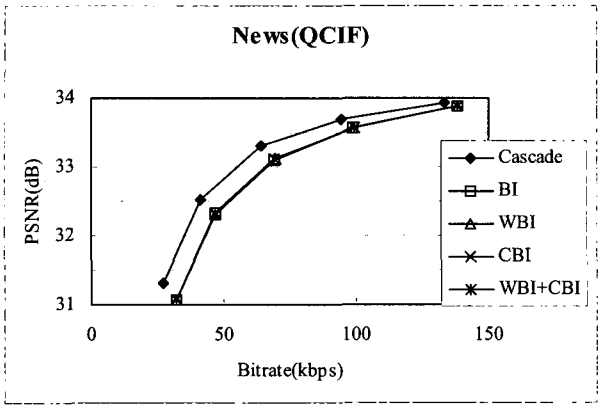
$$MV = \frac{w_2h_2 \cdot MV_2 + w_4h_4 \cdot MV_4}{w_2h_2 + w_4h_4} \quad (3)$$

이상 3가지 움직임 벡터 보간 방법을 사용하여 MPEG-4의 정수 화소 단위에서의 움직임 벡터를 H.264에서 재사용하여 H.264 정수 화소 단위에서 검색 영역을 ±2로 줄임으로써 움직임 예측에 필요한 계산량은 줄이면서 좋은 성능의 화질을 얻을 수 있다.

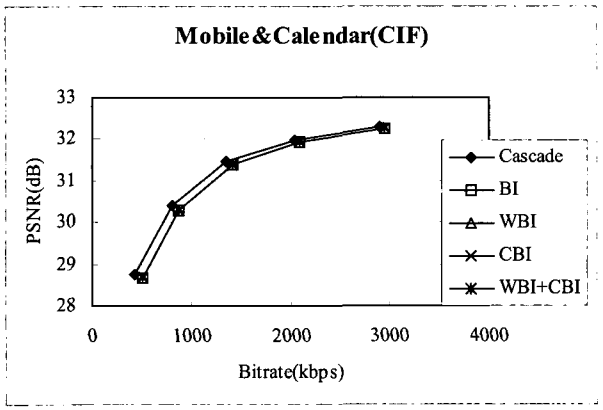
그림 9는 제안된 화소 영역 트랜스코딩의 전체 흐름도를 보여주고 있다. H.264 한 개의 MB에 대해서 MPEG-4의 MB 모드에 따른 블록 모드 변환을 수행한 후 변환 된 블록 모드에 따라 MPEG-4의 움직임 벡터 정보로 제안된 움직임 벡터 보간 방법을 이용하여 스킵된 프레임의 움직임 벡터를 찾는다. 이렇게 찾아낸 각 블록 모드에 대해 RDO를 수행하여 한 개의 MB의 최종 블록 모드를 결정한다.

IV. 실험 결과

본 실험은 MPEG-4 SP를 지원하는 MoMuSys 복호기[14]와 H.264 (MPEG-4 Part10 AVC) BP를 지원하는 JM42 (Joint Model) 부호기[15]를 이용하여 수행되었다. 실험은 Pentium-IV 2.66 GHz에서 H.264 개발 시 실험 영상으로 사용한 각각 30 Hz 프레임율로 300장 저장된 7개의 QCIF(Quarter Common Interface Format, 176×144)와 CIF(Common Interface Format, 352×288) 영상을 이용했다. 실험한 JM42 코드의 기본 조건은 가변 블록 크기(16×16, 16×8, 8×16, 8×8, 8×4, 4×8, 4×4)의 움직임 예측 및 보상, ±16의 정수단위 움직임 벡터 검색 영역, 1/4화소 움직임 예측 및 보상, 4×4 Integer DCT, RDO(Rate-Distortion Optimization)를 사용했다. 각 영상의 첫 번째 장은 Intra frame이고 나머지는 모두 Predictive frame으로 압축하였으며, MPEG-4에서 30 Hz 프레임 율의 영상을 H.264에서 프레임을 1장씩 스킵하면서 15 Hz의 프레임 율로 각 영상 300장에 대

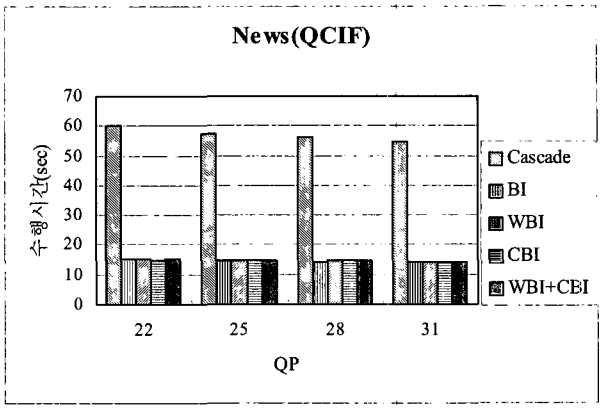


(a)

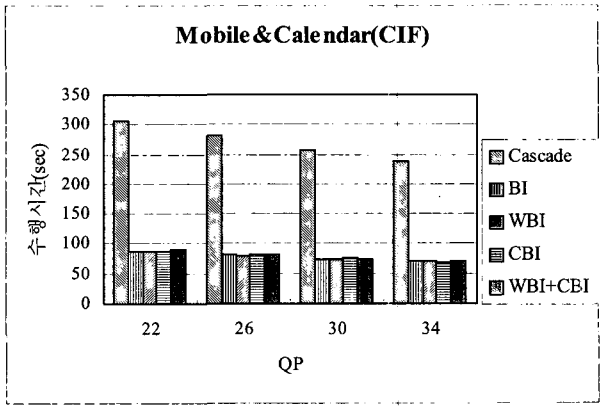


(b)

그림 10. 율-왜곡 곡선 (a) "News" (b) "Mobile&Calendar"
Fig. 10. Rate-Distortion curve (a)"News" (b) "Mobile & Calendar".



(a)



(b)

그림 11. 직렬 화소영역 트랜스코딩과 제안된 트랜스코딩 시간 비교 (a) "News" (b) "Mobile&Calendar"
Fig. 11. Execution time comparison of the cascaded pixel-domain transcoding to the proposed transcoding (a) "News" (b) "Mobile&Calendar".

해 실험하였다. H.264에서 움직임 벡터 보간시 정수 화소 단위에서 MPEG-4의 움직임 벡터를 본 논문에서 제안된 움직임 벡터 보간 방법들을 이용하여 후보 벡터를 찾고 다시 후보 벡터에서 ±2의 검색 영역으로 움직임 벡터를 찾은 후 1/2 화소와 1/4 화소에서 각각 주위 8개의 화소만을 검색하여 1/4화소 움직임 벡터를 찾았다.

그림 10의 (a)와 (b) 두 그래프는 직렬 화소영역 트랜스코딩과 본 논문에서 제안한 블록 모드 변환을 이용한 BI, WBI, CBI, WBI+CBI 방법에 의한 실험결과를 두 영상에 대한 율-왜곡 곡선 (Rate-Distortion curve)으로 보여주고 있다. 그림 10(a)의 QCIF 크기의 Nwes 영상의 그래프에서 각 방법을 비교해 보면 비슷한 성능을 보여주고 있으며 직렬 화소영역 트랜스코딩 성능과 제안된 각 방법의 성능을 비교해 보면 높은 비트율에서는 0.2dB, 낮은 비트율에서는 최대 0.7dB의 PSNR 손실을 보여주고 있다. 그림 10(a)로부터 본 논문에서 제안한 방식은 약 70kbps에서 약 33.1dB의 성능을 보이고 있는데, 이는 모바일 단말용 응용을 위하여 적당한 화질을 제공한다고 할 수 있다. 그림 10(b)의 CIF 크기의 Mobile&Calendar 영상에서는 제안된 방법들간에 거의 비슷한 성능을 보여주고, 제안된 방법들과 직렬 화소영역 트랜스코딩은 높은 비트율에서는 0.2dB, 낮은 비트율에서는 0.9dB의 차이를 보이는 것을 알 수 있다.

그림 11에서는 이들 두 영상에서 사용한 트랜스코딩 방법들에 대한 수행시간을 보여준다. 직렬 화소영역 트랜스코딩과 비교했을 때 각 제안된 방법들이 QCIF 영상에서는 3.2배에서 4배, CIF 영상에서는 3.3배에서 4배 빠른 수행시간을 보여주며 제안된 방법들 간에 거의 비슷한 수행시간을 보여준다. 그림 11의 세로축은 압축된 300장의 MPEG-4 비트스트림을 150장의 H.264로 다시 압축하는 시간을 나타내며, 가로축 QP는 H.264의 양자화 계수를 나타낸다.

표 1은 MPEG-4에서의 비트율을 H.264에서 QP를 조정하여 비트율이 약 1/2로 줄었을 때 각 영상에 대한 직렬 화소영역 트랜스코딩과 제안된 BI, WBI, CBI,

표 1. 각 영상들의 특정 QP에서의 제안된 방법 및 직렬 화소영역 트랜스코딩의 수행시간 비교

Table 1. Comparison of the proposed methods and the cascaded pixel-domain transcoding in particular QP of each.

	MPEG-4 Bitstream size (kbyte)	Quantization Parameter (QP)	Experimentation	PSNR	H.264 Bitrate (kbps)	Transcoding Execution Time (sec)
News (QCIF)	95.9	28	Cascade	33.11	55.61	56.86
			BI	32.92	60.89	14.19
			WBI	32.92	60.94	14.12
			CBI	32.91	60.97	14.69
			WBI+CBI	32.93	61	14.77
Container (QCIF)	70.6	28	Cascade	33.07	38.03	56.15
			BI	32.89	40.92	14.31
			WBI	32.89	40.92	14.3
			CBI	32.89	40.92	13.86
			WBI+CBI	32.89	40.92	14.06
Foreman (QCIF)	139	30	Cascade	31.48	72.54	55.76
			BI	31.28	85.13	17.45
			WBI	31.28	85.37	17.64
			CBI	31.27	85.3	17.68
			WBI+CBI	31.28	85.68	17.12
Silent (QCIF)	92.2	29	Cascade	32.22	46.41	55.1
			BI	31.99	53.02	15.2
			WBI	31.97	52.96	15.02
			CBI	31.99	52.84	15.34
			WBI+CBI	31.97	52.87	15.25
Paris (CIF)	51[illegible]	30	Cascade	31.06	246.51	226.26
			BI	30.88	268.62	55.72
			WBI	30.88	268.87	56.34
			CBI	30.89	268.76	55.5
			WBI+CBI	30.87	268.92	55.76
Mobile&Calendar (CIF)	3070	26	Cascade	31.69	1565.96	281.32
			BI	31.62	1622.94	81.97
			WBI	31.62	1623	80.3
			CBI	31.62	1622.86	80.8
			WBI+CBI	31.61	1622.72	81.12
Tempete (CIF)	1340	28	Cascade	31.23	761.47	210.32
			BI	31.1	820.82	63.02
			WBI	31.1	820.89	61.97
			CBI	31.1	820.6	61.39
			WBI+CBI	31.1	820.6	61.08

WBI+CBI 방법에 의한 PSNR대 비트율, 트랜스코딩 수행 시간에 대한 결과를 보여준다. 다양한 영상에 대해서도 그림 10, 11에서 보았던 비슷한 성능의 결과임을 알 수 있다.

V. 결 론

본 논문에서 움직임 벡터 보간을 이용하여 MPEG-4에서 H.264로의 트랜스코딩을 위해 MPEG-4에서 정의된 MB 모드를 H.264에서 이용하기 위한 MB 모드를 어떻게 변환할 것인지, H.264에서 프레임 율을 줄여 비트율을 줄인 경우에 MPEG-4의 움직임 벡터를 재사용하여 H.264에서 움직임 벡터를 찾는 3가지 움직임 벡터 보간 방법을 제안했다. 블록 모드 변환과 3가지 움직임 벡터 보간 방법을 적용해 본 결과 제안한 방법들간에는 수행 시간과 PSNR 모두 비슷한 성능을 보였다. 따라서 프레임 율을 이용한 비트율 조정, 블록 모드 변환과 본 논문에서 제안한 3가지 움직임 벡터 보간 방법을 이용하여 움직임 벡터를 재사용 한다면 직렬 화소영역 트랜스코딩에 비해 PSNR 손실은 있으나 계산 복잡도는 줄이면서 심각한 화질 열화가 없는 모바일 환경에 적합한 MPEG-4에서 H.264로의 트랜스코더를 구현할 수 있을 것으로 기대된다. 또한 앞으로 트랜스코딩의 복잡도를 더욱 줄이면서 단말에서 요구하는 비트율로 비트스트림을 전송하는 트랜스코딩용 비트율 제어(rate control) 방식에 관한 연구가 필요하다고 생각한다.

참 고 문 헌

[1] ITU-T. Recommendation H.261: Video codec for audiovisual services at px 64kbit/s, version 1, December 1990; version 2, Mar. 1993.

[2] Video Coding for Low Bit-Rate Communications, ITU-T Recommendation H.263+, 1998.

[3] "Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbps-Part 2 ISO/IEC 11172-2", ISO/IEC /JTC1/SC29/WG11 1993.

[4] "Generic Coding of Moving Pictures and Associa -ted Audio Information: Video", ISO/IEC 13818 -2, 2000.

[5] MPEG-4 Video Group, "MPEG-4 Video Verifica -tion Model Version 17.0", ISO/IEC JTC1/SC29 /WG11 N3515, Jul. 2000.

[6] "Joint Final Committee Draft (JFCD) of Joint Video Specification(ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC)", Joint Vide Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, Jun. 2003.

[7] Vetro, A.; Christopoulos, C.; Huifang Sun, "Video Transcoding Architectures and Techniques:An Overview", IEEE Signal Processing Magazine, vol. 20, Issue: 2, Mar. 2003.

[8] J.L. Wu, S.J. Huang, Y.M. Huang, C.T. Hsu, and J. Shiu, "An efficient JPEG to MPEG-1 transco -ding algorithm,"IEEE Trans. Consumer Electron, vol. 42, Issue: 3, pp. 447-457, Aug. 1996.

[9] P.Yin, M. Wu, and B.Lui, "Video transcoding by reducing spatial resolution," in Proc. IEEE Int. Conf. Image Processing, Vancouver, BC, Canada, vol. 1, pp.972-975, Oct. 2000.

[10] Mei-Juan Chen, Ming-Chung Chu, and Chih -Wei Pan, "Efficient Motion-Estimation Algori -thm for Reduced Frame-Rate Video Transco -der", IEEE Trans. Circuits Syst. Video Technol. vol. 12, no.4, pp.269-275, Apr. 2002.

[11] Jeongnam Youn, Ming-Ting Sun, Chia-Wen Lin, "Motion Vector Refinement for High-Perfor mance Transcoding" IEEE Trans. Multimedia, vol.1, no.1, pp.30-40, Mar. 1999.

[12] J.Youn and M.-T. Sun, "Motion Estimation for High Performance Transcoding", IEEE Int. Conf. Consumer Electronics, vol.44, Issue:3, Aug. 1998.

[13] K.T. Fung, Y.L. Chan, and W.C. Siu, "New Architecture for Dynamic Frame-Skipping Trans -coder", IEEE Trans. Image Processing. vol.11, Issue: 8, pp.886-900, Aug. 2002.

[14] MoMuSys-FDIS-V1.0-990812

[15] http://ftp.imtc-files.org/jevt-experts/reference-software/jm42.zip

저 자 소 개

이 성 선(학생회원)

2003년 세종대학교 전산과학과 학사

2003년~현재 세종대학교 인터넷 공학과 석사 과정

<주관심분야: 트랜스코딩, 영상처리, H.264>

이 영 렬(정회원)

1985년 서강대학교 전자공학과 학사

1987년 서강대학교 전자공학과 석사

1999년 한국과학기술원 전기 및 전자공학과 박사

2001년 8월 삼성전자 중앙연구소 DMS Lab.수석 연구원

2001년 9월~현재 세종대학교 인터넷공학과 부교수

<주관심분야: 영상처리(압축,복원), 영상정송, 멀티미디어 시스템, Video Transcoding>