수정된 적응적 움직임 벡터 해상도 부호화 방법

*장명훈 **한종기 ***배진수

세종대학교

*goolapa83@gmail.com, **hjk@sejong.ac.kr, ***baej@sejong.ac.kr

Modified Adaptive Motion Vector Resolution

*Jang, Myoung-Hun **Han, Jong-Ki ***Bae, Jinsoo

Sejong University

요약

기존의 참조 소프트웨어인 MPEG-2, MPEG-4, H.264/AVC에서는 움직임 벡터를 찾을 때 항상 고정된 해상도를 사용하였으며 다른 참조 소프트웨어인 KTA에서는 움직임 벡터를 찾을 때 움직임벡터의 해상도를 슬라이스 단위로 성능이 가장 높은 해상도를 선택해서 사용하였다. 하지만 움직임 벡터의 해상도는 블록마다 서로 다르기 때문에 블록별로 서로 다른 해상도를 적응적으로 사용할 필요가 있다. 적응적인 움직임 벡터 해상도 부호화 방법은 이러한 점을 이용하여 블록 별로 현재 블록의 움직임 벡터가 1/4 해상도인지 1/8 해상도인지에 판단하고 그에 대한 정보를 복호기에 전송해준다.

제안하는 알고리즘은 적응적 움직임 벡터 해상도를 사용하여 부호화 할 때 1/8 해상도 움직임 벡터가 성능이 없다고 판단되는 곳에선 적응적 움직임 벡터 해상도 방식을 사용하지 않고 1/4 해상도로만 움직임 벡터를 찾는다. 이러한 경우 해상도 정보를 복호기에 전송하지 않아 부호화 효율을 높일 수 있고 또한 1/8 해상도에 대한 움직임 예측을 하지 않기 때문에 부호화기 복잡도를 낮출 수 있다. 실험결과 평균 0.2%의 성능을 얻을 수 있었으며 부호화기 복잡도는 4% 감소하였다.

1. 서론

움직임 벡터를 찾기 위해서 초기의 참조 소프트웨어(H.261[1])에서는 움직임 벡터를 정수 화소 단위에서만 찾았다. 하지만 어떤 경우에는 참조 프레임의 보간된 샘플 위치에서 움직임 추정을 수행함으로써 더 나은 움직임 보상의 결과를 얻을 수 있다. 따라서 기존의 MPEG-2[2], MPEG-4[3]에서는 반 화소 단위로 움직임 벡터를 찾으며 H.264/AVC[4]에서는 1/4 화소 단위로 움직임 벡터로, 그리고 KTA[5]에서는 1/8화소 단위로 움직임 벡터를 고려 할 수 있게 했다.

일반적으로 높은 해상도의 움직임 벡터를 사용하면 낮은 해상도의 움직임 벡터를 사용한 것과 비교해서 좀 더 정교한 움직임 벡터를 찾을 수 있어 예측 에러값을 효과적으로 줄일 수 있지만 움직임 벡터를 표현하는 인덱스가 늘어나게 되어 많은 비트가 발생하게 된다. 따라서 코딩 효율을 높일 수 잇게 적응적으로 움직임 벡터의 해상도를 결정해 줄 필요가 있다.

현재 새롭게 표준을 논의하는 HEVC에서는 이러한 문제점을 해결하기 위해 적응적 움직임 벡터 해상도 부호화 방법(Adaptive Motion Vector Resolution-AMVR)[6]을 제안하였다. AMVR은 현재의 움직임 벡터의 해상도를 1/4 화소 단위로 할 것인지 1/8 화소 단위로 할 것인지를 결정해서 결정된 정보를 움직임 벡터가 발생할 때 마다 복호기에 전송해주는 기술이다.

본 연구는 지식경제부 및 정보통신산업진흥원의 IT융합 고급인력과정 지원사업의 연구결과로 수행되었음 (NIPA-2011-C6150-1101-0003)

하지만 AMVR은 매번 1/8화소 단위가 고려되어지기 때문에 복잡도가 증가 하고 움직임 벡터가 발생할 때마다 플래그 비트가 추가적으로 발생하여 성능을 저하시킨다.

일반적으로 차분 움직임 벡터의 크기가 크게 발생하면 높은 해상도의 움직임 벡터를 사용하는 것이 비효율적이다.[7] 본 논문에서는 이를 이용하여 실제 1/8화소 단위로 움직임 벡터를 찾는 것이 효율적인 구간에서만 AMVR 방식을 사용하고 1/8화소 단위로 움직임 벡터를 찾는 것이 비효율적인 부분에서는 1/4화소 단위로만 움직임 벡터를 찾아 부호화 효율을 높이고 부호화기 복잡도를 낮추는 알고리즘이다.

2. 본론

앞에서도 이야기 했듯이 AMVR은 현재 움직임 벡터를 부호화 할 때 현재 움직임 벡터의 해상도를 알려주는 플래그가 있다. 따라서 일반적으로 움직임 벡터를 찾기 위해서 다음과 같은 수식 (1)이 최소가 되는 움직임 벡터를 찾는다.

$$M\_Cost = D + \lambda R(MVD + AMVR_{flag} + etc) \quad (1)$$

이때, MVD(차분움직임 벡터)는 수식 (2)와 같이 현재 움직임 벡터에서 예측움직임 벡터와의 차이를 이용하여 구하며 D는 수식 (3)과 같이 현재 블록의 픽셀 값과 예측 블록의 픽셀 값의 차이를 이용하여 구할 수 있다.

$$MVD = MV - PMV \quad (2)$$

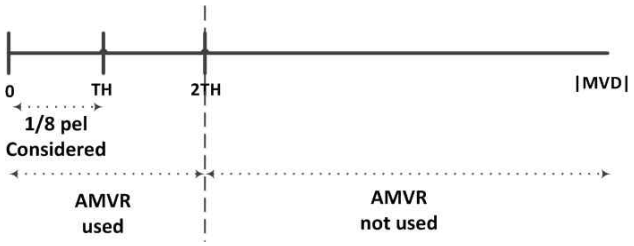
$$D=\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}|or i_{ij}-ref_{ij}| \quad (3)$$

AMVR은 만약 현재 움직임벡터의 해상도 1/8단위라면 식 (2)에서 계산된 MVD를 그대로 사용하고 만약 현재 움직임 벡터의 해상도가 1/4 단위라면 MVD값은 식(4)와 같이 변한다.

$$MVD=(MV-PMV)/2 \quad (4)$$

움직임 벡터의 해상도가 높아지면 고려되어지는 보간 샘플들이 많아져 에러값을 효과적으로 낮출 순 있지만 반대로 차분 움직임 벡터를 표현하는 인덱스가 늘어나면서 비트가 증가되는 경향이 있다. AMVR은 이러한 점을 고려하여 에러가 효과적으로 감소하는 부분에서는 현재 해상도를 1/8로 결정하며 에러가 효과적으로 감소하지 않는 부분에서는 현재 해상도를 1/4로 결정한다. 하지만 AMVR은 모든 움직임 벡터가 발생 할 때 마다 추가적으로 1비트가 필요로 하게 되므로 1/8 해상도가 필요 없는 영상의 경우에는 부호화 효율이 떨어지게 된다.

제안하는 알고리즘은 차분 움직임 벡터의 크기가 커지면 1/8 해상도에 대한 부호화 효율이 없는 점을 이용하여 차분 움직임 벡터의 크기에 따라 AMVR을 사용하는 부분과 사용하지 않는 부분을 나눈다. 1/8 해상도의 성능이 있는 부분에서만 AMVR을 사용하고 1/8 해상도의 성능이 없는 부분에서 사용하지 않는다면 사용하지 않는 부분에서는 flag 비트를 제거되어 부호화 효율을 높일 수 있다.



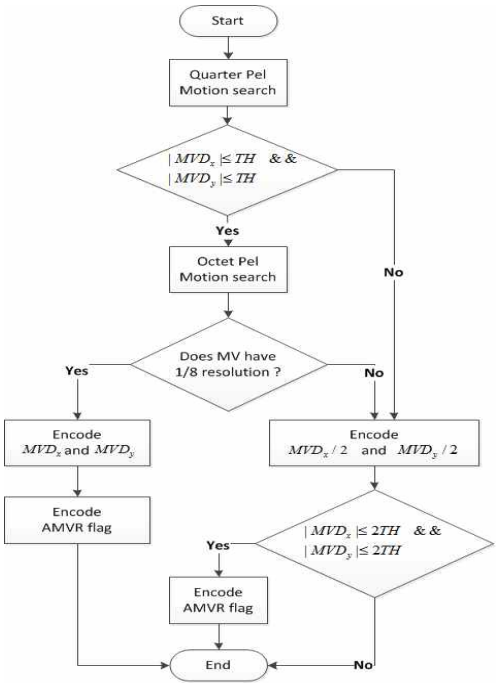
<그림 1> 수정된 적응적인 움직임 벡터 해상도 결정 방법

제안하는 알고리즘은 <그림 1>과 같다. <그림 1>에서 보면 1/8 해상도는 차분 움직임 벡터가 TH 이내에 있는 경우에만 고려되며 그 이후에는 고려되지 않는다. 따라서 부호화기에서는 수식 (5)가 만족하면 1/8해상도를 고려하여 움직임 벡터를 찾으며 만족하지 않을 경우에는 1/4 해상도 단위로만 움직임 벡터를 찾아 인코더 부호화 복잡도를 낮춘다.

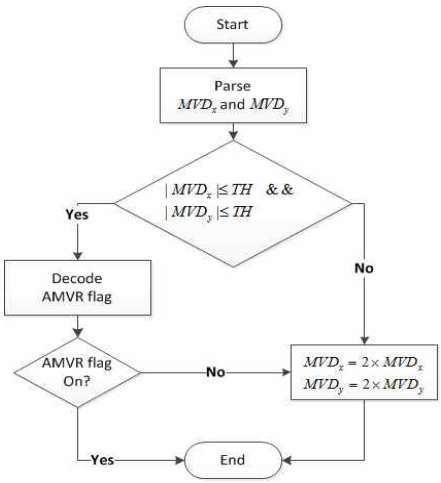
$$MVD_x \le TH \quad \&\& \quad MVD_y \le TH \quad (5)$$

AMVR은 움직임 벡터의 해상도를 1/8 해상도 단위로 만들어 놓고 움직임 벡터의 해상도가 1/4 해상도면 실제 차분 움직임 벡터의 크기를 절반으로 하여 부호화 한다. 따라서 TH이후의 차분 움직임 벡터의 크기는 반이 된다. 이 때 크기가 반이된 차분 움직임 벡터의 값이 TH 아래로 내려가면 복호기에서 복호된 차분움직임 벡터가 어떤 값인지 판별이 되지 않기 때문에 AMVR의 사용범위를 2TH로 지정해 주어야 한다.

복호기에서는 복호된 차분 움직임 벡터의 크기가 TH보다 작으면 AMVR에 대한 플래그 비트를 읽고 만약 복호된 차분 움직임 벡터가 TH보다 크면 AMVR에 대한 플래그 비트를 읽지 않는다.



<그림 2> 부호화기 순서도



<그림 3> 복호기 순서도

3. 실험 조건 및 실험 결과

제안하는 알고리즘은 성능을 검증하기 위해 참조 소프트웨어인 H.M 0.9[8]을 사용하여 제안한 알고리즘을 부호화기와 복호화기에 가각 구현하였다. 실험에서 1/8 해상도 고려되는 구간 TH를 4로 설정하며 다른 실험 조건은 [9]의 실험 컨디션을 따른다.

	Random access			Low delay		
	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate
Class A	-0.2	-0.4	-0.4			
Class B	-0.2	-0.3	-0.5	-0.1	-0.1	-0.2
Class C	-0.1	-0.2	-0.2	-0.1	0.2	-0.2
Class D	-0.2	-0.3	-0.2	-0.2	-0.4	-0.5
Class E				-0.2	-0.5	0.1
All	-0.2	-0.3	-0.3	-0.1	-0.2	-0.2
Enc Time[%]	96%			96%		
Dec Time[%]	100%			101%		

<표 1> 실험 결과

실험결과 Random access에서는 0.2%, Low delay에서는 0.1%의 성능을 봤으며 부호화기 복잡도는 Random access, Low delay 각각 4%씩 감소하였다.

4. 결론

본 논문에서는 적응적인 움직임 벡터 해상도를 기법을 이용하여 1/8 해상도의 움직임 벡터를 고려 할 때 움직임 벡터의 크기 에 따라 적응적인 움직임 벡터 해상도 기법을 적용하는 방식에 대해서 설명하였다. 컴퓨터 실험 결과 Random access 상황에서는 0.2% Low delay에서는 0.1%의 성능을 얻을 수 있었으며, 부호화기 복잡도는 Random access와 Low delay 모두 4%정도 감소하였다.

5. 참고 문헌

[1] Comite Consultaif International Telegraphie et Telephonie (CCITT), H.261 video for audiovisual services at p*64 kbits, Geneva, 1990

[2] Video - Generic coding of moving pictures and associated audio, International Standard ISO/IEC 13818-2, Nov. 1994

[3] ISO/IEC 14496-2 “Information technology Coding of audio-visual objects - Part2: Visual,” June 2004.

[4] Draft ITU-T Recommendation and Final Draft International Standard of Joint Specification (ITU-T Rec. H.264/ISO/IEC 14496-10 AVC), Mar. 2003

[5] ITU-T VCEG, Reference ITU-T VCEG-KTA Software is available at http://iphome. hhi.de/suehring/tml/download/KTA/

[6] JCT-VC “Video coding technology proposal by Qualcomm” 1st JCT-VC Meeting, Dresden, Germany, Aprill 2010

[7] M.H.Jang, C.W. Seo and J.K. Han, “Motion Estimation using Region-based Adaptive Motion Vector Precision,” The 26th International Technical Conference on Circuits/Systems, Computers and Communications, Jun. 20-22 2011.

[8] T.Weigand, W.-J. Han, J.-R. Ohm, and G. J. Sullivan, “High Efficiency Video Coding (HEVC) text specification Working Draft 1,” JCTVC-C403, Guangzhou, China, October 2010.

[9] F. Bossen “Common test conditions and software reference configurations”, JCTVC-C500. Guangzhou, China, Oct. 2010