시간해상도 감소 트랜스코딩을 위한 ASW 움직임벡터 정밀화 알고리즘에 관한 연구

서 동 완, 권 혁 민, 최 윤 식
연세대학교 전기전자공학과
전화 : 02-2123-2774 / 핸드폰 : 019-9157-7283

Efficient Motion Refinement Algorithm based on ASW for Reduced Frame-Rate Video Transcoder

Abstract

In this paper, we propose efficient motion vector refinement algorithm for frame-rate reduction transcoding. The proposed algorithm is to set the search range for motion refinement based on the incoming motion vector. The algorithm calculates the importance of motion vector of the skipped frame and then selects two motion vector to set search range. Through this process, we determine the accuracy of incoming motion vector and set the search range for refinement adaptively by means of the accuracy. In experiments, we show efficiency of our algorithm to reduce the search points for refinement.

I. 서론

인터넷을 통한 VOD(Video On Demand) 서비스의 일반화, 무선 멀티미디어 서비스의 활성화 및 디지털 방송의 시작으로 다양한 멀티미디어 매체가 발생하였고 네트워크 멀티미디어 서비스가 증가하였다. 이러한 어플리케이션에서 다양한 채널의 대역폭에 맞도록 부호화된 동영상 스트림의 비트율을 조정할 필요가 발생하였고, 어플리케이션의 종류에 따라 부호화 방식의 변화의 필요성 또한 발생하였다. 트랜스코딩을 통하여 미리 부호화된 동영상 스트림을 낮은 비트율로 변화시키는 트랜스코딩에 의해 부호화된 동영상 스트림의 비트율을 다양한 채널 상황에 맞게 조절할 수 있다. 유선망으로 전송된 입력 스트림을 무선망으로 전송하기 위해 낮은 대역폭의 출력 스트림으로 트랜스코딩하는 것은 높은 비율의 비트율 조정이 필요하다. 하지만 높은 비율의 비트율의 변화는 입력 스트림의 전체 프레임 율로 트랜스코딩할 경우 화질에 큰 손상을 주게 된다. 발생하는 화질 저하를 보상하기 위해 일반적으로 프레임 율의 감소나 프레임 스킵(skip)을 사용하여 프레임당 할당되는 비트를 증가시켜 해결한다.

트랜스코더의 실시간 응용을 위해 부호기의 움직임 예측을 최소화하기 위해 입력 스트림에서 복호화된 움직임벡터를 출력 움직임벡터로 재사용한다.[1,2,3,4] 프레임 율이 변화되면 입력 움직임벡터가 스킵된 프레임을 참조할 수 있으므로 출력 움직임벡터로 바로 사용하지 못한다. 출력움직임 벡터를 입력 움직임벡터에서 추정하기 위한 알고리즘과 이에 기초한 움직임 벡터 정밀화 알고리즘에 관한 연구들이 진행되었다.[1,2,3,4] 기존의 알고리즘은 추정된 움직임벡터에 기초하여 검색 영역을 결정하고 3SS(Three Step Search)과 같은 검색 포인트를 줄이는 알고리즘들에 의해 행한다. 이는 움직임 정밀화를 넓은 검색 영역에서 행하므로 움직임 정밀화의 효율을 떨어뜨리게 된다.

본 논문에서는 추정된 움직임 벡터 및 그의 변동성을 입력 움직임 벡터에 의해 추정하여 검색영역을 정

본 연구는 대학 IT 연구센터 육성 지원사업의 연구결과로 수행되었음

함으로써 검색 포인트를 증가시키지 않고 트랜스코더의 효율성을 증가시키는 알고리즘을 제안한다. 즉 검색 영역을 입력 움직임벡터에 의해 적응적으로 선택하는 ASW(Adaptive Search Window)에 기반하는 움직임 정밀화 알고리즘을 제안한다. Ⅱ에서는 프레임 율 감소 트랜스코딩에서 입력 움직임 벡터에 의해 출력움직임 벡터를 추정하는데 있어서의 문제점을 분석하고 이를 해결하는 기존의 알고리즘을 설명한다. Ⅲ에서는 Ⅱ에서 결정되는 움직임벡터와 그 변동성에 의한 적응적 검색 영역을 잡는 방법을 제시하겠다. Ⅳ와 Ⅴ에서는 실험결과와 결론을 내리도록 하겠다.

Ⅱ. 프레임 율 감소 시 출력 움직임벡터의 추정

일반적으로 트랜스코딩에서는 복잡도를 감소시키기 위해 입력 움직임벡터로 부터 출력 움직임벡터를 추정하여 사용한다. 프레임 율이 감소되는 경우 입력 움직임벡터가 참조하는 프레임이 출력 스트림 내에 존재하지 않아 더 이상 적절한 출력 움직임벡터를 추정하기 어렵다. 예를 들어, 그림 1과 같이 하나의 프레임이 스킵된 경우에 트랜스코딩에 있어서 출력 움직임벡터를 결정할 때의 문제를 설명하겠다.

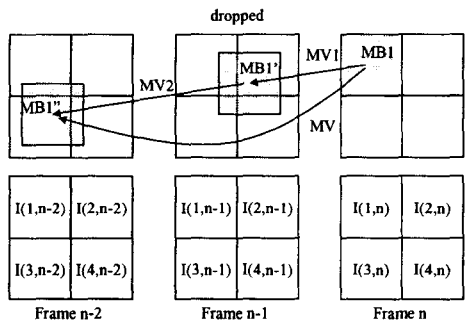


그림 1. 프레임 율 감소 시 최적 움직임벡터

그림 1에서와 같이 Frame n-1이 스킵된 경우의 Frame n의 매크로블록 MB1의 움직임벡터는 Frame n-2에서 다시 움직임 추정을 실행하여야 최적의 움직임벡터를 찾을 수 있다. 이는 복잡도를 증가시키므로 입력 움직임벡터를 통해 준-최적화된 출력 움직임벡터를 구하여 활용한다. 준-최적화된 출력 움직임벡터는 식 1)과 같이 MB1과 가장 잘 맞는 MB1'의 움직임벡터와 입력 움직임벡터의 합을 통해 구할 수 있다.

$$MV = MV1 + MV2 \qquad 1)$$

그러나 MB'이 매크로블록 경계와 일치하지 않아서 MV2는 입력 스트림에서 알 수 있는 정보가 아니다. 입력 스트림에서 알 수 있는 Frame n-1의 입력 움직임 벡터 $I(1,n-1)$, $I(2,n-2)$, $I(3,n-3)$, $I(2,n-1)$에서 MV2를 추정해야 한다.

MV2를 추정하는 알고리즘으로 FDVS(Forward Dominant Vector Selection)[2,3]과 ADVS(Activity Dominant Vector Selection)[4] 알고리즘이 제안되었다. FDVS 알고리즘은 그림 1에서 MB1'과 겹치는 4개의 주변 블록들 중에서 MB1'과 가장 많이 겹치는 매크로블록의 움직임벡터를 MV2로 선택한다. 그림 1에서는 MV2는 $I(2,n-1)$의 움직임벡터로 선택되어 최종적인 출력 움직임벡터는 식 2)와 같다.

$$MV = MV1 + I(2,n-1) \qquad 2)$$

ADVS 알고리즘은 객체의 경계에서 더 많은 양의 예측 에러를 생성한다는 현상을 고려하여 MB1'이 주변 4개의 매크로블록과 겹치는 부분의 Activity가 최대가 되는 매크로블록의 움직임벡터를 선택하여 출력 움직임벡터를 계산한다. 예를 들어 MB1'과 겹치는 블록의 Activity가 최소가 되는 매크로블록이 오른쪽 아래의 매크로블록이라면 MV2는 $I(4,n-1)$이 되고 출력 움직임벡터는 식 3)과 같이 구해진다.

$$MV = MV1 + I(4,n-1) \qquad 3)$$

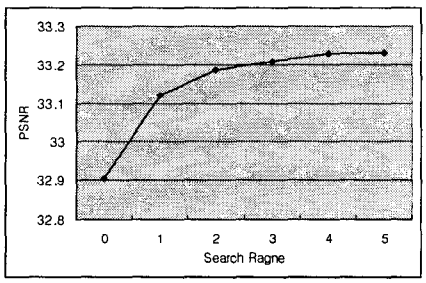
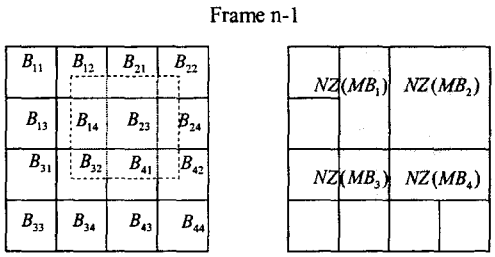


그림 2. 검색영역에 증가에 따른 PSNR : Carphone (352×288, 150장)의 MPEG-2 스트림을 ADVS 알고리즘에 의해 MPEG-4 스트림으로 트랜스코딩

두 알고리즘 모두 프레임이 스킵된 경우 입력 움직임벡터에서 각기 측정 기준은 다르나 움직임의 경향성을 판단하여 최종적인 출력 움직임벡터를 결정한다. 앞서 설명했듯이 이렇게 계산된 출력 움직임벡터는 준-최적 움직임벡터이므로 트랜스코더의 효율을 증가시키기 위해 출력 움직임 벡터 주위를 움직임 정밀화 과정을 수행한다. 기존의 고속 정밀화 알고리즘은 결정된 출력 움직임벡터에 기반하여 검색영역을 계산하고 검색영역 내에서 3SS과 같이 검색 포인트들을 듬성하게 하여 정밀화한다. 또한 그림 2와 같이 일반적으로

[-2,2]의 검색 영역내에서 정밀화하면 거의 움직임 추정을 새로이 하는 것과 근사화된다고 알려져 있다. 본 논문에서는 검색 영역을 입력 움직임벡터에 의해 적응적으로 설정하여 움직임 정밀화를 효율화하는 알고리즘을 제안한다.

Ⅲ. ASW에 기반한 움직임 정밀화 알고리즘



$$NZ(MB_1) = NZ(B_{12}) + NZ(B_{14})$$
$$NZ(MB_2) = NZ(B_{21}) + NZ(B_{22}) + NZ(B_{23}) + NZ(B_{24})$$
$$NZ(MB_3) = NZ(B_{32})$$
$$NZ(MB_4) = NZ(B_{41}) + NZ(B_{42})$$

그림 3. 출력 움직임벡터 및 검색영역 결정 움직임벡터 결정

트랜스코딩에서 출력 움직임벡터는 앞서 설명한 ADVS 알고리즘으로 결정하고, 검색영역을 결정하기 위해서 ADVS 알고리즘의 개념을 이용하여 검색영역 결정 움직임벡터를 계산한다. 그림 1에서의 예를 그대로 적용하여 제안 알고리즘에서 출력 움직임벡터를 결정하는 방법과 검색영역을 규정하기 위한 움직임벡터를 계산하는 과정을 설명하겠다. ADVS 알고리즘에서 사용하는 Activity은 입력 스트림에서 "0"이 아닌 DCT 계수의 개수로 측정한다. MB1과 겹쳐진 매크로블록의 움직임벡터의 영향력을 고려한 것이다. 즉 Activity가 적으면 움직임벡터가 약간 잘못되어도 차이(residual) DCT 계수의 변화가 적을 것으로 예상하고 Activity가 많으면 움직임벡터에 의해 차이 DCT 계수의 변화가 클 것으로 예상하여 비트량을 높이지 않는 움직임벡터를 선택하는 기준으로 삼는 것이다. 그림 1의 Frame n-1을 그림 3과 같이 블록 경계로 나눈 후 Activity를 겹쳐지는 블록에 대해서만 계산을 한다. 각각의 매크로블록의 움직임벡터의 영향력을 Activity $NZ(MB_i)$로 나타내어 이가 최대가 되는 매크로블록의 움직임벡터를 식 3)과 같이 출력 움직임벡터의 계산에 사용한다. 그리고 두 번째로 Activity가 큰 블록의 움직임벡터를 검색영역 결정 움직임벡터를 계산하는데 사용한다. 출력 움직임벡터를 계산하는데 사용하는 움직임벡터는 식 4)와 같이 하고, 검색영역을 결정하는데 사용하는 움직임벡터는 식 5)와 같이 계산한다.

$$MV2 = \arg max_{MV_i} NZ(MB_i) \quad 4)$$

$$MVS = \arg max_{MV_i \neq MV2} NZ(MB_i) \quad 5)$$

MVS는 검색영역을 결정하는데 사용하는 움직임벡터이고, MV_i는 MB_i의 움직임벡터이다.

출력 움직임벡터는 움직임벡터의 영향력을 고려함과 동시에 매크로블록 경계가 겹치는 4개의 움직임벡터의 경향성을 대표한다. 만약 두 번째 영향력이 있는 움직임벡터 즉, 검색영역을 결정하는 움직임벡터 MVS가 MV2와 같다면 이는 거의 정확한 움직임벡터로 간주할 수 있다. 따라서 MVS와 MV2가 같다면 검색영역을 크게 늘려주지 않아도 된다. 본 논문에서는 검색영역을 [-1,1]로 설정해 주고, 그렇지 않은 경우는 그림 4에서와 같이 검색영역을 MV2와 MVS에 의해 계산된 MV1을 검색영역의 바운더리로 설정하여 움직임 정밀화 과정을 진행한다. MB1'과 겹치는 4개의 움직임벡터의 경향성을 반영한 검색 영역이므로 기존의 검색영역을 설정하는 방법보다는 효율적인 검색영역이 된다. 그림 4에서 V_1과 V_2는 다음 식 6)과 7)로 표현된다.

$$V_1 = MV1 + MV2 \quad 6)$$

$$V_2 = MV1 + MVS \quad 7)$$

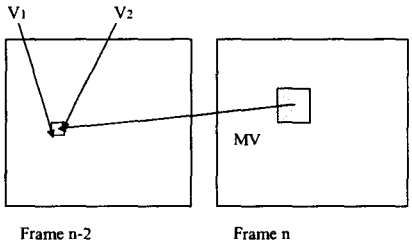


그림 4. ASW 알고리즘

최종적인 출력 움직임벡터는 V_1과 V_2에 의해 설정되는 검색영역 S_R에 의해 식 8)과 9)와 같은 움직임 정밀화 과정을 통하여 결정한다.

$$(O_x, O_y) = (MV_x + D_x, MV_y + D_y) \quad 8)$$

$$(D_x, D_y) = \arg min_{(m,n) \in S_R} SAD_R(m,n) \quad 9)$$

SAD_R은 Block-matching 알고리즘의 SAD(Sum of Absolute Difference)를 S_R에서 계산하는 값을 의미한다.

Ⅳ. 실험 결과

본 논문의 실험은 CIF 크기의 Foreman 과 News 시퀀스를 사용하여 30fps의 MPEG-2 시퀀스를 15fps의 MPEG-4 시퀀스로 트랜스코딩하였다. 또한 ASW 움직임 정밀화 기법을 FDVS와 ADVS로 기본 움직임벡터를 형성한 후 [-2,2]의 검색영역 내에서 정밀화한 기법과 비교하였다. 그림 5와 6에서 알 수 있듯이 FDVS와 ADVS 기법에 [-2,2] 검색영역의 정밀화를 취한 방법과 ASW 알고리즘을 적용한 방법 모두 Full Search를 한 방법과 거의 같은 성능을 보인다. 그러나 본 논문에서 제안한 ASW 알고리즘은 FDVS와 ADVS에 [-2,2]의 검색영역으로 정밀화를 취한 방법에 비해 표 1에서 보듯이 매크로블록 당 검색 포인트 수를 거의 1/3 수준으로 떨어뜨려 트랜스코더 내의 복잡도를 감소시키는 동시에 효율을 그대로 유지한다. 이는 움직임벡터의 경향성과 프레임 스킵이 일어나면서 움직임의 상관도에 따른 검색영역의 설정으로 매크로블록 마다 다른 검색 영역을 선택하기 때문이다.

Ⅴ. 결론

본 논문에서는 프레임 율 감소 트랜스코딩에 있어서 입력 움직임벡터에 의해 검색영역을 결정하는 ASW 알고리즘으로 효율적인 움직임 정밀화 기법을 제시하였다. 스킵된 프레임에서 매크로블록 경계가 일치하지 않는 경우 두 개의 움직임벡터를 추정하여 하나는 출력 움직임벡터의 기본 벡터를 계산하는데 사용하고 다른 하나는 검색영역을 결정하는데 사용하였다. 제안 알고리즘은 움직임의 변동성이 크면 검색영역을 크게 하고 그렇지 않으면 검색영역을 작게 하여 필요 없는 검색 포인트를 줄여 움직임 정밀화의 효율을 높였다.

본 논문은 실시간 트랜스코딩 알고리즘에 있어 계산량을 크게 늘리지 않고 움직임 추정을 효율적으로 실행하여 화질 열화를 최소화하는 알고리즘으로 사용될 수 있다.

참고문헌(또는 Reference)

[1] J. Youn, M. T. Sun; C. W Lin, "Motion estimation for high performance transcoding," Consumer Electronics, IEEE Trans. on , Vol 44, PP 649 -658, Aug 1998.

[2] J. Youn, M. T. Sun ,"A fast motion vector composition method for temporal transcoding", Circuits and Systems, ISCAS '99. Vol 4, pp. 243 -246, Jul 1999.

[3] J. Youn, M. T. Sun; C. W Lin, "Motion vector refinement for high-performance transcoding," Multimedia, IEEE Trans. on, Vol. 1, pp. 30-40, Mar 1999.

[4] M. J. Chen, M. C. Chu, C. W. Pan, "Efficient motion-estimation algorithm for reduced frame rate video transcoder ," CSVT, IEEE Trans. on, Vol. 12, pp 269-275, Apr 2002.

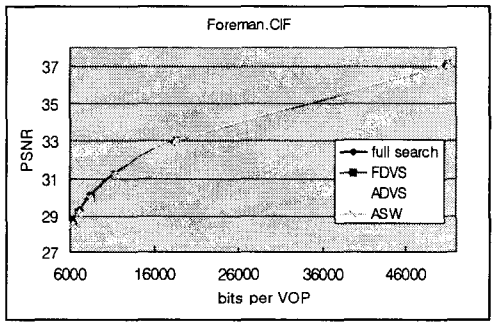


그림 5. Transcoding 후 PSNR과 bits (Foreman)

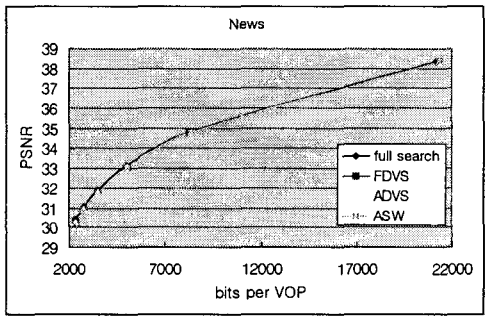


그림 6. Transcoding 후 PSNR과 bits (News)

<표 1> 매크로블록 당 검색포인트 수

Algorithm	FDVS	ADVS	ASW
Foreman	16.49	16.34	5.85
News	14.97	14.96	5.87