높은 처리량을 갖는 HEVC 복호기를 위한 CABAC 변환 계수 부호화 방법

최정아, 호요성
광주과학기술원 정보통신공학부
jachoi@gist.ac.kr, hoyo@gist.ac.kr

요 약

HEVC (high efficiency video coding) 표준의 엔트로피 부호화 방법인 CABAC(context-based adaptive binary arithmetic coding)은 높은 부호화 성능을 제공하지만, 복잡도가 높다는 단점을 갖는다. 특히, CABAC의 정규 부호화 엔진은 이전 빈의 부호화를 통해 업데이트된 확률 상태와 범위를 사용하기 때문에 높은 데이터 의존성을 보이며, 확률 구간을 읽고 현재 상태를 판단하는데 많은 시간이 소요된다. 본 논문에서는 CABAC 변환 계수 부호화의 처리량을 개선하기 위해 부호화된 빈의 통계적 특성을 이용한 복잡도 모델을 제안하고, 변환 블록 결정을 위한 율-왜곡 최적화 비용함수에 복잡도 모델을 추가하는 방법을 제안한다. 실험을 통해 제안하는 방법이 큰 부호화 성능 변화 없이 약 11.6%의 복잡도를 감소시키는 것을 확인했다.

1. 서론

디지털 TV의 보급이 확산되면서 시장 전반적으로 대형 TV에 대한 선호도가 높아지고 있으며, 삼성전자와 LG전자 등의 대기업들은 앞다투어 대형 TV를 출시하고 있다. 또한 모바일 시장에서도 기본 화면 해상도가 HD(high-definition)급인 스마트폰이 출시되기 시작했다. 이처럼 소비자들의 차세대 영상 서비스에 대한 수요에 발맞춰 다양한 차세대 멀티미디어 제품들이 출시되는 가운데, 고화질 영상처리 기술에 대한 관련 산업계 및 학계의 관심이 높아지고 있다.

이러한 고품질, 고해상도 영상 콘텐츠는 H.264/AVC 표준을 비롯한 기존 영상 압축 표준으로는 사용자가 원하는 수준의 영상 서비스를 제공하는데 한계가 있다. 따라서 영상 압축 분야의 양대 표준화 단체인 ISO/IEC MPEG과 ITU-T VCEG은 2010년 1월 교토 회의에서 차세대 영상 압축 표준 개발을 위해 JCT-VC(joint collaborative team on video coding)라는 공동 협력팀을 구성하기로 합의하고, 2013년 1월 H.264/AVC 표준보다 약 두 배 부호화 효율을 향상시킨 HEVC (high efficiency video coding) 표준을 완성했다 [1][2].

HEVC 표준은 기본적으로 기존 영상 압축 표준들과 유사한 하이브리드 부호화 구조를 가지며, 세부적으로는 기존 영상 압축 표준들과 차별되는 부호화 기술들인 최대 64×64 부호화 단위와 최대 32×32 변환 단위를 지원하는 쿼드트리 구조, 35 가지의 화면내 예측 모드, 병합 모드 (merge mode), 처리량을 향상시킨 단일 엔트로피 부호화, 간략화된 디블록킹 필터, 새로운 형태의 루프 필터인 SAO (sample adaptive offset), 변환 생략 모드 (transform skip mode) 등을 포함한다.

HEVC 표준의 엔트로피 부호화 방법은 H.264/AVC의 엔트로피 부호화 방법 중 하나인 CABAC(context-based adaptive binary arithmetic coding)으로, 전체적인 동작과 명칭은 동일하지만 기존 CABAC의 처리량을 대폭 향상시킨 기술이다. 그 간의 HEVC 표준화 회의에서 Nguyen [3], Kim [4], Lainema [5], Chen [6] 등 많은 전문가들이 CABAC의 처리량 개선과 관련해 다양한 기술을 제안했다. 특히, Chen이 개발한 HTB (high throughput binarization) 모드는 동작이 단순하면서 성능이 뛰어나 HEVC 표준에 채택되었다.

하지만 아직도 CABAC의 높은 복잡도는 큰 문제로 남아있으며, 특히 CABAC 변환 계수 처리에 따른 복잡도는 HEVC 복호기의 전체 복잡도의 30%를 차지한다는 보고가 있었다 [7]. 따라서 본 논문에서는 높은 처리량을 갖는 CABAC 변환 계수 부호화 방법을 제안한다. 먼저 CABAC 변환 계수 부호화를 수행한 후 얻은 비트스트림들의 통계적 특성을 이용해 CABAC의 변환 계수 부호화 복잡도를 모델링하고, 이를 통해 변환 블록 크기 결정 단계에서 비트율, 왜곡뿐 아니라 복잡도까지 함께 고려하는 방법을 제안한다.

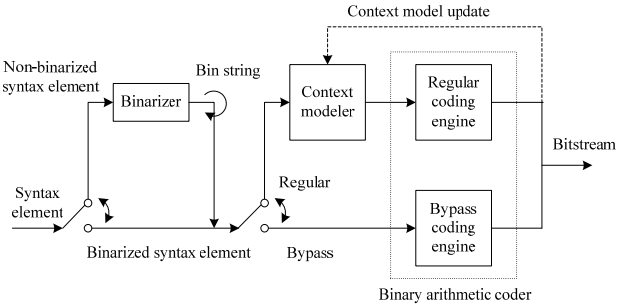


그림 1. CABAC 블록도

2. HEVC 엔트로피 부호화의 개요

2.1 CABAC의 구조

그림 1은 단일 구문요소를 부호화하기 위한 CABAC의 블록도를 보여준다. CABAC의 부호화 과정은 먼저 입력 신호가 이진값이 아닌 구문요소인 경우에 이진화를 통해 입력 신호를 이진값로 변환한다. 입력 신호가 이미 이진값인 경우에는 이진화를 거치지 않고 바이패스 된다. 여기서, 이진값을 구성하는 각각의 이진수를 빈(bin)이라고 한다. 예를 들어, 이진화된 후의 이진값이 110인 경우, 1, 1, 0 각각을 하나의 빈이라고 한다.

이진화된 빈들은 정규 부호화 엔진 또는 바이패스 부호화 엔진으로 입력된다. 정규 부호화 엔진은 해당 빈에 대해 확률값을 반영하는 문맥을 모델을 할당하고, 할당된 문맥 모델에 기반해 해당 빈을 부호화한다. 정규 부호화 엔진에서는 각 빈에 대한 부호화를 수행한 뒤에 해당 빈에 대한 확률 모델을 갱신할 수 있다. 이렇게 부호화되는 빈들을 문맥 부호화 빈(context-coded bin)이라 한다.

바이패스 부호화 엔진은 입력된 빈에 대해 확률을 추정하는 절차와 부호화 후에 해당 빈에 적용했던 확률 모델을 갱신하는 절차를 생략한다. 문맥을 할당하는 대신 균일한 확률 분포를 적용해 입력되는 빈을 부호화함으로써 부호화 속도를 향상시킨다. 이렇게 부호화되는 빈들을 바이패스 빈(bypass bin)이라 한다.

엔트로피 부호화는 정규 부호화 엔진을 통해 부호화를 수행할 것인지, 바이패스 부호화 엔진을 통해 부호화를 수행할 것인지를 결정하고, 부호화 경로를 스위칭할 수 있다. 엔트로피 복호화는 부호화와 동일한 과정을 역순으로 수행한다.

2.2 CABAC의 변환 계수 부호화

표 1은 변환 블록에서 양자화된 변환 계수, 즉 잔여신호 부호화를 위한 구문요소를 보여준다. 여기서 음영 처리된 구문요소는 문맥 부호화 빈을 이용해 부호화 되는 것을 의미한다. CABAC으로 변환 계수를 부호화하는 과정은 다음과 같다.

표 1. 변환 계수 부호화를 위한 구문요소

last_significant_coeff_x_prefix
last_significant_coeff_y_prefix
last_significant_coeff_x_suffix
last_significant_coeff_y_suffix
coded_sub_block_flag
sig_coeff_flag
coeff_abs_level_greater1_flag
coeff_abs_level_greater2_flag
coeff_abs_level_remaining
coeff_sign_flag

먼저 구문요소 *last_significant_coeff_x*와 *last_significant_coeff_y*를 이용해 변환 블록내의 마지막 0이 아닌 계수의 (x, y) 위치 정보를 부호화한다. 그 다음, 변환 블록을 4×4 하위 블록으로 분할한 후, 각 4×4 하위 블록마다 1비트의 구문 요소 *coded_sub_block_flag*를 사용해 현재 하위 블록 내에 0이 아닌 계수가 존재하는지를 나타낸다.

*coded_sub_block_flag*가 0이면 더 이상 전송할 정보가 없으므로 현재 하위 블록에 대한 부호화 과정을 종료한다. 반대로 값이 1이면 *sig_coeff_flag* 부호화 과정을 계속해서 수행한다. 마지막 0이 아닌 계수를 포함하는 하위 블록은 *coded_sub_block_flag* 부호화가 불필요하고, 변환 블록의 DC 정보를 포함하고 있는 하위 블록은 0이 아닌 계수를 포함할 확률이 높으므로 *coded_sub_block_flag*을 부호화하지 않고, 이 값이 1이라 가정한다.

만약 *coded_sub_block_flag*가 현재 하위 블록 내에 0이 아닌 계수가 존재한다고 나타내면 역으로 스캔된 순서에 따라 이진값을 갖는 *sig_coeff_flag*를 부호화한다. 스캔 순서에 따라 각각의 계수에 대해 1비트 구문 요소 *sig_coeff_flag*를 부호화한다. 만약 현재 스캔 위치에서 계수의 값이 0이 아니면 *sig_coeff_flag*값은 1이 된다. 여기서, 마지막 0이 아닌 계수를 포함하고 있는 하위 블록의 경우, 마지막 0이 아닌 계수에 대해서는 *sig_coeff_flag*를 부호화할 필요가 없으므로 부호화 과정이 생략된다.

*sig_coeff_flag*가 1인 경우에만 레벨 정보 부호화가 수행되는데, 레벨 정보 부호화 과정에서는 세 개의 구문 요소를 사용한다. 먼저 해당 계수가 1보다 큰 지 여부를 나타내는 *coeff_abs_level_greater1_flag*가 부호화되고, 이 값이 1인 경우에만 해당 계수가 2보다 큰 지 여부를 나타내는 *coeff_abs_level_greater2_flag*가 부호화된다. 이 두 플래그가 모두 1인 경우에만 *coeff_abs_level_remaining* 구문 요소가 부호화 된다. 각 계수의 부호는 1비트 심볼인 *coeff_sign_flag*를 이용해 부호화 한다.

3. 제안하는 방법

3.1 복잡도 모델

문맥 부호화 빈은 이전 빈을 처리하면서 업데이트 한 확률 상태와 범위를 사용하기 때문에 높은 데이터 의존성을 보인다. 즉, 문맥 부호화 빈은 현재 빈의 부/복호화가 모두 수행된 후에 다음 빈의 부/복호화를 수행할 수 있기 때문에 병렬 처리에 어려움이 있다. 또한, 확률 구간을 읽고 현재 상태를 판단하는 데에도 많은 시간이 소요된다.

따라서 제안하는 방법에서는 CABAC의 문맥 부호화 빈으로 부/복호화 되는 구문요소들을 이용해 CABAC의 복잡도를 모델링한다. CABAC의 복호화

복잡도는 정규 부호화 엔진의 동작 수에 비례하고 바이패스 빈은 문맥 부호화 빈에 비해 상대적으로 빈 처리 과정이 단순하므로 제안하는 복잡도 모델에서는 문맥 부호화 빈의 수만을 고려한다.

표 1에서 살펴본 바와 같이 문맥 부호화 빈으로 부호화되는 구문요소는 *last_significant_coeff_x_prefix*, *last_significant_coeff_y_prefix*, *coded_sub_block_flag*, *sig_coeff_flag*, *coeff_abs_level_greater1_flag*, *coeff_abs_level_greater2_flag*이다. 이들 구문요소에 해당하는 문맥 부보화 빈의 수를 각각 x_{last_x}, x_{last_y}, x_{csb}, x_{sig}, x_{gr1}, x_{gr2}라고 할 때, 다음과 같이 CABAC 변환 계수 부/복호화 복잡도 모델을 설계할 수 있다.

$$C = \omega_0 + \omega_1 \cdot x_{last\_x} + \omega_2 \cdot x_{last\_y} + \omega_3 \cdot x_{csb} + \omega_4 \cdot x_{sig} + \omega_5 \cdot x_{gr1} + \omega_6 \cdot x_{gr2} \quad (1)$$

여기서, $\omega_0, \omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_6$은 가중치를 의미한다. 각 항의 가중치를 계산하기 위해 미리 다수의 실험 영상에 대해 부호화를 수행한 후, Intel의 VTune™ performance analyzer[8]를 이용해 CABAC 변환 계수 복호화 복잡도를 측정한다. 다음, 식 (1)의 각 기여 요인 x_{last_x}, x_{last_y}, x_{csb}, x_{sig}, x_{gr1}, x_{gr2}의 수를 측정한다. 마지막으로, 제한 최소자승법(constrained least square method)을 이용해 가중치의 최적값을 찾는다.

제한 최소자승법을 이용해 얻은 가중치의 값은 식 (2)와 같다.

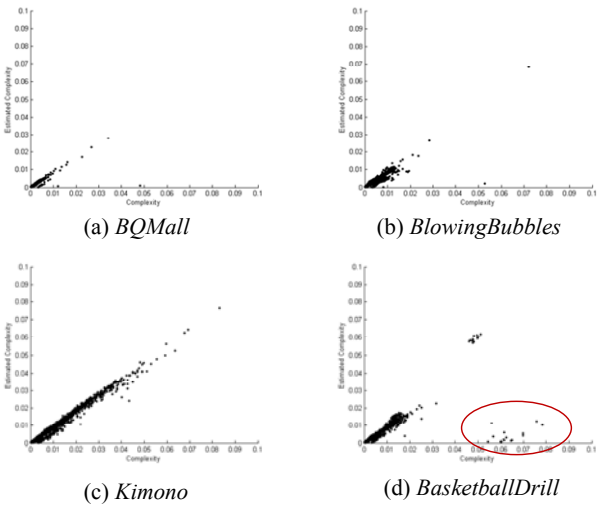
$$\omega_0 = 3.795\times10^{-1},\ \omega_1 = 4.690\times10^{-1},\ \omega_2 = 4.201\times10^{-1},\ \omega_3 = 0,\ \omega_4 = 3.772\times10^{-1},\ \omega_5 = 4.869\times10^{-1},\ \omega_6 = 0 \quad (2)$$

여기서 가중치 ω_3과 ω_6이 0으로 근사되는 것을 알 수 있는데, 이는 관련 구문 요소가 복잡도에 큰 영향을 미치지 않음을 의미한다. 그림 2는 다수의 실험 영상에 대해 제안한 복잡도 모델로 계산한 복호기 복잡도와 실제 CABAC의 복잡도를 비교한 것이다. 그림 2를 살펴보면, 원으로 표시한 부분과 같이 약간의 예측 오차가 있긴 하지만 비교적 실제 복잡도를 잘 예측하는 것을 알 수 있다.

3.2 복잡도를 고려한 율-왜곡 최적화

HEVC는 변환과정을 결정하는 TU(transform Unit)을 정의하며, TU는 재귀적인 트리 구조를 사용하여 여러 개의 하위 블록으로 분할될 수 있다. 이와 같은 트리 구조의 사용으로 TU는 4×4에서 32×32의 다양한 블록 크기를 지원할 수 있다. 변환 블록의 크기는 라그랑지 율-왜곡 최적화 (rate-distortion optimization) 기술을 이용해 계산된다.

율-왜곡 최적화 기술은 압축에 의한 영상의 손실과 영상을 부호화하는데 소모되는 비트를 이용해 비용값을 계산하고, 최소의 비용값을 갖는 후보를 사용해 부호화하도록 하는 방법이다. 비용값을 연산하기 위해 식 (3)을 사용한다.



(a) *BQMall* (b) *BlowingBubbles*

(c) *Kimono* (d) *BasketballDrill*

그림 2. 제안한 복잡도 모델의 정확도 평가

$$J = D + \lambda \cdot R \quad (3)$$

여기서, D는 원본 이미지와 복원한 이미지의 왜곡 정도이며, R은 비트수, λ는 라그랑지 승수이다.

율-왜곡 최적화를 이용한 가변적인 TU 분할 기술은 높은 부호화 효율을 얻을 수 있는 핵심 기술 중 하나이다. 하지만, 필연적으로 부호화 복잡도가 증가하게 되고 이러한 부호화 복잡도의 증가는 HEVC의 단점 중 하나이다.

제안하는 방법에서는 율-왜곡 최적화 단계에서 영상 왜곡과 비트수뿐만 아닌 복잡도도 함께 고려하기 위해 앞에서 설계한 복잡도 모델을 비용값 계산식에 추가했다.

$$J = D + \lambda \cdot R + \lambda \cdot C \quad (4)$$

여기서, D는 앞에서 말한 것과 같이 는 원본 이미지와 복원한 이미지의 왜곡 정도이며, R은 비트수, λ는 라그랑지 승수, C는 앞에서 제안한 복잡도 모델이다.

4. 실험 결과

제안하는 높은 처리량을 갖는 HEVC CABAC 변환 계수 부호화 방법을 HM 8.0에 구현한 후, JCT-VC에서 제공하는 영상을 이용해 실험을 수행했다. 실험 조건은 HEVC 표준화에서 사용한 공통 실험 조건 중 화면내 메인 실험 조건을 사용했다.

복잡도 성능 평가를 위해 변환 계수 부호화에서 정규 부호화 모드로 부호화되는 문맥 부호화 빈의 수를 각각 측정하였다. 부호화 성능 평가를 위해서는 BD-rate(Bontegaard deltra bit-rate)를 이용해 비트율 변화를 계산했다. 이 실험에서는 HTB 모드를 사용하지 않은 HEVC 코덱을 기준으로 하고, HTB 모드를 사용했을 때의 성능 및 제안한 방법의 성능을 각각 측정하였으며, 그 결과는 표 2와 같다.

표 2. 제안한 알고리즘의 복잡도 감소량 및 부호화 성능 비교 (기준: HEVC (HTB = off))

영상	QP	HEVC (HTB = on)		HEVC (HTB = on) + 제안하는 방법	
		Δ문맥 부호화 빈의 수 (%)	BD-rate (%)	Δ문맥 부호화 빈의 수 (%)	BD-rate (%)
PeopleOnStreet (2560×1600)	22	-11.1	+0.1	-12.8	+0.6
	27	-8.5		-11.8	
	32	-7.1		-12.1	
	37	-6.2		-12.5	
Cactus (1920×1080)	22	-7.8	0	-12.2	+0.6
	27	-7.9		-10.5	
	32	-5.4		-9.9	
	37	-3.0		-9.7	
Kimono (1920×1080)	22	-15.9	-0.1	-17.2	+0.3
	27	-16.1		-17.1	
	32	-13.8		-15.3	
	37	-9.8		-11.6	
BQMall (832×480)	22	-12.4	-0.1	-14.5	+0.4
	27	-9.1		-11.8	
	32	-4.6		-9.5	
	37	-2.0		-9.6	
BasketballDrill (832×480)	22	-7.6	0	-10.8	+0.7
	27	-5.2		-10.3	
	32	-7.2		-11.7	
	37	-1.6		-5.8	
BlowingBubbles (416×240)	22	-8.1	-0.2	-10.9	+0.4
	27	-6.6		-9.8	
	32	-1.9		-7.6	
	37	-2.4		-12.2	
평균		**-7.6**	**-0.1**	**-11.6**	**+0.5**

표 2의 실험 결과로부터, 제안한 방법과 기존의 HTB 방법을 함께 사용하는 경우, HTB를 사용하지 않은 HEVC 변환 계수 부호화 방법에 비해 평균 11.6%의 문맥 부호화 빈의 수를 감소시키는 것을 알 수 있었다. 또한, 제안한 방법은 HTB에 추가로 평균 4%의 문맥 부호화 빈의 수를 감소시켰다. 제안한 방법을 사용함으로써 발생하는 부호화 성능 저하는 평균 0.5%로 크지 않다.

5. 결론

본 논문에서는 HEVC CABAC 변환 계수 부호화에서 높은 복잡도의 요인으로 알려진 문맥 부호화 빈의 통계적 특성을 이용해 새로운 복잡도 모델을 제안하였다. 또한, HEVC 부호기의 율-왜곡 최적화 과정에서 제안한 복잡도 모델을 이용함으로써 CABAC 변환 계수 복호화의 처리량을 향상시키는 방법을 제안했다. 실험을 통해 제안한 방법이 기존의 변환 계수 복호화 방법에 비해 큰 부호화 성능 저하 없이 평균 약 11.6%의 문맥 부호화 빈의 수를 감소시키는 것을 확인했다.

감사의 글

이 논문은 2013년도 한국연구재단의 지원을 받아 연구되었음(NRF-2013K2A2A2000661).

참고문헌

[1] B. Bross, W. Han, J. Ohm, G. Sullivan, and T. Wiegand, “High Efficiency Video Coding (HEVC) Text Specification Draft 8,” *JCTVC-J1003*, July 2012.

[2] 호요성, 최정아, UHD 고화질 영상 압축 기술: HEVC 알고리즘 이해와 프로그램 분석, *진샘미디어*, 2013.

[3] T. Nguyen, D. Marpe, M. Siekmann, and T. Wiegand, “Non-CE1: High Throughput Coding Scheme with Rice Binarization,” *JCTVC-H0458*, Feb. 2012.

[4] S. Kim, K. Misra, L. Kerofsky, and A. Segall, “Non-CE1: High Throughput Binarization (HTB) Method with Modified Level Coding,” *JCTVC-H0510*, Feb. 2012.

[5] J. Lainema, K. Ugur, and A. Hallapuro, “CE1.D1: Nokia Report on High Throughput Binarization,” *JCTVC-H0232*, Feb. 2012.

[6] J. Chen, W. Chien, R. Joshi, J. Sole, and M. Karczewicz, “Non-CE1: Throughput Improvement on CABAC Coefficients Level Coding,” *JCTVC-H0554*, Feb. 2012.

[7] F. Pescador, M. Chavarrias, M. Garrido, E. Juarez, and C. Sanz, “Complexity Analysis of an HEVC Decoder based on a Digital Signal Processor,” *IEEE Transactions on Consumer Electronics*, vol. 59, no. 2, pp. 391-399, May 2013.

[8] J. Reinders, VTune™ Performance Analyzer Essentials: Measurement and Tuning Techniques for Software Developers, *Intel Press*, 2005.