레터논문-12-17-2-19 http://dx.doi.org/10.5909/JEB.2012.17.2.422

HEVC를 위한 CU기반 병합 후보 리스트 구성 방법

김 경 용[a], 김 상 민[a], 박 광 훈[a]‡, 김 휘 용[b], 임 성 창[b], 이 진 호[b]

CU-based Merge Candidate List Construction Method for HEVC

Kyung Yong Kim[a], Sang Min Kim[a], Gwang Hoon Park[a]‡, Hui Yong Kim[b], Sung Chang Lim[b], and Jin Ho Lee[b]

요 약

본 논문에서는 기존 PU기반 움직임 병합 후보 리스트 생성 방법과 비교하여 계산 복잡도를 감소시키고 개선된 병렬성을 제공하기 위하여 CU 기반의 움직임 병합 후보 리스트 생성 방법을 제안한다. 제안하는 방법에서 하나의 CU는 오직 하나의 움직임 병합 후보 리스트를 생성한다. 그래서 CU내의 모든 PU 블록은 PU형태에 관계없이 오직 하나의 공통 움직임 병합 후보 리스트를 사용한다. 제안방법의 실험 결과, 기존 방법보다 인코더의 복잡도는 3% ~ 6% 감소하였으며, 디코더의 복잡도는 거의 변화가 없었다. 반면 제안방법이 기존 방법보다 부호화 효율이 0.2% - 0.5% 감소된다는 단점이 존재한다. 하지만 제안방법은 하드웨어 구현이 간단하며, 계산 복잡도를 감소시키고 향상된 병렬성을 제공한다는 장점을 갖는다.

Abstract

This paper proposes the CU-based approach for merge candidate list construction for providing reduced complexity and improved parallelism compared to the PU-based one. In the proposed method, a CU can have only one merge candidate list. So, Only one common merge candidate list is used for all PUs in a CU regardless of the PU partition type. The simulation results of proposed method showed that the encoder computational complexity was decreased by 3% to 6% and the decoder computational complexity was negligible change with the penalty of roughly 0.2% - 0.5% coding loss. The proposed method has several advantages: it provides simpler design, reduced complexity, and improved parallelism.

Keyword : HEVC(High Efficiency Video Coding), Merge, CU(Coding Unit)

Ⅰ. 서 론

HD급 이상의 고해상도, 고화질 영상에 적합한 효율적인 압축 표준을 제정하기 위해서 MPEG과 VCEG에서는 JCT-VC(Joint Collaborative Team on Video Coding)을 구성하였고, 해당 그룹에서 차세대 비디오 코딩 기술인 HEVC (High Efficiency Video Coding)[1]라는 표준을 진행하고 있다.

HEVC에서는 다양한 크기의 정사각형으로 정의된 CU

a) 경희대학교 전자정보대학 컴퓨터공학과
Dept. of Computer Engineering, College of Electronics and Information, Kyung Hee University
b) 한국전자통신연구원
ETRI
‡ 교신저자 : 박광훈(ghpark@khu.ac.kr)
※ 본 연구는 방송통신위원회의 ETRI 연구지원 사업[KCA-2011-(11921-02001), 무안경 다시점 3D지원 UHDTV 방송 기술개발]과 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 지원사업(NIPA-2011-(C1090-1111-0001))의 연구결과로 수행되었음.
· 접수일(2011년12월21일),수정일(2012년2월6일),게재확정일(2012년2월6일)

(Coding Unit)[1]를 기본 부호화 단위로 사용한다. 또한 효율적인 화면 간 예측을 위하여 HEVC에서는 AMVP (Advanced Motion Vector Prediction)와 움직임 병합(Merge) 알고리즘을 이용하여 움직임 정보를 부호화하고 있다[1]. AMVP[2]는 Motion Vector Competition[3] 알고리즘을 개선시킨 것이다. 이는 현재 블록의 주변 블록으로부터 움직임 벡터 후보 리스트를 만든 후, 그 중에서 현재 블록의 움직임 벡터와 가장 잘 어울리는 움직임 벡터를 선택하는 알고리즘이다. 움직임 병합[4] 방법도 역시 AMVP와 비슷한 개념이지만, 현재 블록의 움직임 정보를 주변 블록과 모두 동일하게 만든다는 점에서 다르다. 이러한 움직임 병합 방법은 초기에는 CU(Coding Unit) 기반으로 적용하였으나[4] 현재에는 부호화 효율이 더 우수한 PU(Prediction Unit) 기반의 움직임 병합[1]이 적용되었다. 이러한 PU기반의 움직임 병합 방법은 부호화 효율은 좋으나 초기 CU기반 움직임 병합 방법에 비해 매우 복잡하다. 또한 PU 간의 의존성으로 인해 PU간 부호화 및 복호화가 병렬적으로 수행되기 힘들다는 단점도 존재한다. 본 논문에서는 이러한 PU기반 움직임 병합 방법의 단점인 높은 복잡도와 병렬성 문제를 해결하기 위한 방법을 제안한다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 기존의 PU 기반 움직임 병합 방법에 대해서 설명하고 그것의 문제점에 대하여 분석한다. Ⅲ장에서는 Ⅱ장에서 분석한 내용을 바탕으로 CU 기반 움직임 병합 후보 리스트 생성 방법을 제안한다. Ⅳ장에서는 제안하는 방법에 대한 성능을 평가하고 장단점을 기술한다. Ⅴ장에서는 결론을 기술한다.

Ⅱ. PU 기반 움직임 병합 방법 및 문제점

HEVC에는 움직임 정보의 효율적인 부호화를 위하여 AMVP와 움직임 병합 방법이 채택되었다. 움직임 병합 방법은 현재 블록의 움직임 정보를 현재 블록에 인접한 주변 블록의 움직임 정보와 동일하게 만드는 방법이다. 즉, 현재 블록의 움직임 정보에 대한 파라미터들(단방향 혹은 양방향 예측에 대한 정보, 참조 픽쳐에 대한 정보, 움직임 벡터 정보)을 모두 주변 블록의 움직임 정보와 동일하게 만든다. 이는 동질성을 갖는 움직임 영역에서 움직임 정보가 변화 없이 공간적으로 전파됨으로써 움직임 정보의 부호화 효율을 증가시킨다.

움직임 병합 방법은 초기에 CU 단위의 정사각형 블록에만 적용되었다. 또한 공간적인 움직임 병합에만 적용되었다. 하지만 움직임 정보의 부호화 효율을 증가시키기 위하여 PU 단위의 움직임 병합 방법이 채택되었다[5]. 또한 움직임 병합 방법에 이용되는 주변 블록들의 개수도 증가하였다[6]. 그러한 주변 블록에는 공간적으로 인접한 블록뿐만 아니라 참조 픽쳐에서 현재 블록과 동일한 위치(Collocated)에 있는 시간적 블록도 사용되었다. 또한 최적의 움직임 후보 인덱스를 파싱하는 문제 및 병렬성 제공을 위해서 움직임 병합을 위한 고정적 움직임 후보 리스트가 채택되었다[7]. 또한 고정적 움직임 후보 리스트의 부호화 효율 감소 문제를 해결하기 위하여 움직임 후보 리스트의 빈 공간에 새로운 움직임 후보를 생성하여 추가하는 방법이 채택되었다[7]. 새로운 움직임 후보는 움직임 후보 리스트 안의 원본 움직임 후보들을 조합하여 만들어진다.

위와 같은 방법들로 인해 PU 기반의 움직임 병합 방법은 부호화 효율을 증가시켰지만, 계산 복잡도도 증가시켰다. 또한 PU 단위로 움직임 병합이 수행됨으로써, CU내의 PU 들 간의 의존성이 생긴다. 예를 들어, 그림 1은 하나의 CU가 두 개의 PU로 나누어져 부호화되는 경우, 각 PU를 움직임 병합하기 위해 이용되는 공간적 주변 블록들을 나타낸다.

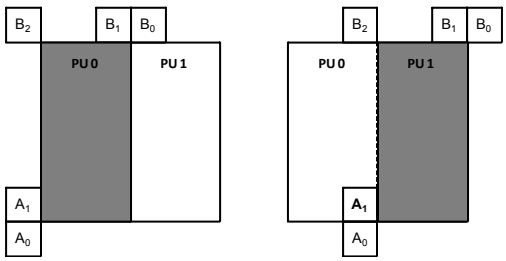

그림 1. 각 PU를 움직임 병합하기 위해 이용되는 공간적 주변 블록들
Fig. 1. Spatial neighboring blocks used in each PU for merging

그림 1에서 각 PU는 첫 번째 PU블록(PU 0)부터 두 번째 PU블록(PU 1)까지 순차적으로 부호화된다. 여기서 두 번째 PU블록(PU 1)을 위한 움직임 병합 후보 리스트는 첫 번째 PU블록의 움직임 정보를 사용하여 생성한다. 따라서

두 번째 PU블록의 부호화를 위해서는 첫 번째 PU블록의 움직임 정보가 반드시 필요하기 때문에 각 PU블록들은 순차적으로 부호화될 수밖에 없다. 즉, 각 PU는 병렬적으로 부호화될 수 없다.

또한 하나의 LCU를 부호화하기 위해서 생성되는 움직임 병합 후보 리스트의 수는 상당히 많다. 그림 2는 하나의 CU가 여러 개의 PU들로 분할되는 형태를 나타낸다. 여기서 각각의 분할 형태마다 PU의 위치와 크기가 다르다. 따라서 PU의 위치와 크기에 따라 각각 다른 움직임 병합 후보 리스트가 생성된다. 표 1은 움직임 병합 후보 리스트의 생성 횟수를 CU의 크기 별로 나타낸 것이다. 표 1에서는 HEVC 공통 실험 조건[8]에 따라 LCU의 크기를 64x64로, SCU의 크기를 8x8로 설정하였고 CU가 'NxN'의 형태로 나누어지는 경우는 제외하였다. 표 1에서 CU의 크기가 64x64 인 경우, '2Nx2N'인 형태는 1개의 PU로 부호화되고, 나머지 6가지 형태는 2개의 PU로 부호화되므로 움직임 병합 후보 리스트의 생성횟수는 13번이다. 또한 CU의 크기가 32x32 인 경우, 크기가 64x64 CU 내에 서로 위치가 다른 4개의 32x32 CU가 존재하므로 움직임 병합 후보 리스트의 생성횟수는 52번이다. CU의 크기가 16x16 인 경우에도 역시 위치가 다르므로 208번이 생성된다. 마지막으로 CU의 크기가 8x8 인 경우에는 '2NxnU', ‘2NxnD', 'nLx2N', 'nRx2N' 형태가 존재하지 않는다. 그래서 CU의 크기가 8x8 인 경우에는 움직임 병합 후보 리스트의 생성횟수는 320번이다. 따라서 모든 PU형태와 위치 그리고 크기를 고려할 경우, 하나의 LCU에서 움직임 병합 후보 리스트의 생성횟수는 총 593번이 된다. 이는 PU기반의 움직임 병합 방법이 높은 복잡도를 가지는 이유 중의 하나이다. 따라서 본 논문에서는 이러한 복잡도를 감소시키고 병렬성을 제공하기 위한 방안을 제시한다.

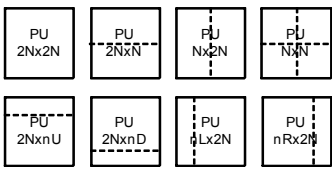

그림 2. PU의 형태들
Fig. 2. Types of PU

표 1. 움직임 병합 후보 리스트의 생성 횟수
Table. 1. The number of the generation of the merge candidate list

CU 크기	움직임 병합 후보 리스트 생성 횟수
64x64	13
32x32	13 * 4 = 52
16x16	13 * 4 * 4 = 208
8x8	5 * 4 * 4 * 4 = 320
전체	593

III. CU기반 움직임병합 후보리스트 생성 방법

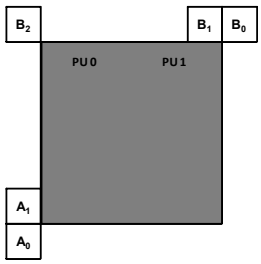

그림 3. CU 기반 움직임 병합 후보 리스트 생성을 위한 공간적 주변 블록들
Fig. 3. Spatial neighboring blocks to be used for the CU-based merge candidate list construction

그림 3은 CU 기반 움직임 병합 후보 리스트 생성을 위해 사용되는 공간적 주변 블록들을 나타낸다. 그림 3과 같이 CU내의 첫 번째 PU블록(PU 0)과 두 번째 PU블록(PU 1)은 공간적으로 동일한 위치에 있는 주변 블록들을 사용한다. 또한 참조 픽쳐에서 현재 CU와 동일한 위치에 있는 시간적 주변 블록도 동일하게 사용한다. 즉, CU내의 모든 PU 블록은 PU형태 및 위치, 크기에 관계없이 오직 하나의 공통 움직임 병합 후보 리스트를 생성하여 움직임 정보를 부호화하는데 사용한다.

부호화시 기존 PU기반 움직임 병합 후보 리스트 생성 방법은 하나의 CU당 최대 17번을 생성하는 반면, CU기반 움직임 병합 후보 리스트 생성 방법은 하나의 CU당 오직 한번만 생성한다. 또한 복호화시 기존 방법은 하나의 CU당 최대 4번 생성하는 반면 제안 방법은 하나의 CU당 오직 한번만 생성한다. 따라서 움직임 병합 후보 리스트를 생성하는 횟수 관점에서 계산 복잡도가 크게 감소한다.

하드웨어로 구현 시 기존 방법은 서로 다른 PU형태마다 모듈이 각각 필요하므로 총 17개의 서로 다른 모듈이 필요

하다. 하지만 제안 방법은 오직 하나의 모듈만 필요하므로 하드웨어 구현이 간단하다는 장점을 갖는다. 또한 기존 방법의 경우, 하나의 CU가 여러 개의 PU들로 나누어져 부호화 될 경우, PU들간의 의존성 때문에 병렬성 제공이 힘들었다. 하지만, 제안 방법의 경우에는 움직임 병합 후보 리스트를 PU들이 공유하여 사용하기 때문에 PU들 간의 독립적으로 부호화 및 복호화가 가능하게 되어 완벽한 병렬성 제공이 가능하다.

IV. 성능 평가 및 분석

본 논문에서 제안하는 방법을 검증하기 위해 HEVC HM 4.0 참조 소프트웨어(reference software)에 제안하는 방법을 구현하였다. 실험 영상 및 기타 실험 조건은 HEVC 공통 실험조건[8]을 준수하였다. 제안하는 방법의 계산 복잡도를 측정하기 위해 부호화 및 복호화하는 과정의 평균 시간(Encoding Time)을 측정하였다. 또한 부호화 효율을 측정하기 위해 평균적인 bit-rate 증가량을 나타내는 BD-Bitrate 방법[9]을 사용하였다. 여기서 BD-Bitrate 값이 음수이면 압축률이 향상되었다는 것을 의미한다.

실험 결과(표 2), 제안방법이 기존 방법보다 인코더의 복잡도는 3% ~ 6% 감소하였다. 또한 부호화 효율 관점에서 제안 방법이 기존 방법보다 평균적으로 비트량이 0.2% ~ 0.5% 증가하였다. 제안하는 방법은 움직임 병합 후보 리스트를 생성하는 방법의 계산 복잡도를 감소시키고 병렬성을 제공하기 위한 것이다. 즉, 제안 방법은 복잡도를 감소시키고 병렬성을 제공한다는 장점이 있는 반면 부호화 효율을 감소시키는 단점이 존재한다.

표 2. 제안 방법의 실험 결과
Table. 2. Simulation results of the proposed method

	Random Access HE			Random Access LC		
	Y	U	V	Y	U	V
Class A	0.2%	0.0%	0.0%	0.2%	0.1%	0.2%
Class B	0.2%	0.1%	0.2%	0.1%	0.2%	0.2%
Class C	0.3%	0.2%	0.3%	0.3%	0.2%	0.3%
Class D	0.4%	0.3%	0.3%	0.4%	0.1%	0.4%
Overall	0.3%	0.2%	0.2%	0.2%	0.2%	0.3%
Enc Time[%]		97%			96%	
Dec Time[%]		100%			100%	

	Low delay B HE			Low delay B LC		
	Y	U	V	Y	U	V
Class B	0.3%	0.3%	0.5%	0.3%	0.3%	0.4%
Class C	0.4%	0.5%	0.6%	0.4%	0.7%	0.3%
Class D	0.5%	0.7%	0.3%	0.4%	0.6%	0.3%
Class E	0.9%	1.2%	1.5%	0.7%	0.7%	0.5%
Overall	0.5%	0.6%	0.7%	0.4%	0.5%	0.4%
Enc Time[%]		94%			96%	
Dec Time[%]		100%			99%	

V. 결 론

본 논문에서는 계산 복잡도를 감소시키고 완벽한 병렬성을 제공하기 위하여 CU 기반의 움직임 병합 후보 리스트 생성 방법을 제안하였다. 제안방법의 실험 결과, 기존 방법보다 인코더의 복잡도는 3% ~ 6% 감소하였으며, 디코더의 복잡도는 거의 변화가 없었다. 반면 제안 방법이 기존 방법보다 부호화 효율이 감소되었다. 하지만 제안방법으로 인해 움직임 병합 후보 리스트를 생성하는 횟수가 크게 감소할 수 있었다. 또한 제안 방법은 CU당 오직 하나의 움직임 병합 후보 리스트만 필요하므로 하드웨어 구현이 간단하다는 장점을 갖는다. 그리고 제안 방법은 PU들 간 움직임 병합 후보 리스트를 공유하여 사용하기 때문에 PU들 간의 병렬성 제공이 가능하다.

참 고 문 헌

[1] Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, “WD4: Working Draft 4 of High-Efficiency Video Coding,” JCTVC-F803, Torino, Italy, July 2011.

[2] K. McCann, W.-J. Han, I.-K. Kim, J.-H. Min, E. Alshina, A. Alshin, T.Lee, J. Chen, V. Seregin, S. Lee, Y.-M. Hong, M.-S. Cheon, and N.Shlyakhov, “Samsung’s Response to the Call for Proposals on Video Compression Technology,” JCTVC-A124, April 2010.

[3] G. Laroche, J. Jung and B. Pesquet-Popescu, “RD optimized coding for motion vector predictor selection,” IEEE Trans. Circuits Syst. Video Technol., vol. 18, pp.1247-1257, 2008.

[4] M. Winken , S. Boße, B. Bross, P. Helle, T. Hinz, H. Kirchhoffer, H. Lakshman, D. Marpe, S. Oudin, M. Preiß, H. Schwarz, M. Siekmann, K. S¨uhring and T. Wiegand, “Description of video coding technology proposal by HHI,” JCTVC-A116, Dresden, Germany, April 2010.

[5] T.K. Tan, W.-J. Han, B. Bross, J. Jung, K. McCann, Y. Suzuki, G. Clare, H. Schwarz and A. Fujibayashi, “BoG report of CE9: Motion Vector Coding,” JCTVC-D441, January 2011.

[6] J. Jung, B. Bross, “CE9: Summary report for CE9 on motion vector coding,” JCTVC-D149, January 2011.

[7] T. Sugio, T. Nishi, “Parsing Robustness for Merge/AMVP,” JCTVC-F470, Torino, Italy, July 2011.

[8] Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, “Common test conditions and software reference configurations,” JCTVC-F900, Torino, Italy, July 2011.

[9] G. Bjøntegaard, “Calculation of average PSNR differences between RD-curves,” ITU-T SG16 Q.6, VCEG-M33, Texas, USA, April 2001.