HEVC 부호화기에서 CU 정보 병합 알고리즘을 이용한 빠른 CU 부호화 방법

*이재영 **한종기

세종대학교

*ljy321456@naver.com, **hjk@sejong.edu

Fast encoding algorithm using CU merge scheme in HEVC

*Lee, Jae-Yung **Han, Jong-Ki

Sejong University

요약

MPEG과 ITU-T에서 최근 표준화가 진행되고 있는 HEVC는 H.264/AVC에 비해, CU(coding unit), PU(prediction unit), TU(transform unit)의 다양한 형태 분할 단위를 갖는 것을 큰 특징으로 한다. 이 중, CU와 TU는 쿼드트리 형태의 재귀적 분할 구조를 가지도록 구성되는데, 압축 효율은 향상시키지만 높은 부호화 복잡도를 갖는 단점이 있다. 본 논문에서는 이러한 재귀적 분할 구조를 변환하여 가장 작은 CU의 정보를 병합하여 큰 CU의 정보를 빠르게 결정하는 방법을 제안한다. 제안한 방법을 HEVC의 CU 부호화에 적용한 결과, 부호화 복잡도를 32-45% 가량 감소시키면서 압축 효율 하락은 0.6-0.9%로 억제 할 수 있었다. 또한, HM6.1에 구현되어 있는 고속 탐색 알고리즘과 비교 할 경우, 압축 효율 하락을 0.2-0.3%로 억제하면서 부호화 복잡도를 8-12% 감소시킬 수 있었다.

1. 서론[1)]

최근 MPEG과 ITU-T에서는 H.264/AVC의 압축 효율을 2배 이상 향상시키는 것을 목표로 하는 신규 영상 압축 기술인 High Efficiency Video Coding(HEVC)에 대한 표준화를 공동으로 진행하고 있다[1]. HEVC는 H.264/AVC와 마찬가지로 motion compensation과 DCT를 함께 사용하는 hybrid coding 방법에 기반하고 있으나, 전체 압축 시스템을 구성하는 각각의 요소 기술에서 많은 개선과 발전이 이루어져, 현재 H.264/AVC 대비 압축 효율을 약 40% 이상 향상시킬 수 있는 것으로 알려져 있다[2].

H.264/AVC에서는 하나의 slice를 16x16 크기를 갖는 매크로블록(macroblock; MB)들로 나누고, 분할모드에 따라16x16, 16x8, 8x16, 8x8 등의 크기로 분할한 뒤, 8x8 분할의 경우 재차 8x8, 8x4, 4x8, 4x4 단위로 나눌 수 있도록 허용하는 구조를 가지고 있다. 이에 비해, HEVC에서는 최소 8x8에서 최대 64x64까지의 크기를 가질 수 있는 최대코딩단위(largest coding unit; LCU)들로 slice를 나누고, 각각의 LCU들이 쿼드트리 형태로 재귀적 분할되는 자유로운형태를 갖고 있다. 이렇게 분할된 단위를 코딩단위(coding unit; CU)라고 하며, 이 CU들은 추가적으로 예측단위(prediction unit; PU)로 분할된다.

인트라 예측 혹은 인터 예측 과정이 수행되고 나면, 원본 신호와 예측신호 간 차이인 잔차신호를 변환하는 과정을 거치게 된다. 이 과정에서, H.264/AVC의 경우 분할모드에 따라 전체 MB에 동일한 형태의 변환 크기가 사용되는 반면, HEVC에서는 CU를 기준으로 재차 재귀적 분할을 수행하고, 이 변환단위(transform unit; TU) 크기의 변환을 잔차 신호에 적용한다. 이러한 CU 및 TU의 재귀적 유닛 구조는 H.264/AVC에 비해 압축 성능을 향상시키는데 있어 가장 큰 역할을 하지만, 부호화기의 복잡도가 크게 증가하는 문제점을 갖고 있다.

H.264/AVC의 경우 부호화기의 복잡도를 낮추기 위한 많은 연구들이 있어 왔다. [3]에서는 매크로블록의 rate-distortion cost를 분석하여 skip 모드로 미리 결정하기 위한 조건을 제안하였고, [4]에서는 잔차 신호의 변환 계수가 모두 0이 되는 조건을 분석함으로써 부호화기의 복잡도를 낮출 수 있음을 보였다. 한편, 이러한 조건들은 HEVC에 적용한 경우에도, 재귀적분할 유닛 구조 전체를 탐색하는 대신 일부분만을 탐색함으로서 부호화기 연산량을 크게 감소시키면서 압축 성능하락을 최소화 할 수 있다[5]. 이러한 방법들은 매우 효과적으로 부호화기의 연산량을 감소시킬 수 있지만, 변환 계수가 모두 0인 경우에 대한 특성만을 사용하기 때문에 그렇지 않은 경우에 대한 분석 및 고속 탐색 알고리즘은 부호화기의 연산량을 추가적으로 감소시킬 수 있을 것으로 예측된다.

본 논문에서는 기존의 쿼드트리 형태의 재귀적 유닛 구조를 변경하여, LCU마다 가장 작은 CU단위부터 부호화하여 발생한 정보를 이용하여 큰 CU의 정보를 빠르게 결정하는 방법을 제안하였다. 또한 제안하는 방법과 HM6.1에 적용되어 있는 고속 탐색 알고리즘과 비교하였을 경우의 성능 비교에 대해서도 그 결과를 보인다.

본 논문의 제 II장에서는 HEVC의 재귀적 분할 구조에 대해 개략적으로 소개한다. 제 III장에서는 작은 CU의 정보를 이용하여 큰 CU의 정보를 결정하는 방법을 설명하고, 제 IV장에서 이에 대한 실험 결과를 보인 후, 마지막으로 제 V장에서 결론을 맺는다.

1) 연락저자: 한종기

본 연구는 지식경제부 및 정보통신산업진흥원의 IT융합 고급인력과정 지원사업의 연구결과로 수행되었음
(NIPA-2012-H0401-12-1003)

2. HEVC의 재귀적 분할 구조

HEVC에서는 slice를 최소 8x8에서 최대 64x64까지의 크기를 가질 수 있는 LCU들로 분할한다. 이 LCU는 최소 크기가 8x8이 될 때까지 쿼드트리 형태로 재귀적 분할될 수 있으며, 이 때, 분할된 각각의 영역을 CU라고 한다. 이러한 각각의 CU들에 대해, 인터 예측 혹은 인트라 예측 방법이 적용되며, 각각 CU들은 여러 모양의 PU가 적용될 수 있다. 즉, 각 CU마다 다른 모양의 PU가 결정 될 수 있으므로, 하나의 LCU 내에서 이러한 CU와 PU들이 최적의 조합으로 혼합되어 사용되는 것이 가능하다. <그림 1>은 HEVC의 LCU, CU, 그리고 PU 간 관계를 보인 것이다.

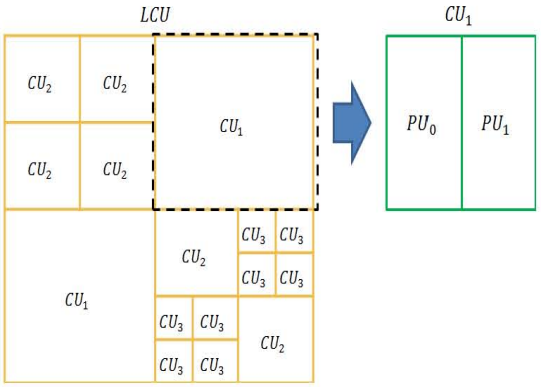

그림 1. HEVC의 LCU, CU, PU간 관계

3. CU 정보의 병합 방법

LCU를 가장 작은 CU크기로만 부호화하였을 때 발생하는 정보를 병합하여 더 큰 CU의 정보를 빠르게 결정한다. 이 방법은 쿼드트리 형태로 표현되는 재귀적 분할 구조를 기존의 LCU부터 SCU까지 분할 부호화가 아닌 SCU부터 수행하여 LCU까지 병합 부호화를 수행한다. 큰 크기의 CU로부터 작은 크기의 CU로 영역을 세밀하게 쪼개는 것이 아니라 작은 영역의 세밀한 정보를 병합하여 큰 영역의 정보로 사용하는 방법이다.

큰 CU로부터 작은 CU의 정보를 예측하는 것은 CU를 실제로 분할하여 부호화 하지 않았기에 예측 한 정보와 실제 부호화된 정보가 많은 차이를 가질 수 있다. 하지만 작은 CU들의 정보로부터 큰 CU를 예측하는 것은 큰 CU안에 모든 영역에 대한 정보를 가지고 있기 때문에 예측 한 정보와 실제 큰 CU로 부호화된 정보가 굉장히 유사하다.

CU를 병합하는 방법은 CU 스캔 방향에 따라 CU 분할 깊이 k를 가지는 CU를 4개씩 그룹으로 나누었을 때 i번째 그룹을 Ω_i^k라고 하면 Ω_i^k안에 존재하는 모든 MV들의 공분산 행렬의 고유값의 합을 J_i^k라 하고, LCU안에 존재하는 모든 MV들의 공분산 행렬의 고유값의 합을 J_{LCU}라 할 때 J_i^k가 J_{LCU}보다 작은 경우 큰 CU로 병합할 수 있다고 판단한다. 이는 다음과 같이 나타낼 수 있다.

$$State = \begin{cases} J_i^k < J_{LCU} & \text{Merge} \\ otherwise & \text{Keep the current state} \end{cases} \quad (1)$$

여기서 J_i^k와 J_{LCU}는 다음과 같이 나타낼 수 있다.

$$\begin{aligned} J_i^k &= |\lambda_1(\gamma_{\Omega_i^k})| + |\lambda_2(\gamma_{\Omega_i^k})| \\ J_{LCU} &= |\lambda_1(\gamma_{LCU})| + |\lambda_2(\gamma_{LCU})| \end{aligned} \quad (2)$$

여기서 $\gamma_{\Omega_i^k}$는 Ω_i^k안에 존재하는 MV들의 집합이며 γ_{LCU}는 LCU안에 존재하는 MV들의 집합이고 λ_1와 λ_2는 공분산 행렬 C_i^k와 C_{LCU}의 고유값으로 공분산 행렬은 다음과 같이 표현 될 수 있다.

$$C_i^k = \begin{bmatrix} R_i^k(xx), & R_i^k(xy) \\ R_i^k(yx), & R_i^k(yy) \end{bmatrix} \quad (3)$$

$$C_{LCU} = \begin{bmatrix} R_{LCU}(xx), & R_{LCU}(xy) \\ R_{LCU}(yx), & R_{LCU}(yy) \end{bmatrix} \quad (4)$$

식 (1)에서 State가 Merge가 된다면 큰 CU의 PU정보를 결정해야 한다. 하나의 CU에서 사용 가능한 PU의 모양은 크게 Inter Frame에서 2Nx2N, Nx2N, 2NxN이 있다. <그림 2>와 같이 각각의 PU마다 MV들이 존재 하는데 큰 CU의 MV들을 작은 CU의 MV들로부터 예측하여 사용할 수 있다.

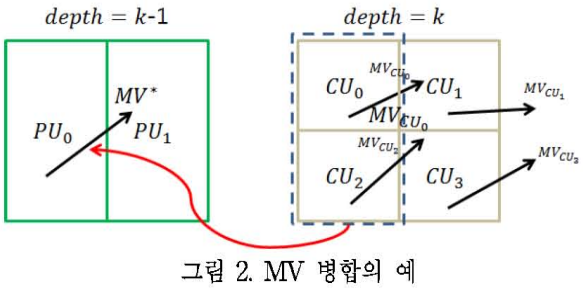

그림 2. MV 병합의 예

<그림 2>에서 MV^*는 PU_0와 대응되는 위치의 작은 CU들 안에 존재하는 MV들의 집합을 $\gamma_{PU_m}^k$라 할 때 $\gamma_{PU_m}^k$으로 생성된 CMV_j중 최적의 Motion Cost를 갖는 MV이다. 이를 표현하면 아래와 같다.

$$\begin{aligned} MV^* &= MV_{Best} \\ Best &= \arg\min(J_{Motion}(CMV_j)),\ j \in Merge, \gamma_{PU}^k \end{aligned} \quad (5)$$

식 (5)에서 k는 CU의 분할 깊이이고 m은 $k-1$ 분할 깊이 CU의 PU에서 Ω_i^k로 대응되는 $k-1$ 분할 깊이 CU안의 PU 인덱스이며, $Merge$는 $\gamma_{PU_m}^k$를 병합하여 만든 MV이다. 여기서 MV_{Merge}는 다음과 같다.

$$MV_{Merge} = \sum_{j \in Merge, \gamma_{PU_m}^k} w_j \gamma_{PU_m}^k(j) \quad (6)$$

$\gamma_{PU_m}^k(j)$는 MV들의 집합 $\gamma_{PU_m}^k$에서 j번째 MV를 나타내며, w_j는 다음과 같이 나타낼 수 있다.

$$w_j = \frac{C_j}{\sum_{n \in \gamma_{PU_m}^k} C_n} \quad (7)$$

$$C_i = 1 - \frac{AMC_i}{\sum_{i \in \gamma_{PU_m}^k} AMC_i} \quad (8)$$

$$AMC_i = \frac{PU^k(\mathrm{i})\text{의 Motion Cost}}{PU^k(\mathrm{i})\text{의 면적}} \quad (9)$$

식(7) C_j는 AMC_i의 가중치이며 AMC_i는 CU 분할 깊이가 k인 CU에 속해있는 PU들의 한 개의 픽셀당 J_{Motion}값이다. 식 (6)을 이용하여 한 픽셀당 J_{Motion}값을 가중치로 하여 $\gamma_{PU_m}^k$의 MV들을 Weighted Sum한 값이 MV_{Merge} 이다.

식(5)를 통해 $k-1$의 분할 깊이를 갖는 CU의 PU 모양마다 $J_{Motion}(MV_{Shape}^*)$를 계산 후 가장 작은 $J_{Motion}(MV_{Shape}^*)$를 가지는 PU 모양을 $k-1$의 분할 깊이를 갖는 CU에 적용한다. 이를 다음과 같이 표현할 수 있다.

$$Est.Shape = \arg\min(J_{Motion}(MV_{Shape}^*)) \quad (10)$$
$$Shape = 2\mathrm{Nx}2\mathrm{N}, 2\mathrm{NxN}, \mathrm{Nx}2\mathrm{N}$$

식(10)에서 예측 된 PU모양과 MV^*를 구한 다음 $k-1$의 분할 깊이를 갖는 CU의 $SKIP$, $Est.Shape$, $k-depth\ CU's$의 J_{RD}를 비교하여 가장 작은 J_{RD}를 갖는 모드를 선택한다. 이는 다음과 같이 나타낼 수 있다.

$$Opt.CU = \arg\min(J_{RD}(Est.Shape), J_{RD}(SKIP), J_{RD}(CU's)) \quad (11)$$

LCU안에 있는 모든 Ω_i^k에 대해 식(11)번을 계산한 후 다음 분할 깊이에서의 집합 Ω_i^{k-1}에서 다시 CU 정보를 병합하는 과정을 반복한다.

4. 실험 결과

제안하는 방법의 효과를 검증하기 위해, HM6.1[6]에 제안한 방법을 구현하고, 압축 및 부호화 연산량 측면에서 비교하였다. 압축 효율 및 부호화 연산량을 비교하기 위해서 JCT-VC에서 사용하는 공통 테스트 조건[7] 중 random-access scenario, low-delay B scenario와 low-delay P scenario를 사용하였다. HM6.1을 기준으로 상대적 변화를 측정하기 때문에 BD-rate[8]에 의해서 구한 동일 PSNR 대비 비트 증가율이 양수일 경우 압축 효율이 하락하였음을 의미하며 실행 속도가 100%보다 작을 경우 부호화기 연산량이 절감 되었음을 의미한다.

표 1. Random Access 성능 비교

	Random Access Main			Random Access HE10		
	Y	U	V	Y	U	V
Class A	0.1	0.0	0.1	0.3	0.1	0.0
Class B	0.4	0.3	0.2	0.8	0.5	0.4
Class C	1.0	0.4	0.7	1.0	0.9	0.7
Class D	1.5	0.7	0.8	1.5	1.2	0.9
Overall	0.8	0.4	0.5	0.9	0.6	0.5
Enc Time[%]	50%			55%		

표 2. Low Delay B 성능 비교

	Random Access Main			Random Access HE10		
	Y	U	V	Y	U	V
Class B	0.1	0.0	0.1	0.1	0.1	0.6
Class C	0.4	0.3	0.2	0.3	0.5	0.8
Class D	1.0	0.4	0.7	0.9	0.6	1.1
Class E	1.3	0.7	0.8	1.4	1.3	1.3
Overall	0.7	0.4	0.6	0.9	0.9	1.2
Enc Time[%]	63%			68%		

표 3. Low Delay P 성능 비교

	Random Access Main			Random Access HE10		
	Y	U	V	Y	U	V
Class B	0.1	0.0	0.1	0.1	0.1	0.2
Class C	0.4	0.1	0.7	0.5	0.3	0.4
Class D	0.7	0.4	0.8	0.9	0.5	0.7
Class E	1.0	0.6	1.0	1.3	0.8	0.9
Overall	0.6	0.3	0.7	0.7	0.5	0.6
Enc Time[%]	55%			60%		

<표 1>, <표 2>, <표 3>은 부호화기의 복잡도를 낮추는 알고리즘을 적용하지 않은 HM6..1을 기준으로 제안하는 방법과 비교를 한 결과이며 <표 4>, <표 5>, <표 6>은 HM6..1에 부호화시 복잡도를 낮추는 알고리즘[3]과 [5]를 함께 적용한 것을 기준으로 제안하는 방법을 비교한 결과이다.

표 4. Random Access 성능 비교

	Random Access Main			Random Access HE10		
	Y	U	V	Y	U	V
Class A	0.0	0.0	0.0	0.0	0.0	0.1
Class B	0.1	0.0	0.1	0.2	0.1	0.1
Class C	0.2	0.0	0.0	0.2	0.2	0.2
Class D	0.3	0.1	0.1	0.4	0.1	0.3
Overall	0.2	0.0	0.1	0.3	0.1	0.2
Enc Time[%]	92%			89%		

표 5. Low Delay B 성능 비

	Random Access Main			Random Access HE10		
	Y	U	V	Y	U	V
Class B	0.1	0.0	0.1	0.0	0.0	0.0
Class C	0.1	0.1	-0.2	0.1	0.0	-0.8
Class D	0.1	0.4	0.4	0.2	0.1	0.2
Class E	0.4	0.5	0.4	0.2	0.1	0.3
Overall	0.2	0.3	0.2	0.2	0.1	-0.1
Enc Time[%]	90%			88%		

표 6. Low Delay P 성능 비교

	Random Access Main			Random Access HE10		
	Y	U	V	Y	U	V
Class B	0.0	0.0	0.0	0.0	0.0	0.
Class C	0.1	0.0	0.0	0.0	0.1	0.1
Class D	0.2	0.1	0.0	0.2	0.1	0.1
Class E	0.1	0.0	0.1	0.1	0.0	0.0
Overall	0.2	0.0	0.1	0.2	0.1	0.1
Enc Time[%]	91%			88%		

5. 결 론

본 논문에서는 쿼드트리 형태의 재귀적 분할 구조를 변경하여 LCU로부터 SCU까지 재귀적 분할을 SCU부터 LCU까지 재귀적 병합하는 형태의 구조와 병합하는 방법을 제안하였다. 제안한 방법은 기존의 HEVC방법 보다 45-32% 감소 시키면서 압축 효율의 저하는 0.6-0.9%로 억제할 수 있다. 기존 HM6.1에 적용되어 있는 고속 탐색 알고리즘과 비교하는 경우, 압축 효율 하락을 0.2-0.3%로 억제 하면서 부호화기 연산량을 8-12% 감소시킬 수 있었다.

참 고 문 헌

[1] G. J. Sullivan, J.-R. Ohm, "Recent Developments in Standardization of High Efficiency Video Coding (HEVC)", SPIE Applications of Digital Image Proc. XXXIII, Proc. SPIE, Vol. 7798, paper 7798-30, Aug. 2010.

[2] T. Wiegand, J.-R. Ohm, G. J. Sullivan, W. J. Han, R. Joshi, T. K. Tan and K. Ugur, "Special Section on the Joint Call for Proposals on High Efficiency Video Coding (HEVC) Standardization," IEEE Transactions on Circuits and Systems for Video Technology, Vol. 20, No. 12, pp. 1661-1666, Dec. 2010.

[3] C. S. Kannangara, E. G. Richardson, M. Bystrom, J. R. Solera, Y. Zhao, A. Maclennan, R. Cooney, "Low complexity skip prediction for H.264 through Lagrangian cost estimation," IEEE Transactions on Circuits and Systems for Video Technology, Vol. 16, No. 2, pp. 202-208, Feb. 2006.

[4] H. Wang, S. Kwong, C. W. Kok, "An efficient mode decision algorithm for H.264/AVC encoding optimization," IEEE Transactions on Multimedia, Vol. 9, No. 4, pp. 882-888, Jun. 2007.

[5] R. H. Gweon, Y. L. Lee, J. Lim, "Early termination of CU encoding to reduce HEVC complexity," JCTVC-F045, 6th JCT-VC meeting, Jul. 2011, Torino, Italy.

[6] B. Bross, W. J. Han, J.-R. Ohm, G. J. Sullivan, T. Wiegand, "High efficiency video coding (HEVC) text specification draft 6," Document JCTVC-H1003, San José, USA, February, 2012.

[7] F. Bossen, "Common test conditions and software reference configurations,"Document JCTVC-H1100 San José, USA, February, 2012.

[8] G. Bjontegarrd, "Calculation of average PSNR differences between RD curves," in ITU-T SC16/Q6 13th VCEG meeting, No. VCEG-M33, Austin, TX, Apr. 2001.