공학박사 학위논문

Full RDO를 사용하는 HEVC 하드웨어를 위한 Rate Control 알고리듬의 개선과 구현

An Improved Rate Control Algorithm and Its Implementation for HEVC Encoders using Full Rate-Distortion Optimization

2015 년 2 월

서울대학교 대학원

전기·컴퓨터 공학부

최 순 우

초 록

HM 인코더에서 적용된 coding tree unit (CTU) 수준의 rate control을 적용하면, rate control을 적용하지 않았을 경우에 비해서 코딩 효율이 나빠져 Bjøntegaard-delta rate (BD rate)가 약 4.14 % 증가한다. 그리고 HM 인코더에서는 rate control 알고리듬이 floating point로 구현되어 있어 HW 구현에 적합하지 않다. 그래서 이 논문은 HEVC의 reference SW인 HM 인코더에 적용되어 있는 rate control 알고리듬의 코딩 효율을 개선한 내용과, HW 구현에 적합하게 수정하고 내용을 설명한 후에, 수정된 rate control 알고리듬의 HW 구현에 대해서 기술한다. 이 논문의 기여는 picture 수준의 bit 할당 방법 개선, HW 구현에 적합한 full RD cost의 사용, log를 취한 log R-log λ model의 도입, 그리고 개선한 rate control 알고리듬의 HW 구현이다.

HM 인코더의 rate control에서 picture 수준의 bit 할당은 이미지 시퀀스에 따라서 이미지 후반부에 bit rate이 부족하여 picture의 peak signal-to-noise ratio (PSNR)이 급격히 떨어지는 현상을 보인다. 이 현상을 완화하기 위하여 전체 이미지 시퀀스에서 target bit 할당을 이미지 초반부에 bit을 조금 덜 할당하여 이미지 시퀀스 후반부에 좀 더 bit을 할당하여 이미지 시퀀스 후반에 PSNR이 떨어지는 현상을 완화시키도록 picture 수준 bit 할당을 위한 수정된 알고리듬을 제안한다.

그리고 transform & full RDO & reconstruction을 위한 pipeline stage에서 full RD cost를 이용한 rate distortion optimization (RDO)을 사용한다고 가정한다. 이 pipeline stage에서 full RD cost 계산하는 HW 구현을 위하여 두 가지 기법을 사용했다. 첫째로 rate

control의 코딩 효율을 높이기 위해서, CTU별 λ가 아닌 picture의 평균 λ를 이용하여 인코딩을 수행하였다. 둘째로 full RD cost 계산의 HW 복잡도를 줄이기 위해서 quantization step size (Q_{step})의 제곱으로 나눈 normalized full RD cost를 사용하여 full RD cost의 dynamic range를 크게 줄였다.

HM 인코더에서 rate control의 R-λ model은 floating point로 구현이 되어 있고 지수 연산을 이용하기 때문에 HW 구현에 적합하지 않다. 그래서 R-λ model을, 선형 연산을 이용할 수 있고 HW 구현에 적합하도록, log를 취하여 log R-log λ model로 변형하였다. HM 인코더에서 사용하는 R-D model인 hyperbolic model의 parameter update할 때 log를 취한 model parameter의 update의 근사이기 때문에 log R-log λ model을 이용하였을 때 코딩 효율이 오히려 조금 좋아졌다. 그리고 rate과 관련된 변수들의 log domain과 real domain에서의 값 변환을 위해서 look-up table (LUT)을 이용한 $\log_2$와 anti-$\log_2$를 구현하였다. 또한 나눗셈 연산도 LUT을 이용하여 HW의 복잡도를 줄여 구현하였다.

제안하는 rate control 방법의 효용성을 5개의 1080p 이미지 시퀀스 Kimono, ParkScene, Cactus, BasketballDrive, BQTerrace에 대하여 인코딩 결과로 판단했다. 인코딩 환경은 common test condition의 random access (RA) configuration으로 TU split을 지원하지 않도록 하여 maximum TU depth를 1로 설정하였다. Rate control의 target rate은 rate control을 사용하지 않고 QP 22, 27, 32, 37로 인코딩한 경우에 발생한 rate들로 정하였다. 이 조건에서 개선한 rate control 알고리듬은 HM 인코더에 적용된 CTU-level rate control의 Y-BD rate 4.14 %를 1.99 %로 감소시킨다. 그리고 후반에

PSNR이 떨어지는 현상을 줄여서 minimum PSNR을 평균 0.11 dB 향상 시켰고 특히 ParkScene 이미지 시퀀스에서는 최대 1.58 dB까지 향상시켰다. 제안한 rate control algorithm을 HW로 GOP, picture, CTU level을 모두 지원하도록 구현했는데, 그 전체 복잡도는 27.5 kgate이고 추가로 32 KB의 메모리가 필요하다. Rate control의 수행에 필요한 cycle budget은 CTU당 4 cycle로 4K 30 fps를 400 MHz에 수행한다고 하였을 경우에 0.06 %의 overhead에 해당하며 전체 인코딩 과정의 영향을 거의 주지 않는 수준이다.

주요어 : high efficiency video coding (HEVC), rate control, hardware implementation, full rate distortion (RD) cost
학 번 : 2007-23058

목 차

표 목차

그림 목차

제 1 장 서 론

High efficiency video coding (HEVC) 표준 [1]은 기존의 H.264/AVC 표준에 비해서 같은 화질에 2배의 압축률을 얻는 것을 목표로 표준화 작업이 수행되어 2013년 4월에 final draft international standard (FDIS)가 확정되었다. HEVC는 H.264에 비해서 코딩 효율을 높이기 위해서 지원하는 normative 이슈인 코딩 툴을 개선하는 것 뿐만 아니라 rate-distortion optimization (RDO)나 rate control과 같은 non-normative인 인코딩 방법의 개선도 이루었다.

HEVC 표준은 H.264 표준이 16x16 macroblock 크기만을 지원한 것에 비해 64x64, 32x32, 16x16의 다양한 coding tree unit (CTU) 크기를 지원하고 64x64, 32x32, 16x16, 8x8의 coding unit (CU) 크기를 지원한다. 그리고 H.264 표준에서는 8x8, 4x4 transform만을 지원하였지만 HEVC 표준에서는 transform 크기를 32x32, 16x16, 8x8, 4x4를 지원한다. 이는 코딩 효율을 향상시켰지만 기존의 비디오 표준에 비해서 다양한 모드 중 최적 모드의 탐색을 위해서 더 많은 연산량이 필요하다. 이 탐색 과정에서 최적 모드를 선택하여 코딩 효율을 높이기 위한 효율적인 RDO가 기존 비디오 표준에 비해서 더 중요하게 되었다. 또한 정해진 채널에 데이터를 전송하기 위해 주어진 방법도 HEVC HM 인코더 [2]에서는 H.264/AVC JM 인코더에 비해서 많이 바뀌고 개선되었다.

스마트폰이나 테블릿 등이 보편적으로 사용되면서 모바일 기기를 통하여 실시간 동영상을 보는 일이 많아졌다. 또한 화상 회의와 영상 통화등이 보편화 되어 실시간으로 영상을 인코딩하여 데이터를 전송하는 것이 필요하다. 따라서 정해진 채널에 데이터를 전송하기 위해 정해진

target rate에 맞추어서 고화질로 비디오 인코딩 하는 것이 필요하다. HEVC 표준은 기존 비디오 표준에 비해 같은 화질을 더 낮은 rate으로 전송할 수 있기 때문에 앞으로 모바일 환경에서 널리 이용될 것이다. 스마트폰이나 테블릿 등의 모바일 기기에서 HEVC 인코딩의 많은 연산량을 저전력으로 수행하기 위해서 HEVC 인코더의 HW 구현이 필수적이다. 이 논문에서는 HEVC 표준으로 정해진 target rate에 맞추어서 인코딩할 때 코딩 효율을 높이도록 HM의 rate control 알고리듬을 개선하고 HW 구현에 적합하게 rate control 알고리듬을 수정하여 실시간 4K 30 fps 급의 HW 인코더에 적용할 수 있도록 한다.

1.1 연구의 배경

HEVC 표준의 코딩 효율을 높이기 위해서는 효율적인 RDO가 필요하다. HEVC reference software인 HM 인코더 [2]에서는 RDO 수행의 복잡도를 줄이기 위해서 SAD와 SATD 기반의 low-complexity rate distortion (RD) cost를 이용한 RDO와 SSE기반의 full RD cost를 이용한 RDO를 지원하고 있다. RD cost는 distortion 추정치 D와 rate 추정치 R에 Lagrangian multiplier (λ)를 곱한 값을 더하여, 즉 $J=D+\lambda \cdot R$ 식을 통하여, 계산하고 이 값이 최소가 되는 모드를 선택하거나 움직임 벡터 (motion vector)를 선택한다. Low-complexity RD cost에서는 distortion으로 식 (1-1)과 같이 SAD와 SATD를 사용하며 이는 original pixel과 predicted pixel의 차로 구해진다. low-complexity RD cost의 rate는 PU 의 선택과 관련된 syntax element (SE)들 mode의 rate를 사용하고 residual의 rate를 포함하지 않기 때문에 residual을 transform, quantization, inverse

quantization, inverse transform 그리고 reconstruction을 하는 과정이 필요하지 않다. Full RD cost의 distortion은 식 (1-2)에서와 같이 SSE를 사용하고 이는 original pixel과 reconstructed pixel의 차의 제곱의 합으로 구해진다. 그리고 FRD cost의 rate는 mode와 residual의 rate를 모두 포함한다. 따라서 residual을 transform, quantization, inverse quantization, inverse transform 그리고 reconstruction을 하는 과정을 수행하여야 하고 transform과 quantization이 수행된 계수의 rate를 구한다. 표 1-1은 HM 인코더에서 low-complexity RD cost와 full RD cost를 사용하는 부분을 보여준다. low-complexity RD cost를 이용하여 candidate의 수를 줄인 후 최종적으로 full RD cost를 이용하여 CU, PU, TU를 선택하게 된다.

$$J_{LRD} = SA(T)D + \lambda_{pred} \cdot R_{mode} \quad (1-1)$$

$$J_{FRD} = SSE + \lambda_{mode} \cdot (R_{mode} + R_{residual}) \quad (1-2)$$

$$\lambda_{pred} = \sqrt{\lambda_{mode}} \quad (1-3)$$

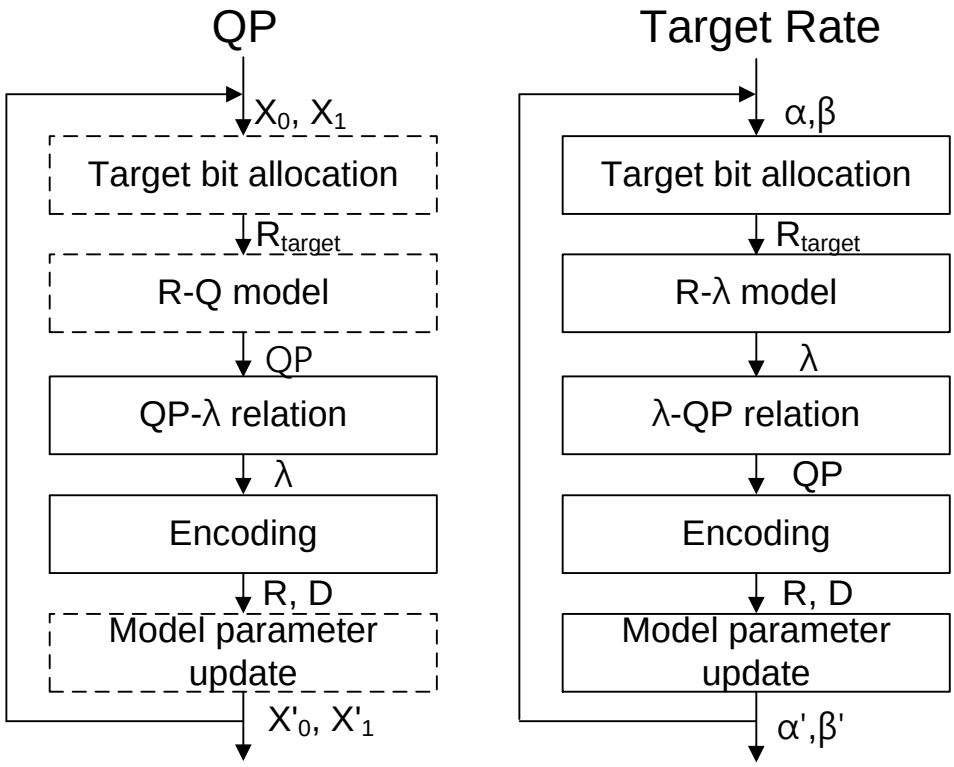
표 1-1 HEVC reference SW인 HM 인코더에서 low-complexity RD cost와 full RD cost를 사용하는 부분

Type	사용
Low-complexity RD cost (SAD/SATD based)	Rough intra mode decision (SATD) Integer ME (SAD) Fractional ME (SATD) Merge excluding 2Nx2N (SATD)
Full RD cost (SSE based)	Fine intra mode decision Skip, 2Nx2N merge CU size decision PU partition decision TU size decision RDOQ decision

비디오 HW 인코더의 구현은 높은 throughput을 만족하기 위해서 기본 unit단위로 pipeline 구조 [3][4]를 이용한다. 그리고 많은 HW 인코더가 HW 복잡도를 낮추기 위해서 표 1-1의 full RD cost를 이용하지 않고 HW 구현 [3][5]-[7]을 하였다. Full RDO를 이용하지 않는 방법은 H.264 표준에서 full RD cost를 이용할 경우와 비교해서 약 15 % [8]의 코딩 효율 저하를 발생 시킨다. Prediction stage에서는, 한 CU의 각 prediction unit (PU) partition의 각 PU 마다, low-complexity RD cost를 이용하여서 motion estimation과 intra prediction을 수행하여 motion vector (MV)를 계산하고, full RD cost로 평가할 intra mode의 후보들을 선택한다. Transform & full RD cost & reconstruction stage에서는 prediction stage의 줄어든 candidate에 대해서 full RD cost를 계산하기 위해서 transform, quantization, full

RD cost estimation을 수행하고 PU partition과 coding unit (CU) 크기의 결정을 수행한다. In-loop filter (ILF) stage에서는 HEVC 표준의 ILF인 deblocking filter와 sample adaptive offset (SAO)를 수행하고 마지막 entropy coding stage에서는 정해진 mode와 계수를 CABAC을 수행하여 bitstream으로 인코딩한다. pipeline 구조로 HW를 구현하며 pipeline 구조로 인하여 reconstruction pixel을 이용할 수 없거나 주위 블록의 결정된 MV를 알지 못하여 발생하는 코딩 효율의 저하가 있다. 이 코딩 효율의 저하를 이 논문의 영역 밖으로 하고 HM 인코더와 같이 코딩 효율의 저하가 없는 것으로 가정하였다.

그림 1-1은 HM 인코더에서 rate control을 on/off인 경우의 인코딩 과정의 비교이다. 그림 1-1 (a)는 rate control을 수행하지 않았을 경우의 인코딩 과정이다. Rate control를 이용하지 않았을 경우를 rate control을 사용하는 경우와 비교하여 생각하면 이미지 시퀀스의 R-Q model의 quadratic model의 X_0, X_1의 model parameter를 알고 target rate에 맞추어서 QP를 정하여서 인코딩 하는 것으로 생각할 수 있다. 그림 1-1 (b)는 HM 인코더의 rate control을 이용하였을 경우의 과정이다. Rate control을 수행하였을 경우는 target rate을 정하고 model parameter α, β를 반영한 R-λ model을 이용하여서 λ값을 결정한다. 그리고 λ값과 QP의 관계를 이용하여서 QP값을 결정한다. 인코딩을 수행하고 인코딩 결과인 발생 rate과 target rate을 비교하여 model parameter를 update한다.



(a) (b)

그림 1-1 (a) rate control을 이용하지 않을 경우의 인코딩 과정
(b) rate control을 이용하였을 경우의 인코딩 과정

표 1-2 HM 인코더의 rate control on/off의 경우 비교

	HM 인코더의 rate control off	HM 인코더의 rate control on
Target rate	Picture특성에 따른 가상의 target R에 대해 QP를 정하고 λ를 정함	Input으로 주어짐
Model parameter	Picture의 QP와 GOP 구조에 따른 QP와 λ관계인 a, W_k	R과 λ관계인 α, β $\lambda = \alpha \cdot R^{\beta}$
$\lambda - Q_{step}$ 관계	$\lambda = (a \cdot W_K \cdot 2^{-8/3}) \cdot Q_{step}^2$	$QP = 4.2005 \cdot \ln\lambda + 13.7122$ ($\lambda = 0.106 \cdot Q_{step}^2$)

표 1-2은 HM 인코더의 rate control on/off의 경우의 target rate 결정, model parameter 사용과 λ와 QP와의 관계를 비교한 것이다. Rate control을 사용하지 않을 경우에는 rate control을 이용할 때 input으로 주어지는 rate 대신에 가상의 target rate에 대해서 QP와 λ가 결정되는 것으로 여길 수 있다. 그리고 rate control을 이용하지 않을 경우에 group of picture (GOP) 구조와 QP를 고려한 model parameter a와 W_{k}는를 이용하여 QP와 λ의 관계를 $a \cdot W_K \cdot 2^{\frac{QP-12}{3}}$ 와 같이 구한다. 반면 rate control을 이용할 경우에는 $QP=4.2005 \cdot \ln \lambda + 13.7122$의 고정된 식을 이용하여 λ로부터 QP를 결정한다. 이를 변형해 보면 rate control을 이용하지 않았을 경우의 $a \cdot W_K \cdot 2^{-8/3}$값을 0.106의 고정된 값을 이용하는 것과 같다.

1.2 관련 연구

Rate control의 목적은 정해진 rate을 전송할 수 있는 채널을 통해서 bitstream을 전송할 경우에 디코더가 채널로부터 bitstream을 받아서 초기 시간 (initial delay)이 지난 후 디코딩을 수행할 때 디코더의 bitstream을 저장하는 coded picture buffer (CPB)가 그림 1-2과 같이 overflow와 underflow가 발생하지 않도록 조절하는 것이다. HEVC 표준은 leaky bucket model을 이용한 hypothetical reference decoder (HRD) [9]를 이용하여서 주어진 profile과 level의 디코더의 버퍼의 underflow와 overflow가 발생하지 않도록 인코딩을 하여야 한다.

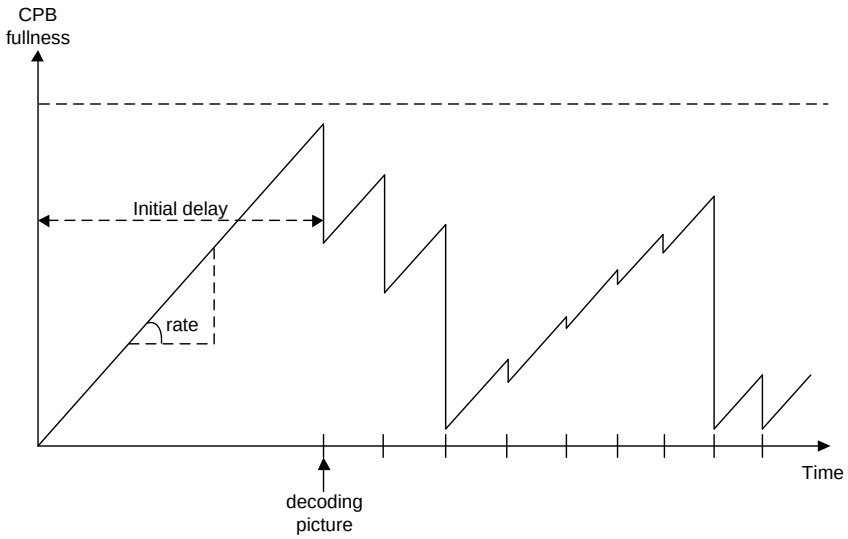


그림 1-2 Constant bit rate (CBR)의 경우 hypothetical reference decoder (HRD) 의 동작

HEVC의 표준화 과정에서 HM 인코더에 처음 적용된 rate control 알고리듬은 unified R-Q model [10]을 이용한 rate control 이다. Rate control을 수행하지 않고 인코딩할 때 RDO 과정에서 모든 CU 크기에서 같은 λ값을 이용하는 것으로부터 pixel별 rate을 기반으로 rate control을 수행한다. 식 (1-4)은 λ값을 유도하는 식이고 식 (1-5)는 모든 CU 크기에서 같은 λ값을 이용하기 때문에 pixel단위 rate과 distortion을 고려하면 HEVC의 다양한 크기의 CU에 대해 같은 R-Q model을 적용할 수 있음을 이용하였다. H.264 표준에서도 많이 이용된 mean absolute error (MAD)값을 이용한 식 (1-6)의 quadratic model [11]을 이용하여 rate으로부터 QP값을 구한다. 식 (1-6)의 X_1, X_2는 model parameter를 나타내고 Q_{step}은 QP값에 대응되는 quantization step size를 나타낸다. QP가 정해지면 λ값은 rate control을 이용하지 않는 경우와 같이 GOP구조를 반영하여 결정한다. 하지만 HM 인코더에 적용되었던 unified R-Q model은 코딩 효율의 저하가 큰 문제점을 가지고 있었다.

$$\frac{\partial J}{\partial R} = \frac{\partial D}{\partial R} + \lambda = 0 \tag{1-4}$$

$$-\lambda = \frac{\partial D_{64\times64}}{\partial R_{64\times64}} = \frac{\partial D_{32\times32}}{\partial R_{32\times32}} = \frac{\partial D_{16\times16}}{\partial R_{16\times16}} = \frac{\partial D/pixels}{\partial R/pixels} \tag{1-5}$$

$$R = X_1 \cdot \frac{MAD}{Q_{step}^2} + X_2 \cdot \frac{MAD}{Q_{step}} \tag{1-6}$$

R-λ model [12]은 R-Q model을 개선하였고 rate으로부터 λ를 결정한다. R-Q model의 Q와 λ의 관계는 R-D의 관계가 exponential 관계를 가지는 것을 가정 [13]으로 유도되었다. 그러나, 그림 1-3에서 보여진 1080p ParkScene의 random access (RA)의 R-D의 관계는 exponential 관계가 아닌 hyperbolic 관계를 가지고 있다. 그래서, 식 (1-7)에서 표현된 R-D의 hyperbolic 관계로부터 유도된 식 (1-8)는 유도된 R과 λ의 관계를 보여준다. R로부터 λ값을 계산한 후 multiple QP 인코딩을 이용하여 유도한 λ값과 QP의 관계인 식 (1-9)을 통해서 QP값을 결정하고 인코딩을 수행한다. HM 인코더의 rate control은 2장에서 자세히 다룬다.

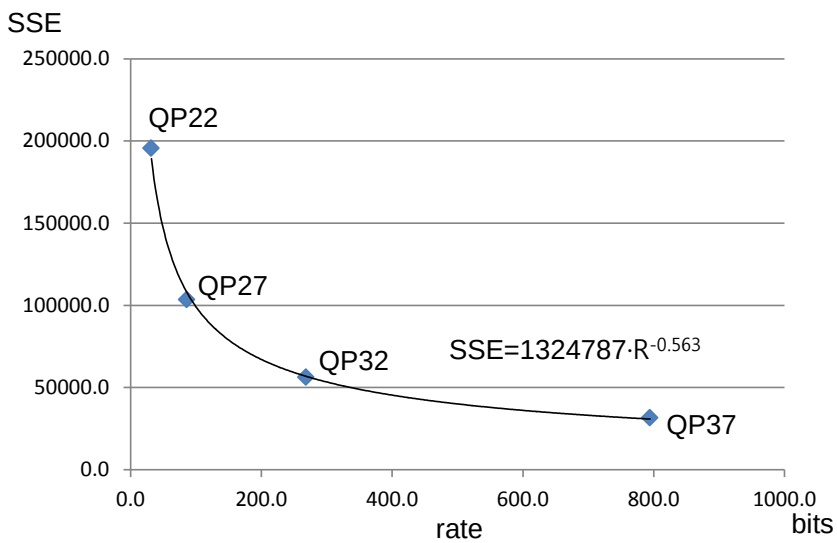


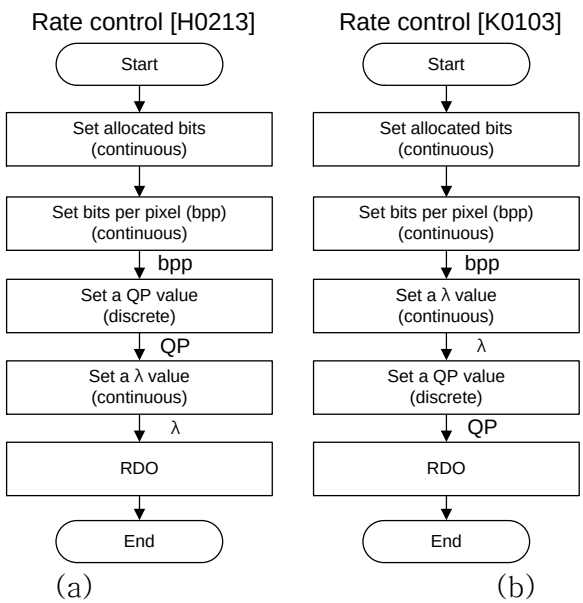
그림 1-3 1080p ParkScene의 random access (RA)의 R-D 관계

$$D = CR^{-K} \quad (1-7)$$

$$\lambda = -\frac{\partial D}{\partial R} = CK \cdot R^{-K-1} = \alpha \cdot R^{\beta} \quad (1-8)$$

$$QP = 4.2005 \cdot \ln \lambda + 13.71222 \quad (1-9)$$

R-λ model [12]은 HM 7.0 인코더에 적용되어 HM 9.0 인코더까지 사용된 unified R-Q model 보다 10 % ~ 40 % 정도까지 코딩 효율이 좋고 rate error도 작아 HM 10.0 인코더부터 채택되었다. 그림 1-4는 R-λ model과 R-Q model의 rate control에 따른 QP와 λ의 결정 순서를 보여준다. Rate와 λ값은 실수값으로 연속적인 값을 가지고 QP는 정수값으로 불연속적인 값을 가진다. QP값을 결정하고 λ값을 결정하면 λ값도 불연속한 값을 가지게 되지만 λ를 먼저 결정하면 λ값은 연속적인 값을 가질 수 있다. R-λ model을 이용한 방법은 불연속적인 λ값의 사용으로 인한 비효율성이 더 적게 반영될 수 있다.



(a) (b)

그림 1-4 Rate control 방법에 따른 QP와 λ의 결정 순서 (a) unified R-Q model (b) R-λ model

HM 인코더에서 rate control을 수행하였을 경우와 rate control을 수행하지 않았을 경우의 전체 인코딩 시간의 차이는 1% 이내 [14]로 rate control의 연산량은 전체 HM 인코더의 인코딩 복잡도에서 무시할 정도이다. 하지만 실시간 인코딩을 만족시키는 고해상도 HW 인코더에서는 대략 수천 cycle 이내에 하나의 64x64 크기의 CTU를 처리하여야 하기 때문에 rate control 알고리듬을 SW로 수행하였을 경우의 수행 cycle을 무시할 수 없다. [15] 논문에서는 H.264의 reference SW인 JM 인코더에 적용된 rate control [17]의 SW의 연산량과 HW 복잡도를 분석하였다. H.264의 reference SW 구현된 rate control [17]를 임베디드 시스템에서 주로 사용하는 RISC 프로세서인 ARM에서 SW로 수행하였을 경우 16x16 크기인 macroblock 당 35k cycle의 수행 cycle을 가지고 HW로 구현[18] 하였을 경우에 144k gate의 복잡도를 가진다. H.264에 적용된 rate control은 이전 인코딩된 정보들을 선형 회귀를 수행하여 model parameter를 계산하기 때문에 곱셈과 나눗셈 연산을 많이 수행하여 복잡도가 높다. [15] 논문에서는 rate control의 복잡도를 줄이기 위해서 JM 인코더의 rate control 알고리듬을 간략히 수정하고 frame-level의 이상의 rate control은 SW로 수행하고 macroblock-level의 rate control은 HW로 수행하도록 SW/HW 분할을 하였다. [16] 논문에서는 HW 구현에 적합하게 rate control 알고리듬을 LUT를 이용하여 floating point로 이용하는 변수들을 fixed point로 수정하였다. 이 논문에서도 $R-\lambda$ model에서 사용되는 log, anti-log, division 연산의 HW 복잡도를 줄이기 위해서 LUT을 이용하였다.

HEVC에 적용된 $R-\lambda$ model을 이용한 rate control은 기존의 rate control 방법들에 비해서 코딩 효율이 뛰어나기 때문에 HW

구현에도 적용할 필요성이 있다. 하지만 아직까지 최대한 알아본 바로는 HM 인코더에 적용된 R-λ model을 HW로 구현하여 적용한 보고는 없었다. 이 논문에서는 R-λ model의 코딩 효율을 개선하도록 알고리듬을 변경한다. 그리고 HW 구현에 적합하게 rate control 알고리듬을 변경하고 HW 구현을 하였다. 그리고 구현한 rate control HW의 수행으로 인한 overhead를 가정한 HW 인코더에서 보인다.

1.3 전체 논문의 구성

이 후 논문의 구성은 다음과 같다. 2장에서는 HEVC HW 인코더의 가정하는 pipeline 구조를 설명하고 pipeline 구조로 인한 코딩 효율의 저하를 기술한다. 3장에서는 HEVC HM 인코더에 적용되어 있는 CTU-level rate control을 설명한다. 4장에서는 HM 인코더에 적용된 CTU-level rate control의 bit 할당 방법의 이미지 시퀀스 후반에 PSNR이 급격이 떨어지는 문제점을 드러내고 PSNR의 저하를 막는 알고리듬을 제안한다. 그리고 CTU-level rate control의 코딩 효율을 높일 수 있는 방법을 적용한다. 또한 RDO과정에서 사용하는 full RD cost의 HW 구현의 복잡도를 줄일 수 있는 full RD cost를 quantization step size (Q_{step})의 제곱으로 normalize한 normalized full RD cost를 설명한다. 5장에서는 rate control 알고리듬을 HW 구현을 수행하기 위해 R-λ model을 HW 구현에 적합하게 log를 취한 log R-log λ model로 수정한다. 그리고 수정한 rate control 알고리듬을 HW로 구현하고 HW 복잡도와 전체 HEVC 인코더에서의 수행 과정을 설명한다. 6장에서는 이 논문의 결론을 정리한다.

제 2 장 HEVC HW 인코더의 pipeline 구성

이 장에서는 가정하는 HEVC HW 인코더의 pipeline 구조를 설명하고 연산량을 줄이기 위해 HM 인코더의 common test condition [19]과 인코딩 configuration 바꾼 것의 코딩 효율의 저하와 HW pipeline 구조로 인하여 dependency를 지키지 못 하여 발생하는 코딩 효율의 저하를 설명한다. 그리고 가정하는 HEVC HW 인코더의 transform & full RDO & reconstruction stage에서의 구현한 HW를 간략히 설명한다.

2.1 가정하는 HEVC HW 인코더의 pipeline 구성

가정하는 HEVC HW 인코더는 4단계 pipeline으로 구성되며 prediction stage, transform & full RDO & reconstruction stage, in-loop filter stage, entropy coding stage로 구성된다. 그림 2-1은 가정하는 HEVC HW 인코더의 구조와 pipeline stage중 prediction stage와 transform & full RDO & reconstruction stage의 구성을 보여준다. Prediction stage와 transform & full RDO & reconstruction stage에서는 CU depth 별로 연산을 병렬적으로 수행하는 것을 가정하였다. 그리고 prediction stage와 transform & full RDO & reconstruction stage의 pipeline은 CTU 단위 [3]가 아닌 CU 크기에 따른 pipeline을 이용한다. 예를 들어, 8x8 CU 와 16x16 CU는 각각 prediction stage와 transform & full RDO & reconstruction stage를 8x8 CU와 16x16 CU의 크기로 각각 pipeline을 수행하여 처리한다.

HEVC는 RDO 과정에서 low-complexity RD cost와 full RD

cost를 이용한다. 그림 2-1에서 low-complexity RD cost를 이용한 RDO는 prediction stage의 intra prediction과 motion estimation과 transform & full RDO & reconstruction stage의 시작 부분에서 실제 merge candidate list와 AMVP candidate list에 대해서 cost를 구할 때 이용된다. Full RD cost는 transform & full RDO & reconstruction stage에서 intra mode 결정, TU partition 결정, PU partition 결정, CU 크기 결정에 이용된다. 논문에서 구현한 rate control은 4개의 pipeline stage 중 full RDO를 수행하는 transform & full RDO & reconstruction stage 에서 이용된다.

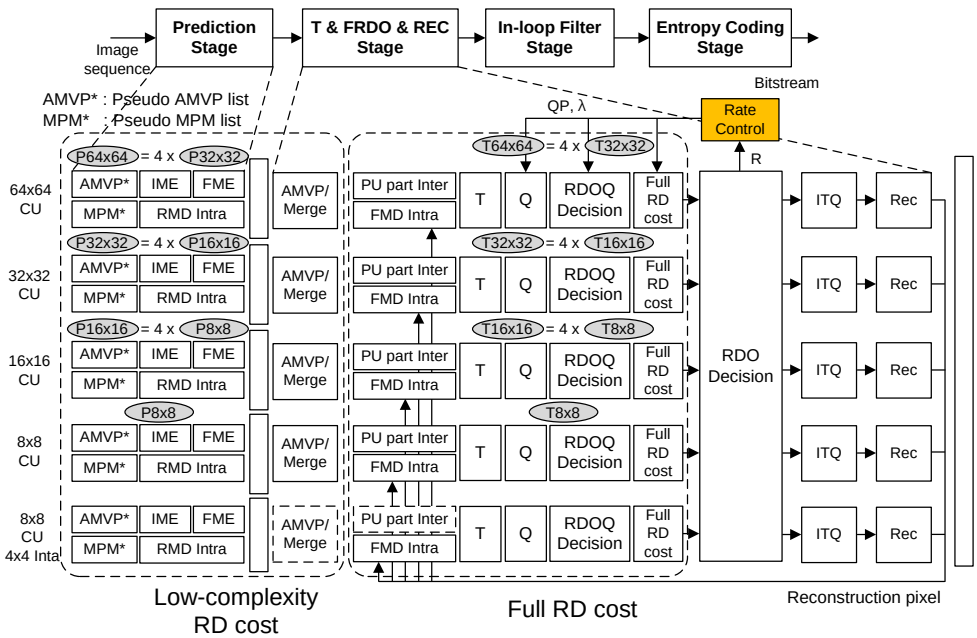
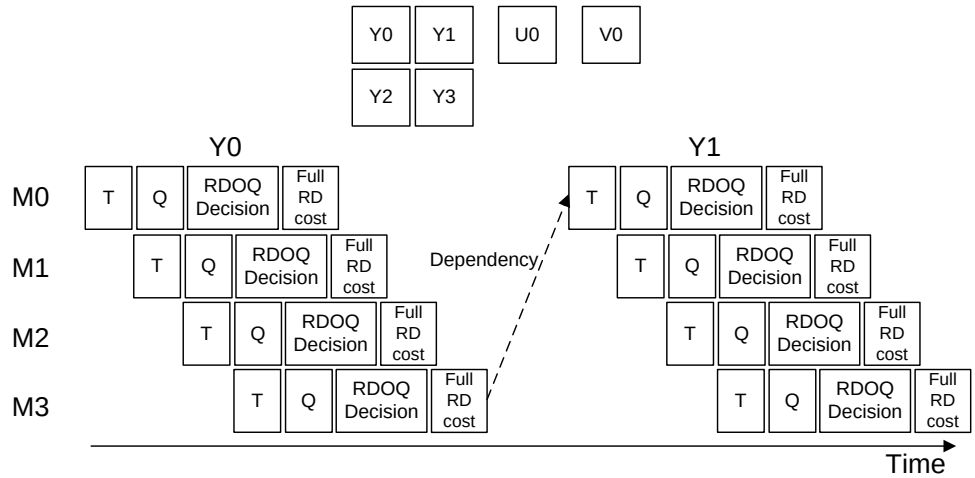


그림 2-1 가정하는 HEVC HW 인코더의 구조

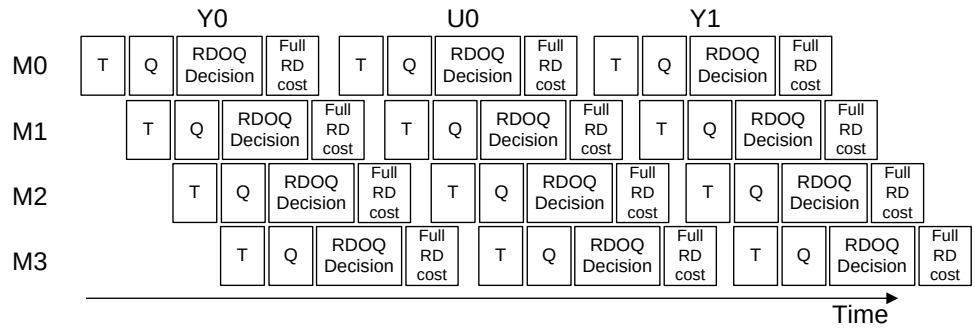
H.264 표준과 HEVC 표준은 코딩 효율을 높이기 위해서 이전에 인코딩된 정보로부터 현재 정보를 prediction하고 차를 전송하여 전송하는 정보의 양을 줄인다. 하지만 HW 구현을 위해서 throughput을 높이기 위해서 dependency를 반영하지 않고 CTU 단위로

pipeline[3][20] 을 수행하게 되면 prediction stage에서 reconstruction이 끝나지 않은 이전의 CTU들의 정보를 이용할 수 없기 때문에 코딩 효율의 저하가 발생한다. 코딩 효율의 저하는 크게 intra prediction에서 주위 block의 reconstruction pixel을 이용하지 못 하여 발생하는 저하와 inter prediction에서 주위 block의 결정된 motion vector (MV)를 이용하지 못 하여 발생하는 저하가 있다.

코딩 효율을 저하시키지 않기 위해 주변 block의 정보를 정확히 이용하기 위해서는 PU 단위로 reconstruction이 끝나기를 기다려야 하기 때문에 prediction stage와 transform & full RDO & reconstruction stage을 병렬적으로 수행하여 HW를 최대한 이용하기 어렵다. Transform & full RDO & reconstruction stage내에서도 같은 CU depth내에서 주변 block의 reconstruction이 완료되기를 기다려야 하고 CU depth간에는 CU 크기 결정으로 인해 synchronization을 수행하여야 하기 때문에 HW의 utilization이 떨어지게 된다. [21] 논문에서는 intra prediction에 대해서 주위 block의 reconstruction pixel이 구해져야 현재 block의 intra prediction을 수행할 수 있는 것을 dependency가 없는 block들의 연산의 scheduling을 통해 HW의 utilization을 높였다. 그림 2-2는 8x8 CU의 4x4 intra PU의 transform & full RDO & reconstruction stage의 mode별 연산과정을 보여준다. 그림 2-2 (a)에서는 reconstruction pixel을 구해야 하는 dependency로 인해서 4x4 PU 간의 bubble이 발생한 것이고 그림 2-2 (b)는 PU간의 dependency가 없는 연산을 삽입시킴으로써 bubble을 줄이는 경우를 보여준다. 이 논문에서는 CTU-level rate control을 주제로 다루기 때문에 CTU 내의 주위 block의 reconstruction pixel이 준비되지 않은 dependency로 인한 bubble을 고려하지 않았다.



(a)



(b)

그림 2-2 8x8 CU의 4x4 Intra의 (a) dependency로 인한 bubble이 생기는 경우 (b) dependency가 없는 연산을 삽입시킴으로써 bubble을 줄이는 경우

제안하는 HEVC HW 구조는 prediction stage와 transform & full RDO & reconstruction stage의 RDO를 수행하는 prediction, transform, full RD cost, reconstruction을 구하는 과정을 CU depth 별로 병렬적인 수행한다. 그리고 CU 크기를 결정할 때 CU depth 간의 synchronization을 수행하여 주변 정보를 정확한 값을 가지도록 하여 코딩 효율의 저하를 막는다. 주변 CU의 정보를 최대한 이용하여 인코딩을 수행하기 위해서 prediction stage와 transform & full RDO &

reconstruction stage에서 CU depth 별로 다른 pipeline granularity를 가져야 한다. 그림 2-3은 prediction stage와 transform & full RDO & reconstruction stage의 CU depth별로의 다른 pipeline의 단위로 수행하는 것을 보여준다. Transform & full RDO & reconstruction stage에서 하위 CU와 상위 CU간의 CU 크기를 결정할 때 미리 연산이 끝난 depth의 CU 크기는 다른 CU 크기의 depth가 완료될 때까지 기다려야 하기 때문에 bubble이 발생한다.

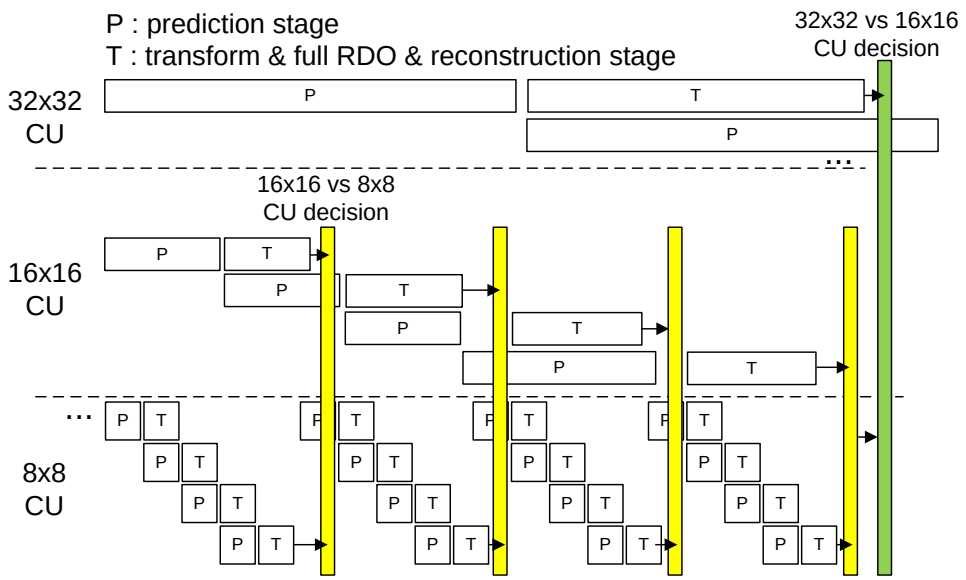


그림 2-3 가정하는 HEVC HW 인코더의 prediction stage와 transform & full RDO & reconstruction stage의 CU 단위의 pipeline 수행

2.2 가정하는 HW 구조의 코딩 효율 저하

가정하는 그림 2-1의 HEVC HW 구조에서는 merge를 transform & full RDO & reconstruction stage의 처음에 정확한 merge candidate list를 구하여 수행한다. Merge를 transform & full RDO & reconstruction stage의 처음에 수행한 이유는 첫째, merge candidate

list는 CU 안의 PU간의 dependency가 존재하지 않는다. 이는 두 개로 나누어진 PU partition 중에 두 번째 PU가 같은 CU내의 PU의 MV를 이용하는 것은 2Nx2N PU와 같기 때문이다. 둘째, merge는 HM 인코더에서 AMVP를 원점으로 이용한 ME에 비해 merge candidate list의 최대 5개의 candidate에 대해서만 cost를 계산하면 되기 때문에 연산량이 작다. 마지막으로 AMVP list는 motion vector predictor값을 잘 못 예측하면 motion vector difference로 인한 코딩 효율의 저하가 발생하지만 merge candidate list는 잘 못 예측하면 실제 merge candidate list를 구하였을 경우에 계산이 안 된 값에 대해서 선택을 할 수 없기 때문에 코딩 효율의 저하가 크다. 따라서 정확한 merge candidate list를 이용하여 merge를 수행할 수 있도록 그림 2-1에서와 같이 transform & full RDO & reconstruction stage의 처음에 수행하도록 하였다. 실제 AMVP candidate list를 구하여 prediction stage에서 구한 MV에 대해서 AMVP에 대한 PU의 low-complexity cost를 구하는 것도 transform & full RDO & reconstruction stage의 처음에 수행하여 inter prediction의 각 PU partition에 대해서 merge인지 AMVP인지와 merge일 경우 merge index를 AMVP일 경우 MV를 결정한다.

HM 인코더에서 ME를 수행할 때 실제 AMVP candidate list 중 cost가 작은 MV를 원점으로 Test Zone Search의 ME 알고리듬을 수행하여 MV를 찾는다. 가정한 HEVC HW 인코더의 ME를 수행하는 prediction stage에서는 정확한 AMVP candidate list를 구할 수 없기 때문에 ME의 시작점을 avail한 위의 CU들의 MV로부터 [22] 논문과 같이 median을 이용하여 구함으로써 표 2-1에서와 같이 약 0.4 %의 BD rate [23]증가가 있다.

표 2-1 Prediction stage에서 avail한 위의 CU들의 MV의 median을 이용하여 ME의 시작점으로 이용하였을 경우의 RA configuration의 BD rate (%)

1080p	BD rate (%)			
	Y	U	V	6:1:1
Kimono	0.15	0.35	0.33	0.18
ParkScene	0.47	0.81	0.32	0.48
Cactus	0.72	0.95	0.71	0.74
BasketballDrive	0.64	1.64	1.04	0.79
BQTerrace	0.08	-0.62	0.23	0.03
Average	0.41	0.63	0.53	0.44

HM 인코더의 intra prediction은 low-complexity RD cost를 이용하는 rough mode decision (RMD)으로 35개의 intra mode중 mode의 수를 줄이고 full RD cost를 이용한 fine mode decision (FMD)로 최종 intra mode를 결정한다. RMD에서는 주위 pixel을 original pixel을 사용하고 FMD에서는 주위 pixel을 reconstruction pixel을 사용하여 full RD cost를 구하여 최종 intra mode를 선택하였다. Intra prediction에서 RMD에서 주위 pixel을 original pixel을 이용하였을 경우에 표 2-2와 같이 0.02 % 정도의 무시할만한 BD rate 증가가 있다. 실제 HW pipeline 구조로 인한 코딩 효율의 저하가 발생하지만 이 논문에서 rate control을 실험할 때는 고려하지 않았다.

표 2-2 Intra prediction의 rough mode decision에서 주위 pixel을 original pixel을 이용하였을 경우의 RA configuration의 BD rate (%)

1080p	BD rate (%)			
	Y	U	V	6:1:1
Kimono	0.01	-0.09	-0.18	-0.01
ParkScene	0.00	-0.13	0.13	-0.01
Cactus	0.01	0.19	0.01	0.02
BasketballDrive	0.10	-0.54	-0.14	0.01
BQTerrace	0.01	0.66	0.76	0.11
Average	0.02	0.02	0.12	0.02

HW 구현시 transform과 full RD cost의 연산량을 줄이기 위해서 표 2-3과 같이 BD rate을 1.0 %정도 증가시키지만 줄일 수 있는 maximum TU depth를 1로 고정하여 TU split search를 수행하지 않도록 인코딩 configuration을 수정하였다. 그리고 natural image에서 표 2-4와 같이 코딩 효율의 저하가 거의 없는 transform skip을 이용하지 않았다. HM 인코더에서 수정한 configuration으로 인코딩하였을 경우를 reference로 두고 rate control 알고리듬을 수정하였을 경우를 실험하였다.

표 2-3 Maximum TU depth = 1로 하여 TU split search를 수행하지 않을 경우의 RA configuration의 BD rate (%)

1080p	BD rate (%)			
	Y	U	V	6:1:1
Kimono	0.86	0.25	0.00	0.75
ParkScene	1.15	0.87	1.04	1.12
Cactus	1.01	0.22	-0.41	0.81
BasketballDrive	0.72	1.11	1.07	0.81
BQTerrace	1.64	1.77	2.90	1.75
Average	1.08	0.84	0.92	1.05

표 2-4 Transform Skip을 이용하지 않았을 경우의 RA configuration의 BD rate (%)

1080p	BD rate (%)			
	Y	U	V	6:1:1
Kimono	-0.15	0.05	0.17	-0.10
ParkScene	-0.03	0.11	0.05	-0.01
Cactus	0.10	0.37	0.11	0.12
BasketballDrive	-0.17	0.41	0.20	-0.07
BQTerrace	0.03	0.52	1.42	0.17
Average	-0.05	0.29	0.39	0.02

2.3 Full RD cost 예측기 HW 구현의 개요

Full RD cost를 이용하여 RDO를 수행하는 transform & full RDO

& reconstruction stage에서 이 논문에서는 다루지 않지만 이미 구현한 rate-distortion optimized quantization (RDOQ) HW와 full RD cost estimator HW의 간략한 설명은 표 2-5와 같다. 4K 30 fps를 400 MHz를 가정하였을 때 실시간 인코딩을 수행하기 위해서 4x4 subblock / cycle의 throughput을 가지는 하나씩을 처리할 수 있도록 구현하였다. 구현한 HW는 transform & full RDO & reconstruction stage에서 CU 크기별 병렬 수행을 지원하기 위해서 병렬로 지원하는 CU 크기만큼의 HW가 필요하여 5개의 depth를 병렬로 처리할 때 약 142만 gate의 HW 복잡도를 가진다. 이 논문에서는 quantization parameter (QP)와 λ결정을 수행할 transform & full RDO & reconstruction stage에서의 rate control의 코딩 효율 개선과 HW 구현을 다룬다.

표 2-5 가정한 HEVC HW 인코더의 transform & full RDO stage의 구현한 RDOQ와 full RD cost estimator의 개요

<table>
<tr><th></th><th>RDOQ</th><th colspan="2">Full RD cost estimator</th></tr>
<tr><td>Throughput</td><td>4x4 subblock / cycle</td><td colspan="2">4x4 subblock / cycle</td></tr>
<tr><td>Pipeline</td><td>6 stages</td><td colspan="2">3 stages</td></tr>
<tr><td rowspan="5">Complexity (gates)</td><td rowspan="5">177.6 K</td><td>mode rate estimator</td><td>6.9 K</td></tr>
<tr><td>residual rate estimator</td><td>53.3 K</td></tr>
<tr><td>distortion estimator</td><td>30.9 K</td></tr>
<tr><td>context memory</td><td>15.7 K</td></tr>
<tr><td>Total</td><td>106.8 K</td></tr>
</table>

제 3 장 HEVC의 CTU-level Rate control의 알고리듬 설명

HM 인코더의 rate control은 HM 7.0 인코더에 처음 적용된 unified R-Q model을 이용한 rate control이 HM 10.0 인코더에서 R-λ model을 이용한 rate control으로 대체된 후, 수 차례 코딩 효율을 높이기 위한 방법들이 제안되어 개선 되었다. I-slice의 rate 계산을 평균 rate에 고정된 비의 bit 할당을 하던 것을 I-slice의 original pixel로부터 계산한 복잡도 [24]를 반영하여 계산하도록 개선되었고 GOP내의 picture들의 bit 할당 방법을 GOP 구조에 따라서 고정된 bit 비율을 이용하여 할당하던 것을 model parameter의 값을 반영 [14]하도록 개선되었다. 이 장에서는 HM 인코더에 적용된 R-λ model의 rate control의 동작을 설명한다.

3.1 HM 인코더의 CTU-level rate control 전체 과정

그림 3-1은 HM 인코더의 R-λ model을 이용한 rate control의 전체 과정을 보여준다. HM 인코더의 입력으로 주어지는 전체 target rate은 초당 kbit인 kbps로 주어진다. 입력 rate에 대해서 HM 인코더의 rate control 알고리듬은 입력으로 주어지는 인코딩하는 이미지 시퀀스 전체에 대해서 target rate을 맞추는 것을 목표로 인코딩을 수행한다. HEVC의 표준화 과정에서 10 초 분량의 다양한 크기와 다양한 frame per second (fps)의 이미지 시퀀스들을 이용되었고 이 이미지 시퀀스들을 입력으로 rate control을 수행하면 total target bits = 10 s x target kbps 을 전체 인코딩 과정에서 맞추는

것을 목표로 인코딩을 수행한다.

Rate control은 GOP level, Picture level 그리고 CTU level로 계층적으로 수행된다. 평균적으로 배분되는 bit과 이전까지 할당하고 남은 bit을 고려하여 현재 target bit 할당을 수행한다. 각 level의 target rate을 결정하고 R-λ model의 model parameter 값을 이용하여서 λ값을 결정한다. 그리고 통계적으로 구한 λ값과 QP의 관계를 이용하여서 QP값을 결정한다. 결정된 λ값과 QP를 이용하여 인코딩을 수행하고 인코딩 결과인 실제 발생 rate과 target rate의 차를 반영하여 model parameter를 update한다.

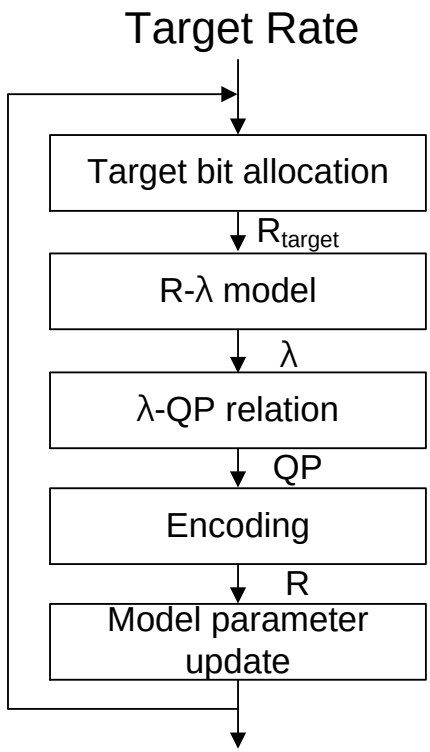


그림 3-1 HM 인코더의 R-λ model을 이용한 CTU-level rate control의 과정

3.2 Target bit allocation

HM 인코더는 GOP의 target rate을 결정할 때 아직 인코딩하지 않은 이미지 들의 정보는 알지 못 하기 때문에 전체 이미지 시퀀스에서 각 picture의 target bit allocation이 최적일 수 없고 rate control을 이용하지 않고 인코딩을 하였을 경우 보다. 코딩 효율을 저하시키는 요인이다.

HM 인코더에서 I-picture의 bit 할당은 [24] 기고서의 방법을 따르고 있다. 이미지 시퀀스의 처음 I-picture는 GOP를 구성하는 picture 수에 포함되지 않고 1 장의 picture로 구성된 GOP로 고려된다. I-picture에 bit 할당을 수행하기 위해서 [24]의 기고서는 picture의 original pixel의 complexity를 반영하는 것을 제안하였다. Complexity는 식 (3-1)과 같이 original pixel을 8x8 Hadamard transform한 matrix의 DC값을 제외한 각 항의 절대값의 합을 이용한다. 식 (3-1)의 h_{ij}는 original pixel을 8x8 Hadamard transform한 값을 나타낸다.

$$C_{Intra} = \sum_{picture} \left(\sum_{i=0}^{7} \sum_{i=0}^{7} |h_{ij}| - |h_{00}| \right) \qquad (3-1)$$

식 (3-1)로부터 구해진 I-picture의 complexity로부터 식 (3-2)의 식을 이용하여서 식 (3-3)의 남은 평균 picture의 rate에서 I-picture의 complexity를 반영하여서 I-picture의 target rate을 결정한다. HM 인코더에서는 기본적으로 a=0.25, b=0.5582값을 이용하고 평균 rate이 pixel 수에 비해서 클 경우에는 식 (3-4)와 같이 a=0.3을 이용한다.

$$R_{I-Slice} = a \cdot \left(\frac{4.0 \cdot C_{Intra}}{R_{Avg}^{Pic}} \right)^{b} \cdot R_{Avg}^{Pic} \quad (3-2)$$

$$R_{avg}^{Pic} = \frac{R_{Left}}{N_{Left}^{Pic}} \quad (3-3)$$

$$a = \begin{cases} 0.25 & 40 \cdot R_{avg}^{Pic} < \#of\ pixels \\ 0.3 & else \end{cases} \quad (3-4)$$

이미지 시퀀스는 I-picture 이 후에 B-picture로 구성된 GOP들이 인코딩된다. HM 인코더의 rate control은 식 (3-5)와 같이 GOP의 크기 N_{GOP}와 GOP 내의 picture의 평균 rate인 R_{Avg}^{Pic} 을 곱하여서 GOP의 target rate을 결정한다.

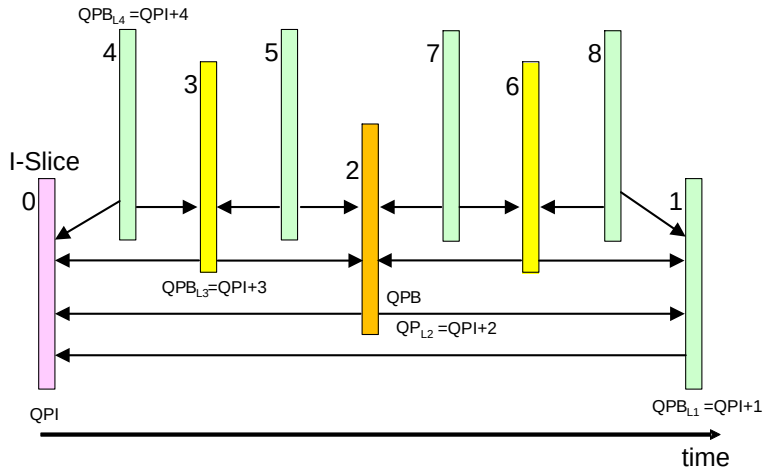
$$R_{GOP} = R_{Avg}^{Pic} \cdot N_{GOP} \quad (3-5)$$

GOP의 target bit을 구하기 위한 picture들의 평균 target bit은 smoothing window (SW)의 개념을 이용하여서 남은 bit을 반영하여 구한다. HM 인코더에서는 GOP의 target bit을 결정할 때는 smoothing window 크기 SW를 40을 이용하여 남은 picture의 수가 40보다 클 경우와 작을 경우를 구분한다. 식 (3-6)은 picture의 평균 rate을 구하는 식으로 남은 picture의 수가 SW보다 클 경우는 전체 sequence의 picture의 평균인 rate인 $\frac{R_{Total}}{N_{Total}^{Pic}}$ 에 sequence에 남은 rate인 R_{Left} 와 $N_{Left}^{Pic} \cdot \frac{R_{Total}}{N_{Total}^{Pic}}$ 의 차이에 $\frac{1}{SW}$ 을 곱해준 값을 더해서 평균 picture 별 rate을 구한다. 남은 picture의 수가 smoothing window 크기 SW보다 작거나 같을 경우에는 이미지 시퀀스가 끝날 때 최종

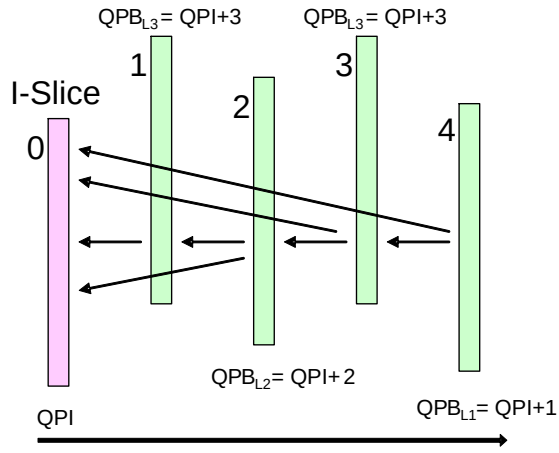
target rate을 최대한 맞추기 위해서 남은 rate을 남은 picture 수로 나누어서 picture의 평균 target rate을 구한다.

$$R_{Avg}^{Pic} = \begin{cases} \frac{R_{Total}}{N_{Total}^{Pic}} + \frac{1}{SW} \cdot (R_{Left} - N_{Left}^{Pic} \cdot \frac{R_{Total}}{N_{Total}^{Pic}}) & N_{Left} > SW \\ R_{Avg}^{Pic} = \frac{R_{Left}}{N_{Left}^{Pic}} & else \end{cases} \quad (3-6)$$

HEVC 표준화 과정에서 all intra (AI) configuration, randomaccess (RA) configuration, lowdelay B (LB) configuration 이 이용되어 다양한 GOP 구조에 대해서 코딩 효율을 테스트하였다. AI configuration은 이미지 시퀀스를 모두 I-picture로 구성하는 것으로 rate control을 이용하지 않을 경우에 모든 picture가 같은 QP값을 가지고 인코딩된다. 그림 3-2는 RA configuration과 LB configuration의 GOP 구조를 보여준다. 그림 3-2 (a)의 RA configuration은 크기가 8인 GOP 크기를 가지고 32 picture마다 I-picture를 삽입한다. RA configuration에서는 picture의 디스플레이 순서와 인코딩 순서가 같지 않으며 계층적인 B-picture의 구조를 가져서 가장 높은 계층의 B-picture는 I-picture의 QP값 보다 4가 큰 QP값을 가진다. 그림 3-2 (b)의 LB configuration은 크기가 4인 GOP를 가지고 시퀀스 처음 picture만 I-picture로 인코딩한다. Picture의 디스플레이 순서와 인코딩 순서가 같으며 계층적으로 QP값을 allocation하여 가장 높은 계층의 B-picture는 I-picture의 QP값 보다 3이 큰 QP값을 가진다.



(a)



(b)

그림 3-2 GOP 구조 (a) RA configuration (b) LB configuration

HM 인코더는 rate control을 수행하여 인코딩할 때에도 GOP 구조를 입력으로 받는다. RA configuration과 LB configuration으로 인코딩할 경우 계층적 picture 구조를 반영하여서 GOP의 picture별 target bit 할당을 수행한다. 이미지 시퀀스의 처음 GOP에 대해서는 이전에 인코딩된 GOP의 정보가 없기 때문에 미리 구해 놓은 표 3-1의 picture별 weight를 이용하여 bits per pixel (bpp)에 따라서 picture 별 bit 할당을 수행하고 두 번째 GOP 부터는 이전 GOP의 인코딩

정보를 반영한 model parameter를 이용하여 picture 별 adaptive ratio bit allocation을 수행한다.

표 3-1 HM 인코더에서 hierarchical bit allocation을 이용하였을 때, RA configuration의 picture의 처음 GOP의 picture 별 bit allocation을 위한 ω_{pic} 결정

Picture level	GOP의 bit per pixel (bpp)			
	(0,0.05]	(0.05,0.1]	(0.1,0.2]	(0.2,+∞)
GPB	30	25	20	15
GPB+1	8	7	6	5
GPB+2	4	4	4	4
GPB+3	1	1	1	1

그림 3-3은 adaptive ratio bit allocation을 수행할 경우에 GOP의 picture 별 bit 할당을 수행하는 과정을 보여준다. 이전 GOP의 picture level 별 model parameter 값과 이전 GOP의 I-picture의 QP값보다 1이 큰 QP값을 가지는 picture level의 λ값인 λ_{Last}에 따른 표 3-2의 λ_{ratio}값을 이용한다. λ_{Last}값이 90보다 큰지 같거나 작은지 따라서 picture level별 λ_{ratio}값을 구분하여 이용한다. Picture level별 model parameter α, β와 λ_{ratio}값으로부터 식 (3-7) (3-8)의 coefficients A, B를 계산하고 식 (3-9)과 같이 picture별 weight를 계산한다. 식 (3-7) (3-8) (3-9)의 i는 GOP내의 picture의 index를 나타낸다. R-λ model에서 구해지는 rate과 관련된 변수는 실제 bit의 수가 아닌 pixel별 rate인 bpp이기 때문에 target rate을 구할 경우에는 pixel 수를 곱하여 실제 rate으로 바꾸어 주어야 한다. 식 (3-9)의 λ_{basic}값은 식 (3-10)를 만족하는 λ_{basic}값을 iteration을 수행하여 범위를 줄여 나가는 방법을 이용하여 값을 찾는다.

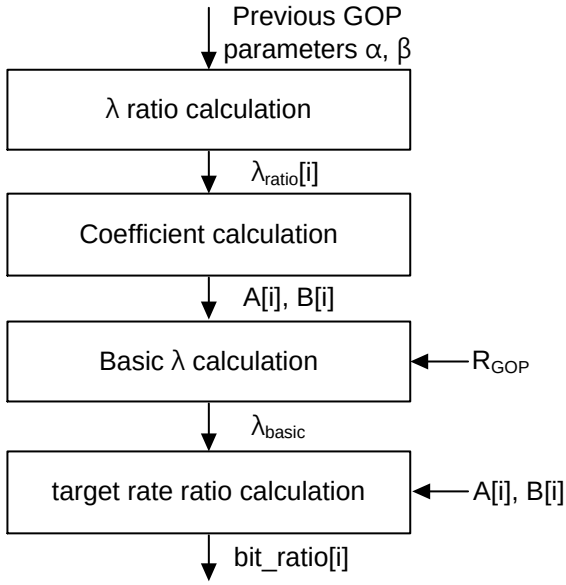


그림 3-3 HM 인코더의 picture-level의 adaptive ratio bit allocation 과정

표 3-2 RA configuration에서 picture-level의 adaptive ratio bit allocation에서 이용되는 λ_{ratio}의 값

	$\lambda_{Last} < 90$	Else
GPB	1	1
GPB+1	$0.725 \cdot \log(\lambda_{Last}) + 0.7963$	4
GPB+2	$1.3 \cdot (0.725 \cdot \log(\lambda_{Last}) + 0.7963)$	5
GPB+3	$3.25 \cdot (0.725 \cdot \log(\lambda_{Last}) + 0.7963)$	12.3

$$A[i] = \left(\frac{\lambda_{ratio}[i]}{\alpha[i]}\right)^{\frac{1}{\beta[i]}} \quad (3-7)$$

$$B[i] = \frac{1}{\beta[i]} \quad (3-8)$$

$$\omega[i] = A[i] \cdot (\lambda_{basic})^{B[i]} \quad (3-9)$$

$$bpp_{target} \approx \frac{1}{GOPsize} \cdot \sum_{GOP} \omega[i] \quad (3-10)$$

GOP 시작 시에 GOP를 구성하는 B-picture들의 target rate을 각

picture의 계산한 weight를 고려하여서 식 (3-11)과 같이 각 초기 target rate $R_{Init}^{Pic}[i]$ 을 결정하여 준다.

$$R_{Init}^{Pic}[i] = R_{GOP} \cdot \frac{\omega[i]}{\sum_{GOP} \omega[i]} \tag{3-11}$$

Picture를 인코딩해 나감에 따라서 초기 target rate에 남은 rate을 반영한 picture의 target rate을 반영하여 식 (3-14)과 같이 새로 picture의 target rate을 계산한다. 식 (3-14)에서 사용되는 남은 rate을 반영한 picture의 rate은 식 (3-12)와 식 (3-13)으로 계산된다. HM 인코더에서는 picture의 target rate을 update하는 식 (3-14)에서 SW 크기를 10을 이용한다. 이는 GOP의 target rate을 구할 때 사용하는 SW 크기인 40보다 작은 값이다.

$$R_{Left}^{Pic}[i] = R_{Left} \cdot \frac{\omega[i]}{\sum_{Left} \omega[i]} \tag{3-12}$$

$$R_{Left} = R_{GOP} - R_{Coded} \tag{3-13}$$

$$R_{Target}^{Pic}[i] = (1 - \frac{1}{SW}) \cdot R_{Init}^{Pic}[i] + \frac{1}{SW} \cdot R_{Left}^{Pic}[i] \tag{3-14}$$

CTU-level rate control을 이용하여 인코딩을 할 때 picture별 할당된 bit을 picture를 구성하는 CTU별로 다시 할당한다. Picture를 구성하는 CTU의 bit 할당을 위해서 GOP에서 picture별 bit 할당을 할 때와 마찬가지로 CTU별 weight를 우선 계산한다. CTU별 weight를 계산은 picture의 평균 λ값과 CTU별 model parameter값을 이용하여서 R-λ model을 이용한 식 (3-15)으로 CTU의 weight를

계산한다. 식 (3-15)의 i는 picture를 구성하는 CTU를 가리키는 index이다. 계산된 CTU 별 weight를 이용하여서 picture를 구성하는 CTU의 rate이 picture의 target rate이 되도록 식 (3-16)을 이용하여 CTU별 초기 target rate을 계산한다.

$$\omega[i] = (\frac{\lambda_{Pic}}{\alpha[i]})^{\frac{1}{\beta[i]}} \tag{3-15}$$

$$R_{Est,Left}^{CTU}[i] = R_{Pic} \cdot \frac{\omega[i]}{\sum \omega[i]} \tag{3-16}$$

CTU를 인코딩해 나감에 따라서 초기 target rate에 남은 rate을 반영한 CTU의 target rate을 반영하여 식 (3-17)과 같이 새로 CTU의 target rate을 계산한다. HM 인코더에서는 CTU의 target rate을 update하는 식 (3-17)에서 SW 크기를 4를 이용한다. 이는 GOP의 target rate을 구할 때 사용하는 SW 크기인 40과 picture의 target rate을 구할 때 사용하는 SW 크기인 10 보다 작은 값이다. 이는 HM 인코더의 rate control이 작은 단위마다 target rate을 맞추는 것 보다 큰 단위에서 전체 target rate을 맞추는 것을 목표로 하고 있음을 의미한다.

$$R_{Target,Curr}^{CTU} = \begin{cases} R_{Est,Curr}^{CTU} + \frac{1}{SW} \cdot (R_{RealLeft}^{CTU} - R_{Est,Left}^{CTU}) & N_{Left} > SW \\ R_{Target,Curr}^{CTU} = \frac{R_{RealLeft}^{CTU}}{N_{Left}^{CTU}} & else \end{cases} \tag{3-17}$$

3.3 λ and QP calculation

R-λ model에서 사용하는 rate은 실제 bit의 수가 아닌 pixel별

rate인 bpp이기 때문에 구해진 target rate을 식 (3-18)과 같이 pixel 수로 나누어서 bpp로 만들어 준다. 구해진 rate을 식 (3-19)의 R-λ model을 이용하여 model parameter α, β를 이용하여 λ값을 계산한다. HM 인코더는 구해진 λ값을 이전 picture나 CTU와 너무 큰 변화가 발생하지 않도록 이전 λ값과 관련된 특정 범위에서 clipping을 수행하여 최종 λ값을 계산한다.

$$R(bpp) = R_{Target} / \#of\ \ pixels \tag{3-18}$$

$$\lambda = \alpha \cdot R^{\beta} \tag{3-19}$$

구해진 λ값으로부터 식 (3-20)의 λ와 QP의 관계를 이용하여 인코딩 때 사용할 정수 QP값을 계산한다. 식 (3-20)는 다양한 이미지 시퀀스와 다양한 GOP configuration에 대해서 다양한 λ을 정하고 multiple QP 인코딩을 수행하여 그림 3-4와 같이 최적 QP의 분포로부터 구한 평균식이다. 그림 3-4에서 ln(λ)값과 최적 평균 QP의 분포를 모으면 그래프로 나타내면 그림 3-5와 같고 평균적인 추세선을 구한 것이 HM 인코더에서 사용하고 있는 식 (3-20)이다. HM 인코더는 구해진 QP값을 λ값과 마찬가지로 이전 picture나 CTU와 너무 큰 변화가 발생하지 않도록 이전 QP값과 관련된 특정 범위에서 clipping을 수행하여 인코딩할 때 사용할 QP값을 구한다.

$$QP = round(4.2005 \cdot \ln\lambda + 13.71222) \tag{3-20}$$

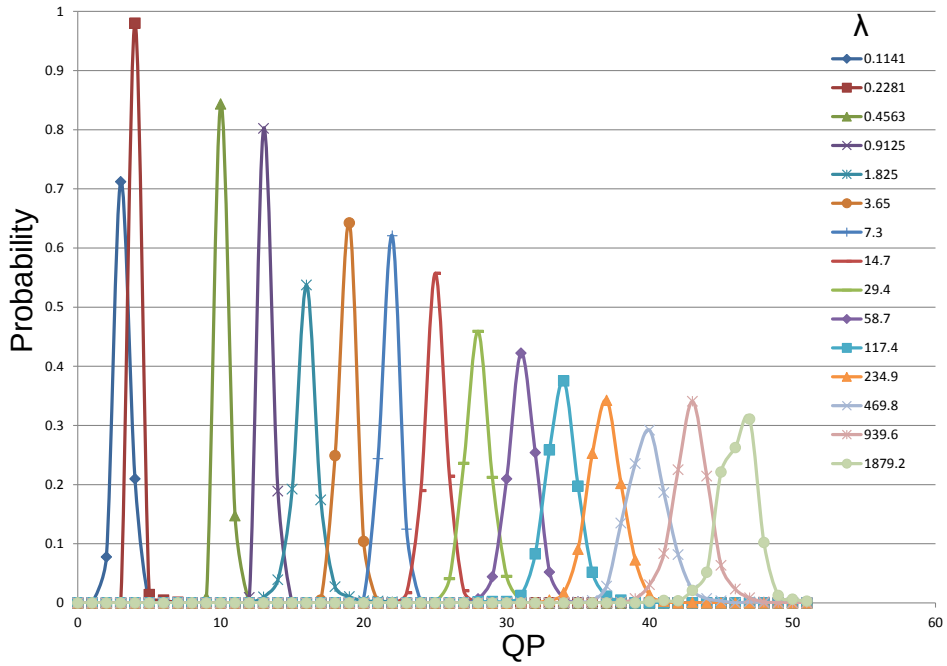


그림 3-4 BlowingBubbles 이미지 시퀀스의 LB configuration의 λ 값에 따른 multiple QP 인코딩 시 최적 QP의 분포

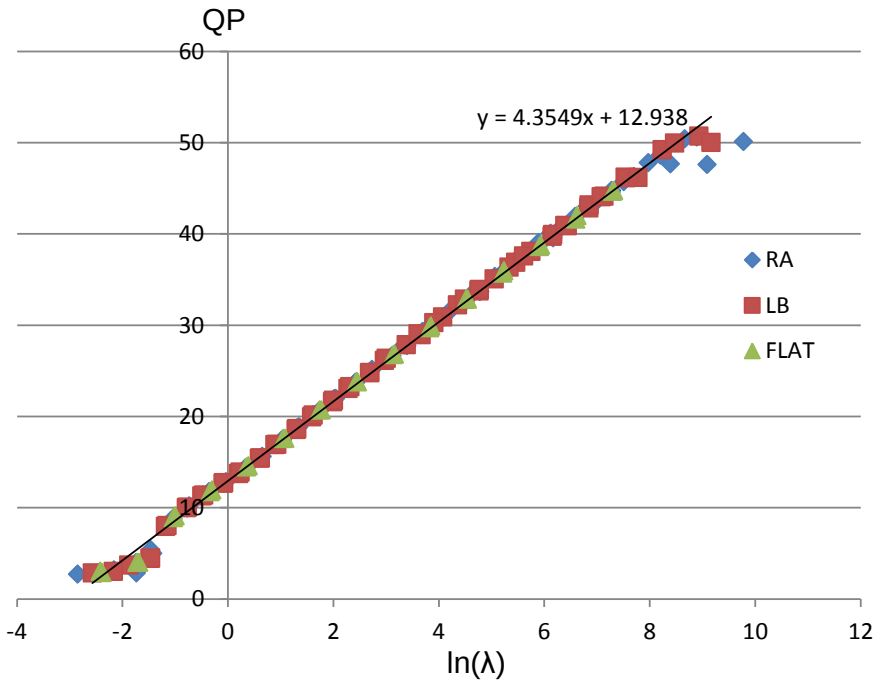


그림 3-5 BlowingBubbles 이미지 시퀀스의 RA, LB configuration과 B-picture의 QP가 모두 같은 flat configuration의 ln(λ)값과 최적 평균 QP의 관계

3.4 Encoding

Rate control 과정에서 계산된 λ와 QP를 이용하여서 다양한 모드들 중 RDO 과정을 통해 최적 모드를 찾는 인코딩을 수행한다. HM 인코더에서는 RDO 과정에서는 rate을 구하기 위해서 rate의 오차가 거의 없는 rate estimator를 이용하고 slice의 모드를 모두 결정한 후에 slice 전체 대해서 실제 entropy 코더를 수행하여 실제 bitstream을 생성한다. 따라서 CTU 단위의 rate control을 이용하는 인코딩 과정에서 오차가 무시할 정도인 rate estimator를 통해 구해진 rate을 실제 rate으로 간주하여 인코딩을 수행한다.

3.5 Model parameter update

HM 인코더에 적용된 R-λ model의 rate control은 CTU단위 rate control을 수행할 때 picture level별과 각각 CTU 별로 model parameter를 따로 관리한다. 그림 3-6은 picture level과 CTU 별로 model parameter가 따로 관리되고 update되는 것을 보여준다.

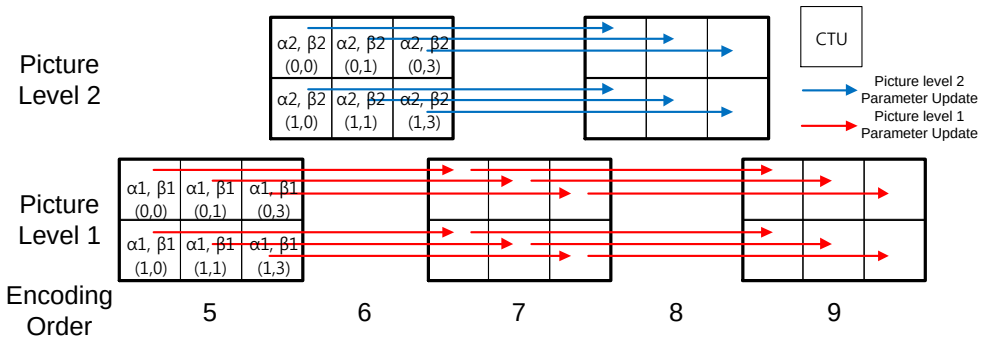


그림 3-6 HM 인코더의 R-λ model의 model parameter update 과정

HM 인코더는 기고서 [24]에서 제안한 I-picture의 bit 할당 방법을 채택하여 I-picture는 B-picture로 구성된 GOP의 hyperbolic 관계에서 구한 α, β와는 다른 model parameter를 가진다. I-picture의 model parameter의 update는 식 (3-21), (3-22)에서와 같이 각각 ln bpp와 $\Delta\lambda$를 target rate과 실제 발생한 rate으로부터 계산하고 식 (3-23), (3-24)에서와 같이 α, β를 update한다.

$$\ln bpp = \ln(C_{Intra} / pixels)^{\beta} = \beta \cdot \ln(C_{Intra} / pixels) \quad (3-21)$$

$$\Delta\lambda = \lambda_{actual} - \lambda_{target} = 0.25 \cdot \beta \cdot (\ln R_{actual} - \ln R_{target}) \quad (3-22)$$

$$\alpha_{new} = \alpha_{old} \cdot e^{\Delta\lambda} = \alpha_{old} \cdot e^{0.25 \cdot \beta} \cdot \frac{R_{actual}}{R_{target}} \quad (3-23)$$

$$\beta_{new} = \beta_{old} + \Delta\lambda / \ln bpp = \beta_{old} + (0.25 \cdot \beta \cdot (\ln R_{actual} - \ln R_{target})) / \ln bpp \quad (3-24)$$

B-picture에서의 model parameter update는 R-λ model의 model parameter인 α, β를 update한다. 발생한 bpp가 0.0001 보다 작을 경우의 B-picture에서의 model parameter의 update는 식 (3-25) (3-26)과 같이 model parameter의 값을 약간 작게 한다. 새로운 값을 기존의 값에 반영하는 정도인 update parameter인 δ_{α} 와 δ_{β} 는 각각 기본적으로 0.1와 0.05의 값을 가지며 시퀀스에 target rate에 커짐에 따라서 δ_{α} 와 δ_{β} 값을 조금씩 큰 값을 이용한다.

$$\alpha_{new} = (1 - 0.5 \cdot \delta_{\alpha}) \cdot \alpha_{old} \quad (3-25)$$

$$\beta_{new} = (1 - 0.5 \cdot \delta_{\beta}) \cdot \beta_{old} \quad (3-26)$$

발생한 bpp가 0.0001 보다 클 경우는 B-picture에서의 model

parameter의 update는 식 (3-27)에서와 같이 실제 발생한 rate을 R-λ model에 대입하여 대응하는 λ값을 계산한다. 계산한 새로운 λ값을 α와 β에 대해서 식 (3-28)과 식 (3-29)를 각각 이용하여서 model parameter를 update한다.

$$\lambda_{real} = \alpha_{old} \cdot R_{real}^{\beta_{old}} \tag{3-27}$$

$$\alpha_{new} = \alpha_{old} + \delta_{\alpha} \cdot (\log \lambda_{target} - \log \lambda_{real}) \cdot \alpha_{old} \tag{3-28}$$

$$\beta_{new} = \beta_{old} + \delta_{\beta} \cdot (\log \lambda_{target} - \log \lambda_{real}) \cdot \log R_{real} \tag{3-29}$$

식 (3-28) (3-29)의 model parameter update 수식의 유도 과정[12]은 adaptive least mean square (LMS) 방법으로 iteration을 한 번 수행하여 target rate과 실제 발생한 rate의 차이를 줄이도록 하여 유도하였다. 식 (3-30), (3-31)은 target rate과 실제 발생한 rate으로부터 R-λ model을 통해 구한 λ값을 보여준다.

$$\lambda_{target} = \alpha \cdot R_{target}^{\beta} \tag{3-30}$$

$$\lambda_{real} = \alpha \cdot R_{real}^{\beta} \tag{3-31}$$

식 (3-31)에 log를 취하여 전개하면 식 (3-32)와 같다.

$$\log \lambda_{real} = \log \alpha + \beta \cdot \log R_{real} \tag{3-32}$$

log를 취한 λ값의 차의 제곱은 식 (3-33)이 된다.

$$e^2 = (\log \lambda_{target} - \log \lambda_{real})^2 \tag{3-33}$$

식 (3-33)을 log α와 β로 각각 미분한 식은 식 (3-34)와 식 (3-35)와 같다.

$$\frac{\partial e^2}{\partial \log \alpha} = \frac{\partial e^2}{\partial \log \lambda_{real}} \cdot \frac{\partial \log \lambda_{real}}{\partial \log \alpha} = -2 \cdot (\log \lambda_{target} - \log \lambda_{real}) \quad (3-34)$$

$$\frac{\partial e^2}{\partial \beta} = \frac{\partial e^2}{\partial \log \lambda_{real}} \cdot \frac{\partial \log \lambda_{real}}{\partial \beta} = -2 \cdot (\log \lambda_{target} - \log \lambda_{real}) \cdot \log R_{real} \quad (3-35)$$

식 (3-32)에 LMS 방법을 log α와 β에 대해 적용한 식은 각각 식 (3-36)과 식 (3-37)과 같다. δ_α와 δ_β는 update parameter를 나타낸다.

$$\log \alpha_{new} = \log \alpha_{old} + \delta_\alpha \cdot (\log \lambda_{target} - \log \lambda_{real}) \quad (3-36)$$

$$\beta_{new} = \beta_{old} + \delta_\beta \cdot (\log \lambda_{target} - \log \lambda_{real}) \cdot \log R_{real} \quad (3-37)$$

식 (3-36)의 새로운 log α로부터 새로운 α를 계산하면 아래 식 (3-38)과 같다.

$$\alpha_{new} = \alpha_{old} \cdot e^{\delta_\alpha \cdot (\log \lambda_{target} - \log \lambda_{real})} \quad (3-38)$$

식 (3-38)을 Taylor 전개하여 1차 항만 남기면 식 (3-39)와 같고 이는 HM 인코더에서 사용하는 model parameter update 식이다.

$$\begin{aligned} \alpha_{new} &\approx \alpha_{old} \cdot (1 + \delta_\alpha \cdot (\log \lambda_{target} - \log \lambda_{real})) \\ &= \alpha_{old} + \delta_\alpha \cdot (\log \lambda_{target} - \log \lambda_{real}) \cdot \alpha_{old} \end{aligned} \quad (3-39)$$

제 4 장 HEVC의 CTU-level Rate control의 알고리듬의 코딩 효율 개선

HM 인코더에서 CTU-level rate control을 수행할 경우에 rate control을 수행하지 않았을 경우보다 Y-BD rate이 평균 4.14 %를 보여 코딩 효율의 저하가 발생한다. 그리고 이미지 시퀀스에 따라서 시퀀스에 후반부에 bit rate 이 부족함으로 인해서 PSNR이 급격히 떨어지는 현상이 발생하는 경우가 있다. 이 장에서는 이미지 시퀀스 초반부에 bit을 조금 적게 할당하는 방법을 이용함으로써 이미지 시퀀스 후반부에 bit rate의 부족함으로 인한 PSNR이 떨어지는 현상을 완화시켰다. 다음으로 rate control을 이용할 경우 코딩 효율의 저하를 줄이는 방법으로 인코딩시 CTU별 λ값이 아닌 picture의 평균 λ값을 이용하는 방법을 이용하였다. 마지막으로 full RD cost 계산 시 HW 구현에 적합하도록 full RD cost를 quantization step size의 제곱으로 normalize한 normalized RD cost를 설명한다.

4.1 Rate control의 실험 환경과 HM 인코더의 실험 결과

Rate control의 코딩 효율을 평가하기 위해서 rate control을 수행하지 않았을 경우의 코딩 효율과 rate control을 수행하였을 경우를 공정히 비교할 수 있는 실험 환경을 구성하여야 한다. Rate control을 수행하지 않을 경우의 HEVC의 common test condition [19]은 10 초 분량의 이미지 시퀀스를 AI, RA, LB configuration에 대해 QP 22, 27, 32, 37을 기준으로 인코딩하여서 BD-rate을 구하여 코딩 효율을 비교한다. 이 논문에서는 10 초 분량의 Kimono, ParkScene, Cactus, BasketballDrive, BQTerrace의 5개의 1080p 이미지 시퀀스를 RA

configuration에 대해서 SW에서는 코딩 속도의 향상이 크고 HW에서는 transform과 full RD cost의 연산을 크게 줄일 수 있는 TU split 결정을 수행하지 않는 maximum TU depth = 1인 환경을 기준으로 하였다. 반면, Maximum TU depth를 1로 결정하고 인코딩하였을 때 RA configuration에서 코딩 효율이 평균 1.0 % 정도 떨어진다.

표 4-1은 HM 인코더의 configuration 파일의 rate control과 관련된 변수 값들을 보여준다. KeepHierarchicalBit 변수는 GOP 구조에 따라서 GOP의 picture별 bit 할당을 하는 방법을 나타낸다. 0은 모든 B-picture가 같은 rate의 비율로 bit 할당을 하는 것이고 1은 고정된 rate의 비율로 hierarchical bit 할당을 하는 것이다. 2는 기본값으로 이전 picture들의 인코딩 결과로부터 구해진 model parameter를 반영하여 picture별 hierarchical bit 할당을 하는 방법이다. CTULevelRateControl 변수는 0일 경우 picture level rate control을 수행하고 1일 경우 CTU level rate control을 수행한다. 기본값으로 1인 CTU level rate control을 이용하고 있다. RCCTUSeparateModel 변수는 0일 경우 CTU level rate control을 수행할 때 picture내의 CTU가 하나의 model parameter를 가지고 인코딩을 수행하고 1일 경우 CTU가 각각의 model parameter를 가지고 인코딩을 수행한다. 1을 기본값으로 가지고 CTU가 각각의 model parameter를 가지고 rate control을 한다. Initial QP 변수는 이미지 시퀀스의 처음 I-picture의 QP값의 결정 방법을 나타낸다. 0일 경우 I-picture의 complexity를 구하여 QP값을 계산하고 1일 경우 QP값을 configuration 파일로 input으로 받아서 I-picture의 QP를 설정한다. 기본값으로 0을 사용하여 I-picture의 complexity를 계산하여 QP를 계산한다.

표 4-1 HM 인코더의 configuration 파일의 rate control과 관련된 변수 값

Variable	값	설명
KeepHierarchicalBit	2	0 : equal bit allocation 1 : fixed ratio bit allocation 2 : adaptive ratio bit allocation
CTULevelRateControl	1	0 : picture level rate control 1 : CTU level rate control
RCCTUSeparateModel	1	0 : Picture내의 CTU가 하나의 R-λ model 이용 1 : CTU마다 각각의 R-λ model 이용
Initial QP	0	0 : 처음 I-picture의 QP를 계산하여 이용 1 : 처음 I-picture의 QP를 input으로 받음

Rate control을 수행하지 않았을 경우의 코딩 효율과 rate control을 수행하였을 경우의 비교를 위해 rate control을 수행할 경우의 target rate을 rate control을 사용하지 않고 기본 QP 22, 27, 32, 37로 인코딩하였을 경우 발생한 rate들로 설정하였다. Rate control을 이용하지 않았을 경우의 rate과 PSNR들을 기준으로 두고 rate control을 수행한 결과로 코딩 효율을 비교하였고 더불어 target rate과 실제 발생 rate의 bit 발생 error를 구하는 것을 실험 환경으로 설정하였다. 평균 PSNR에 대해 BD rate [23]을 구할 경우에 식 (4-1)과 같이 Y, Cb, Cr의 PSNR을 6:1:1로 평균한 것을 이용하였다. 그리고 bit error는 식 (4-2)와 같이 target rate과 실제 발생 rate의 차를 target rate으로 나눈 값을 이용하였다.

$$PSNR_{avg} = (6 \cdot PSNR_Y + PSNR_U + PSNR_V)/8 \qquad (4-1)$$

$$Err = \frac{|R_{target} - R_{actual}|}{R_{target}} \cdot 100 \qquad (4-2)$$

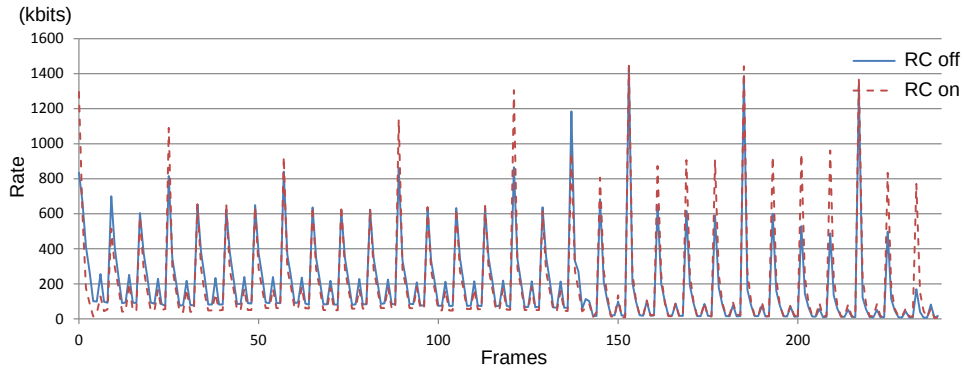
표 4-2는 HM 인코더의 rate control의 RA configuration에 대해 maximum TU depth = 1로 설정하고 기본 QP 22, 27, 32, 37로 인코딩한 것을 기준으로 하고 이를 target rate으로 rate control을 이용하여 인코딩하였을 경우의 BD rate을 보여준다. Rate control을 수행하지 않았을 경우에 비해 luma에 대해서 평균 4.14 %의 BD rate 증가를 보여 코딩 효율의 저하를 보인다. Bit error는 평균 0.22 %로 작음을 알 수 있다.

표 4-2 HM 인코더의 rate control의 CTU-level rate control의 코딩 효율

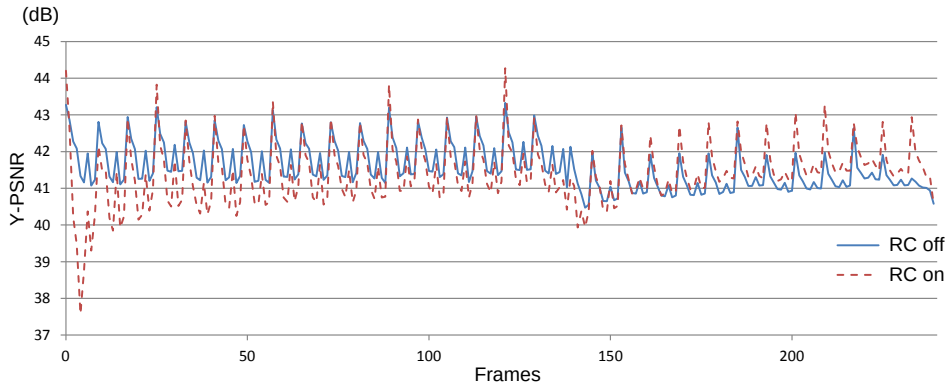
1080p	CTU-level rate control			
	BD rate (%)			Err (%)
	Y	UV	6:1:1	
Kimono	6.38	-6.94	4.16	0.47
ParkScene	2.27	0.06	1.90	0.04
Cactus	2.98	-0.56	2.38	0.01
BasketballDrive	4.72	-2.95	3.10	0.01
BQTerrace	4.36	1.23	3.84	0.57
Average	**4.14**	-1.84	**3.08**	**0.22**

그림 4-1은 rate control을 이용하지 않았을 경우와 이용하였을 경우의 Kimono 이미지 시퀀스의 picture별 발생 rate과 PSNR의 비교이다. Rate control을 이용할 경우와 rate control을 이용하지 않을 경우의 rate과 PSNR은 비슷한 모양을 가지는 것을 볼 수 있다. 이는 rate control을 수행할 경우 rate control을 이용하지 않은 경우와 비슷한 코딩 효율을 가지기 위해서 GOP 구조의 정보를 이용하여 bit 할당을 하고 있기 때문이다. 반면, rate control을 이용할 경우에 rate과 PSNR의 picture별 변화가 rate control을 이용하지 않았을 경우보다 큰 것을 볼 수 있다. 이는 rate control을 이용하지 않을 경우에는 QP의 변화폭이 이미지 시퀀스 전체에서 정해져 있지만 rate control을 이용할

경우에는 QP의 변화폭이 클 수 있기 때문이다.



(a)



(b)

그림 4-1 Kimono 이미지 시퀀스의 rate control을 이용하지 않고 RA configuration의 기본 QP 22로 인코딩하였을 경우와 target rate을 4722.1 kbps로 rate control을 이용하여 인코딩 하였을 경우의 picture별 (a) 발생 rate의 비교 (b) PSNR의 비교

4.2 Bit saving을 이용한 Picture-level bit allocation

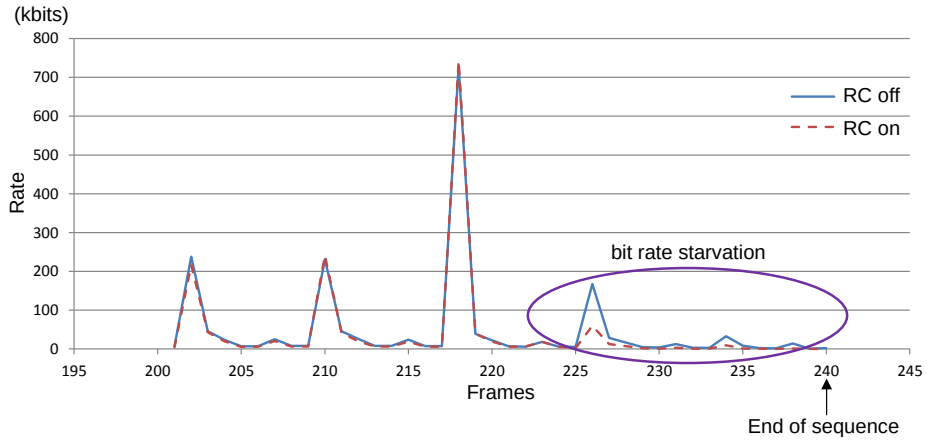
Rate control을 이용하여 인코딩을 수행할 때 전체 이미지 시퀀스의 picture별 복잡도 정보를 알지 못 하고 인코딩을 하기 때문에 이미지 시퀀스에 따라서 이미지 시퀀스의 후반부에 bit rate 부족함 [25] 으로 인해 PSNR이 급격히 떨어지는 현상이 발생한다. 이를 해결하는 것은

이미지 시퀀스 후반을 알지 못 해서 발생하는 만일의 사태(contingency)문제를 푸는 것이다.

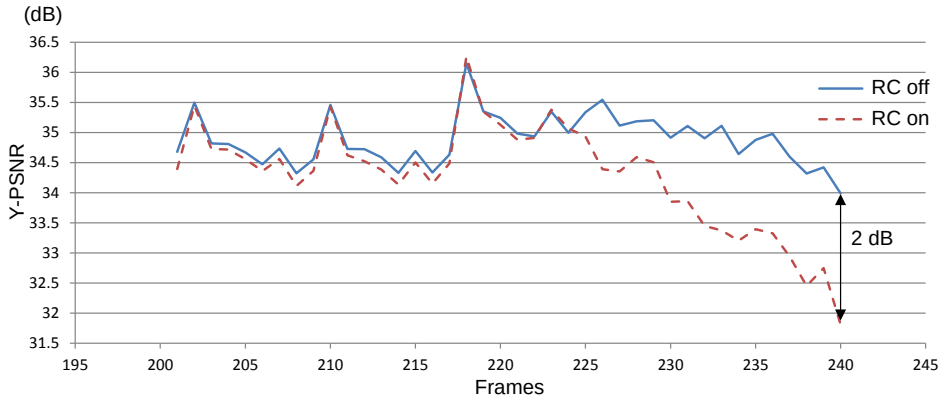
이미지 시퀀스 전체의 maximum distortion은 인지적으로 중요하기 때문에 이미지 시퀀스 전체에서 평균 distortion을 줄이는 것뿐만 아니라 maximum distortion을 줄이는 것이 필요하다. Just noticeable distortion (JND)는 사람이 눈으로 인지할 수 없는 최대 distortion을 의미하는 용어로 주관적 비디오 화질을 이용하여 인코딩을 할 때 이용된다. [26] 논문에서는 최대 distortion을 객관적인 비디오 화질 평가 metric에 반영한 식 (4-3)의 metric을 이용하였다. Maximum distortion을 고려가 주관적과 객관적 화질 평가에서도 이용되기 때문에 rate control을 이용한 인코딩 시에도 이미지 시퀀스이 평균 PSNR 뿐만 아니라 최소 PSNR을 높이는 것이 필요하다.

$$\text{Video Quality Metric} = D_{avg} + 0.005 \cdot D_{max} \qquad (4-3)$$

그림 4-2 (a)와 같이 HM 인코더의 rate control 알고리듬은 이미지 시퀀스에 따라서 후반에 bit rate이 모자라는 현상으로 인해서 그림 4-2 (b)와 같이 PSNR이 급격히 저하가 현상이 발생하는 경우가 있다. 반면 그림 4-3은 이미지 시퀀스 후반에 bit rate이 남아서 rate control을 수행하지 않을 경우보다 PSNR이 높은 경우의 이미지 시퀀스를 보여준다. 이 논문에서는 이미지 시퀀스 후반에 PSNR의 급격한 저하를 완화시키기 위해서 이미지 시퀀스 초반의 picture bit을 조금 적게 bit 할당하는 방법을 적용하였다.

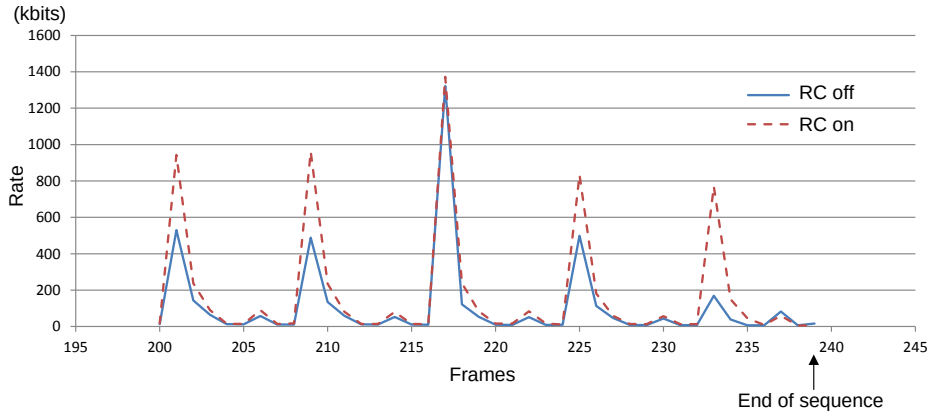


(a)

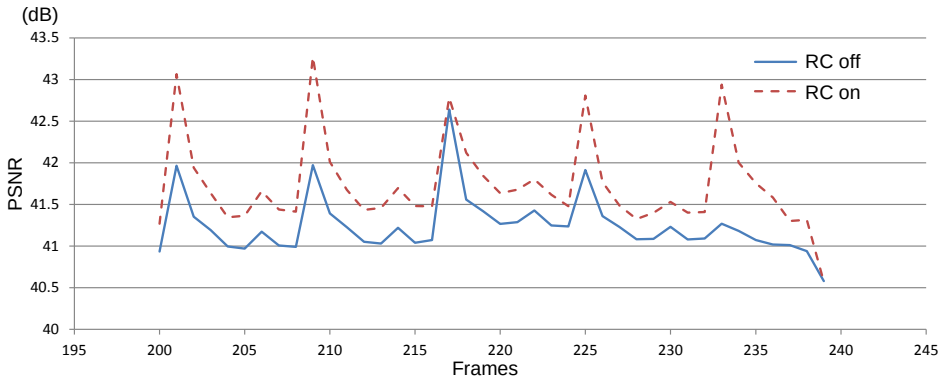


(b)

그림 4-2 ParkScene 이미지 시퀀스의 rate control을 이용하지 않고 RA configuration의 기본 QP 32로 인코딩하였을 경우와 target rate을 1444.83 kbps로 rate control을 이용하여 인코딩 하였을 경우의 시퀀스의 후반의 picture별 (a) 발생 rate의 비교 (b) PSNR의 비교



(a)



(b)

그림 4-3 Kimono 이미지 시퀀스의 rate control을 이용하지 않고 RA configuration의 기본 QP 22로 인코딩하였을 경우와 target rate을 4722.1 kbps로 rate control을 이용하여 인코딩 하였을 경우의 시퀀스의 후반의 picture별 (a) 발생 rate의 비교 (b) PSNR의 비교

HM 인코더에서 picture로 구성된 GOP의 bit 할당은 이미지 시퀀스 전체에 대해서 평균 rate을 맞추기 위한 것으로 그림 4-2 (a)와 같이 시퀀스 후반에 bit rate 이 부족한 현상으로 인해 PSNR이 급격히 낮아지는 것을 보이는 경우가 있다. 이를 완화하기 위해서 식 (4-4)와 같이 시퀀스 초반의 GOP에 bit을 조금 덜 할당하는 방법을 적용하였다. 식 (4-4)의 bit 비축하는 값은 picture를 인코딩함에 따라서 그림 4-4와 같이 줄어들고 SW값보다 남은 picture의 수가 작을 경우에는 HM

인코더에서와 같이 남은 rate을 균등하게 bit 할당을 하게 하였다.

$$R_{Avg}^{Pic} = \frac{R_{Total}}{N_{Total}^{Pic}} + \frac{1}{SW} \cdot (R_{Left} - N_{Left}^{Pic} \cdot \frac{R_{Total}}{N_{Total}^{Pic}}) - M \cdot \frac{N_{Left}^{Pic}}{N_{Total}^{Pic}} \cdot \frac{R_{Total}}{N_{Total}^{Pic}} \quad (4-4)$$

식 (4-5)는 I-picture에 대해서 시퀀스 초반의 bit을 조금 덜 allocation 하는 식이다. I-picture는 GOP에 포함되지 않고 따로 complexity를 구하여 식 (4-6)의 남은 bit을 고려하여 bit allocation을 수행하기 때문에 GOP의 bit allocation과 구분한다.

$$R_{I-Slice} = a \cdot \left(\frac{4.0 \cdot C_{Intra}}{R_{Avg}^{Pic}} \right)^{b} \cdot (R_{Avg}^{Pic} - M \cdot \frac{N_{Left}^{Pic}}{N_{Total}^{Pic}} \cdot R_{Avg}^{Pic}) \quad (4-5)$$

$$R_{avg}^{Pic} = \frac{R_{Left}}{N_{Left}^{Pic}} \quad (4-6)$$

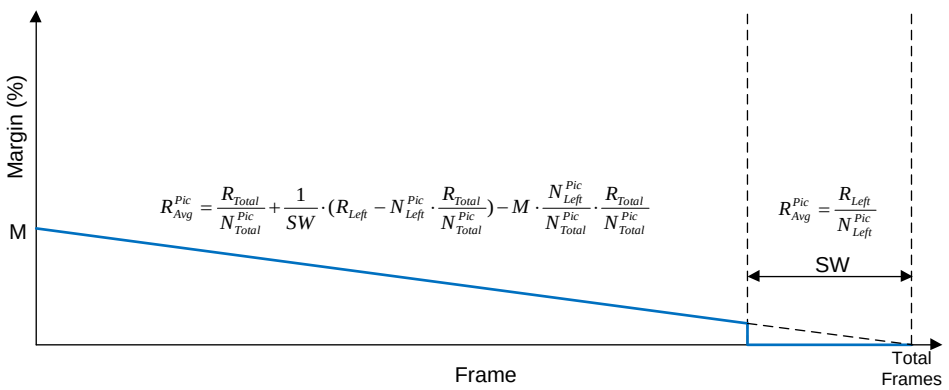
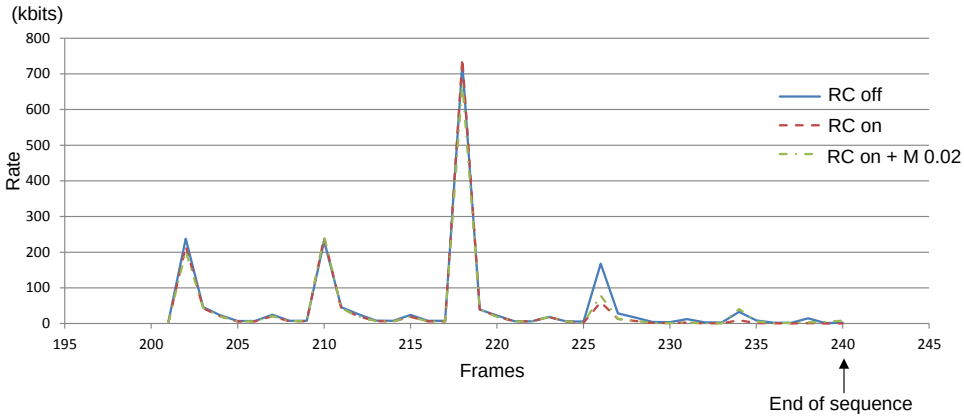
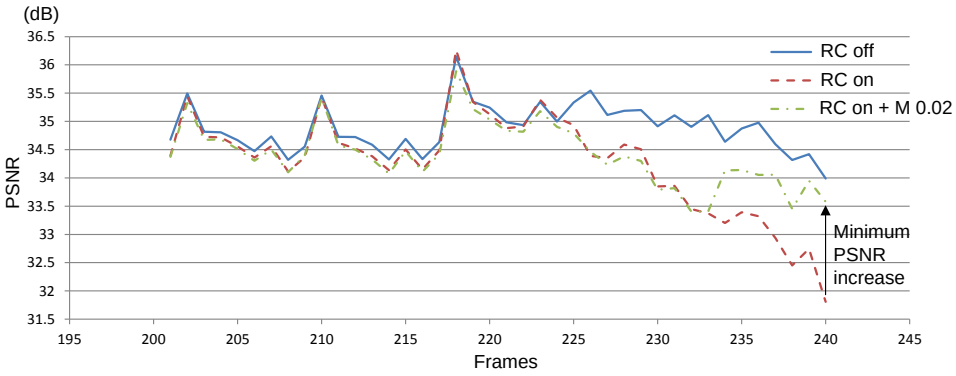


그림 4-4 제안하는 GOP level의 target bit 할당의 frame이 지남에 따른 bit 비축량의 변화

처음에 bit 할당을 식 (4-4)의 M = 0.02를 이용하여 조금 적게 하였을 경우에 그림 4-5 (b)에서와 같이 이미지 시퀀스 후반에 PSNR 저하가 완화되는 현상을 보인다. 이를 객관적으로 측정하기 위해서 이미지 시퀀스의 picture 별 평균 PSNR 값과 더불어 최소 PSNR을 이용하여 rate control의 효율을 판단하는 metric으로 이용하였다.



(a)



(b)

그림 4-5 ParkScene 이미지 시퀀스의 rate control을 이용하지 않고 RA configuration의 기본 QP 32로 인코딩하였을 경우와 target rate을 1444.83 kbps로 rate control을 이용하여 인코딩 하였을 경우의 시퀀스의 후반의 picture별 (a) 발생 rate의 비교 (b) PSNR의 비교

표 4-3는 HM 인코더의 rate control과 식 (4-4)에서 M=0.02로 하였을 경우의 이미지 시퀀스별 PSNR의 평균, 표준편차, 최대값, 최소값 비교이다. M=0.01, 0.02, 0.03, 0.05일 경우에 대해서 실험을 하였고 그 중 M=0.02의 값을 가질 경우가 평균 PSNR의 저하가 거의 없으면서 minimum PSNR의 상승이 커서 M=0.02값의 데이터를 제시하였다. M=0.02로 하였을 경우에 minimum PSNR이 HM 인코더의 rate control을 이용하여 인코딩 하였을 경우에 비해서 평균 적으로

0.11 dB 올라갔고 최대 1.58 dB까지 좋아진 경우도 있다. 표 4-3에서 회색으로 표시된 부분은 bit rate 부족함으로 인해서 시퀀스에 후반부에 PSNR이 떨어지는 현상이 있는 인코딩 환경이다. HM 인코더의 rate control인 M=0인 경우보다 평균 BD rate은 M=0.02인 경우 평균 0.12 %p 상승하여 약간의 코딩 효율의 저하가 있었다. 하지만 minimum PSNR은 평균 0.11 dB 상승하여 최악의 상황에 대한 코딩 효율이 크게 좋아졌다.

표 4-3. HM 인코더의 rate control과 식 (4-4)에서 M=0.02로 하였을 경우의 picture별 PSNR의 평균, 표준편차, 최대값, 최소값 비교

1080p	RC off 시의 QP	Bit rate starvation	HM rate control PSNR		식 (3-3)의 M 2% PSNR		⊿PSNR	
			μ (dB)	min (dB)	μ (dB)	min (dB)	⊿μ (dB)	⊿min (dB)
Kimono	22	x	41.37	37.58	41.37	37.51	0.00	-0.07
	27	x	39.49	35.37	39.49	35.33	0.00	-0.04
	32	x	37.23	33.12	37.20	33.05	-0.03	-0.07
	37	o	34.87	31.12	34.86	31.00	-0.01	-0.12
ParkScene	22	x	39.88	38.43	39.88	38.42	0.00	-0.01
	27	x	37.41	36.07	37.41	36.06	0.00	-0.01
	32	o	34.80	31.81	34.81	33.39	0.01	1.58
	37	o	32.27	29.99	32.28	30.71	0.01	0.72
Cactus	22	x	38.43	37.14	38.43	37.12	0.00	-0.02
	27	o	36.78	35.32	36.77	35.47	-0.01	0.15
	32	o	34.88	33.57	34.88	33.62	0.00	0.05
	37	o	32.67	31.17	32.66	31.19	-0.01	0.02
Basketball Drive	22	x	39.05	37.04	39.05	37.03	0.00	-0.01
	27	x	37.37	35.07	37.36	35.05	-0.01	-0.02
	32	x	35.52	32.72	35.52	32.72	0.00	0.00
	37	x	33.56	30.82	33.55	30.77	-0.01	-0.05
BQTerrace	22	o	37.36	34.72	37.36	34.73	0.00	0.01
	27	o	35.21	33.33	35.21	33.31	0.00	-0.02
	32	o	33.74	29.79	33.73	29.69	-0.01	-0.10
	37	o	32.03	28.26	32.07	28.52	0.04	0.26
Average	-		36.20	33.62	36.19	33.73	0.00	0.11

4.3 Picture의 평균 λ를 이용한 rate control의 코딩 효율 개선

CTU-level rate control을 이용하여 인코딩을 하였을 때 rate control을 이용하지 않았을 경우와 비교하여 코딩 효율이 떨어지는 이유[27]는 세 가지로 볼 수 있다. 첫째, collocated CTU의 정보로부터 현재 CTU의 정보를 예측하여 bit 할당을 하게 되는데 현재 CTU와 collocated CTU와의 correlation이 크지 않은 경우가 있기 때문에 bit 할당이 비효율적으로 이루어 진다. 둘째, picture의 초반 CTU에 bit 할당을 많이 하여서 picture 후반 CTU에 bit rate 부족함 현상으로 코딩 효율의 저하가 발생한다. 셋째, CTU의 특성을 고려하지 않는 uniform model을 적용하여 bit 할당을 하기 때문에 코딩 효율의 저하가 발생한다. 이 논문에서는 세 가지 이유 중 현재 CTU의 정보가 collocated CTU의 정보와 correlation이 크지 않을 수 있음을 이용하여 CTU별 Lagrangian multiplier λ를 이용하지 않고 picture의 평균 λ를 이용하여서 각 CTU를 인코딩을 수행하였다. 반면, model parameter의 update와 QP값의 결정은 HM 인코더의 CTU-level rate control과 같은 방법을 이용하였다.

표 4-4는 CTU-level rate control을 수행할 때 CTU의 계산된 λ대신 picture의 평균 λ를 이용하였을 경우의 BD rate을 보여준다. Y BD rate이 평균적으로 1 % 작아지는 것을 볼 수 있고 bit error는 평균 0.25 % 증가하였다. 코딩 효율이 1 % 정도 증가한 것은 picture의 평균 λ값이 각 CTU의 λ을 예측하여 이용하는 것 보다 평균적으로 오차가 더 작기 때문으로 여겨진다. 이는 HM 인코더에 적용된 CTU-level rate control의 각 CTU의 model parameter 예측이 잘 맞지 않는

경우가 많음을 의미한다.

표 4-4 CTU-level rate control을 수행할 때 CTU의 계산된 λ 대신 picture의 평균 λ를 이용하였을 경우의 BD rate (%)

1080p	CTU-level rate control (HM)				Picture의 평균 λ를 이용한 CTU-level rate control			
	BD rate (%)			Err (%)	BD rate (%)			Err (%)
	Y	UV	6:1:1		Y	UV	6:1:1	
Kimono	6.38	-6.94	4.16	0.47	5.90	-7.96	3.60	1.24
ParkScene	2.27	0.06	1.90	0.04	2.63	3.47	2.80	0.32
Cactus	2.98	-0.56	2.38	0.01	2.57	4.05	2.82	0.14
BasketballDrive	4.72	-2.95	3.10	0.01	1.92	0.52	1.57	0.04
BQTerrace	4.36	1.23	3.84	0.57	2.55	4.87	2.97	0.61
Average	**4.14**	-1.84	**3.08**	**0.22**	**3.11**	0.99	**2.75**	**0.47**

4.4 Full RDO에서 이용하는 normalized RD cost

HM 인코더에서 luma와 chroma를 모두 고려한 FRD cost의 계산식은 식 (4-7)과 같다. ω_{chroma}는 λ_{mode}값이 luma의 QP값과 chroma의 QP값이 다른 경우의 λ값의 QP의 지수항의 영향을 보상해 주기 위한 값으로 식 (4-8)과 같이 계산된다.

$$J_{FRD} = (D_{luma} + \omega_{chroma} \cdot D_{chroma}) + \lambda_{mode} \cdot (R_{mode} + R_{residual}) \quad (4-7)$$

$$\omega_{chroma} = 2^{(QP-QP_{chroma})/3} \quad (4-8)$$

식 (4-9)은 rate control을 이용하지 않을 경우의 [2]에서 설명하는 λ_{mode}값의 결정 방법이다. α는 현재 picture가 reference picture로 이후에 이용되는지 이용되지 않는 지와 GOP에서 B frame의 수에 따라서 정해지는 변수이고 W_k는 GOP 구조에서 현재 picture의 level 따른 weighting factor이다. λ_{mode}값은 QP값이 증가함에 따라서 지수 관계로 증가하기 때문에 넓은 dynamic range를 지원하여야 한다.

HW 구현 시에 넓은 dynamic range를 가지는 λ_{mode}값은 FRD cost를 구할 때 rate과 λ_{mode}의 곱셈이 필요하기 때문에 곱셈기 복잡도가 증가하게 되고 이는 FRD cost를 HW 인코더 구현에서 코딩 효율의 저하는 있지만 잘 사용하지 않는 이유 중 하나이다.

$$\lambda_{mode} = \alpha \cdot W_k \cdot 2^{\frac{QP-12}{3}} \qquad (4-9)$$

이 논문에서는 HW 구현의 복잡도를 줄일 수 있도록 distortion으로 transform domain의 SSE를 적용하여 구한 FRD cost를 quantization step size의 제곱인 Q_{step}^2 으로 나눈 normalized FRD cost를 사용을 제안하였다. Normalized FRD cost를 이용할 때의 장점은 다음과 같다. 첫째, cost를 구할 때 사용하는 변수들의 dynamic range가 줄어들어 HW 구현 시 복잡도가 원래 full RD cost의 구현보다 줄어든다. 둘째, transform domain의 distortion을 계산할 때 inverse transform과 reconstruction 뿐만 아니라 dequantization의 수행 또한 필요하지 않는다. 셋째, 전체 FRD cost를 구할 때 luma와 chroma의 QP값의 차이가 있을 경우 λ값의 QP의 지수항으로 인한 차이를 보상해 주기 위한 chroma weight인 ω_{chroma}의 연산이 필요하지 않다.

식 (4-10)은 normalized FRD cost J_n의 식을 보여준다. J_n은 distortion과 λ_{mode}를 Q_{step}^2 으로 각각 나눈 normalized distortion과 normalized λ를 이용한다. 식 (4-10)의 quantization step size인 Q_{step}은 QP값에 의해 정해지는 값으로 QP가 4에서 1의 값을 가지고 QP값이 6이 커질 때 마다 두 배가 되어 식 (4-11)과 같이 정의된다.

$$J_n = \frac{J}{Q_{step}^2} = D_n + \lambda_n \cdot R \qquad (4-10)$$

$$Q_{step} = 2^{\frac{QP-4}{6}} \qquad (4-11)$$

식 (4-12)는 transform domain의 SSE [28]의 계산을 보여준다. transform domain의 SSE는 reconstructed coefficient r_{ij}와 transformed coefficient c_{ij}의 차의 제곱의 합으로 계산된다. r_{ij}는 reconstructed coefficient로 quantized level l_{ij}와 Q_{step}의 곱으로 나타낼 수 있고 식 (4-13)과 같다.

$$D = \sum_{j=0}^{N-1}\sum_{i=0}^{N-1}(c_{ij} - r_{ij})^2 \qquad (4-12)$$

$$r_{ij} = l_{ij} \cdot Q_{step} \qquad (4-13)$$

식 (4-14)는 normalized distortion의 계산이다. l_{ij}^{float}는 식 (4-15)에서와 같이 transform coefficients c_{ij}를 Q_{step}으로 실수 연산으로 나눈 값으로 truncation이나 rounding을 수행하지 않은 실수 값을 의미한다. Normalized distortion은 l_{ij}^{float}와 l_{ij}의 차의 제곱의 합으로 계산이 되며 식 (4-13)의 r_{ij}를 계산하기 위한 dequantization의 계산을 수행하지 않는다.

$$D_n = \frac{D}{Q_{step}^2} = \sum_{j=0}^{N-1}\sum_{i=0}^{N-1}(l_{ij}^{float} - l_{ij})^2 \qquad (4-14)$$

$$l_{ij}^{float} = \frac{c_{ij}}{Q_{step}} \qquad (4-15)$$

식 (4-16)는 normalized λ의 계산이다. Normalized λ는 λ_{mode}의 QP에 관한 식 (4-9)의 지수항이 없어지기 때문에 λ_{mode}에 비해서 dynamic range가 크게 줄어들게 된다.

$$\lambda_n = \frac{\lambda_{mode}}{Q_{step}^2} = \alpha \cdot W_k \cdot 2^{-\frac{8}{3}} \qquad (4-16)$$

그림 4-6는 rate control을 이용할 경우와 이용하지 않을 경우의 QP와 GOP구조에 따른 $\lambda_{mode} = \lambda_n \cdot Q_{step}^2$ 에서 λ_n의 값의 비교를 보여준다. HM 인코더의 rate control을 이용할 경우에는 GOP 구조와 QP에 관계없이 일정한 값을 이용한다. Rate control을 사용하지 않을 경우에는 GOP 구조와 QP값을 반영한다. 예를 들어 reference picture로 이용되어 전체 sequence에 영향을 많이 주는 picture인 I-slice의 경우 λ_n값을 다른 picture에 비해 작은 값을 이용하여서 distortion이 다른 picture에 비해 작게 인코딩을 수행한다.

Normalized RD cost를 구하여 RDO를 수행할 때 normalized λ의 소수 부분의 bit 수를 7 bit 이상을 지원하면 코딩 효율의 저하가 무시할 정도이다. Normalized λ는 그림 4-6에서와 같이 0.5 이하의 값을 가지기 때문에 총 6 bit의 bit width만 지원하면 되는 것이다. 반면, λ_{mode}의 값은 Q_{step}^2 의 값은 QP가 0인 경우와 51인 경우에 217배의 차이가 나기 때문에 fixed point로 표현하였을 경우에 최소 17 bit의 bit width가 필요하다. 따라서 normalized λ를 이용하는 것이 λ_{mode}를 이용하는 것 보다 HW 구현의 복잡도가 크게 줄어들게 된다.

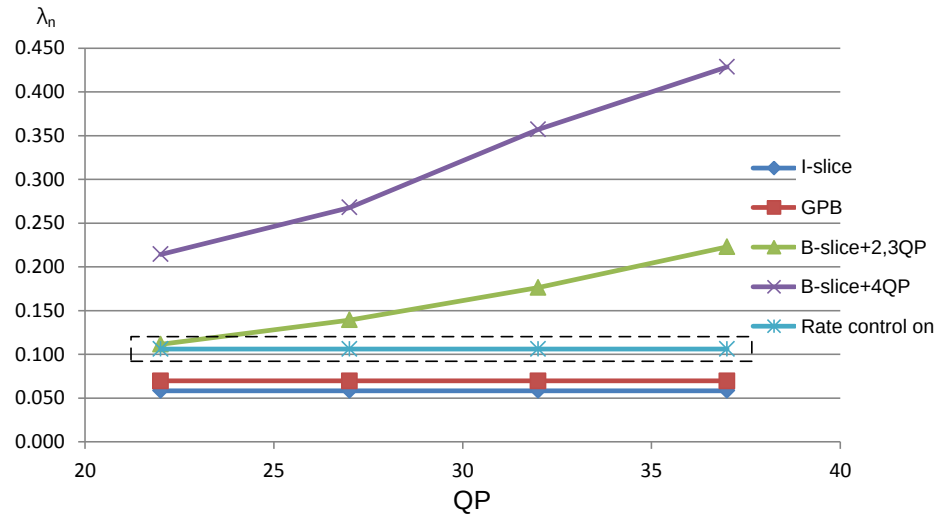


그림 4-6 Rate control on/off 시의 QP와 GOP구조에 따른 $\lambda_{mode} = \lambda_n \cdot Q_{step}^2$에서 λ_n의 값 비교

Picture의 평균 λ값을 이용하여 인코딩을 할 때 normalized λ값은 picture의 평균 QP와 CTU의 QP의 차이를 반영해 주어야 한다. 식 (4-17)은 normalized λ값과 quantization step size의 관계를 보여준다. Picture의 평균 λ값을 이용할 경우에는 식 (4-18)에서와 같이 picture의 평균 QP와 CTU의 QP의 차이를 반영한 normalized λ값을 이용한다. picture의 평균 λ값에 해당하는 normalized λ값은 QP차이마다 그림 4-7과 같이 미리 계산해 두고 이용할 수 있다.

$$\lambda_n(QP) = \frac{\lambda}{Q_{step}^2} \tag{4-17}$$

$$\lambda_{pic,n}(QP) = \frac{\lambda_{pic}}{2^{\frac{QP-4}{3}}} = \frac{\lambda \cdot (\lambda_{pic}/\lambda)}{2^{\frac{QP-4}{3}}} = \lambda_n(QP) \cdot 2^{\frac{QP_{Pic}-QP}{3}} \tag{4-18}$$

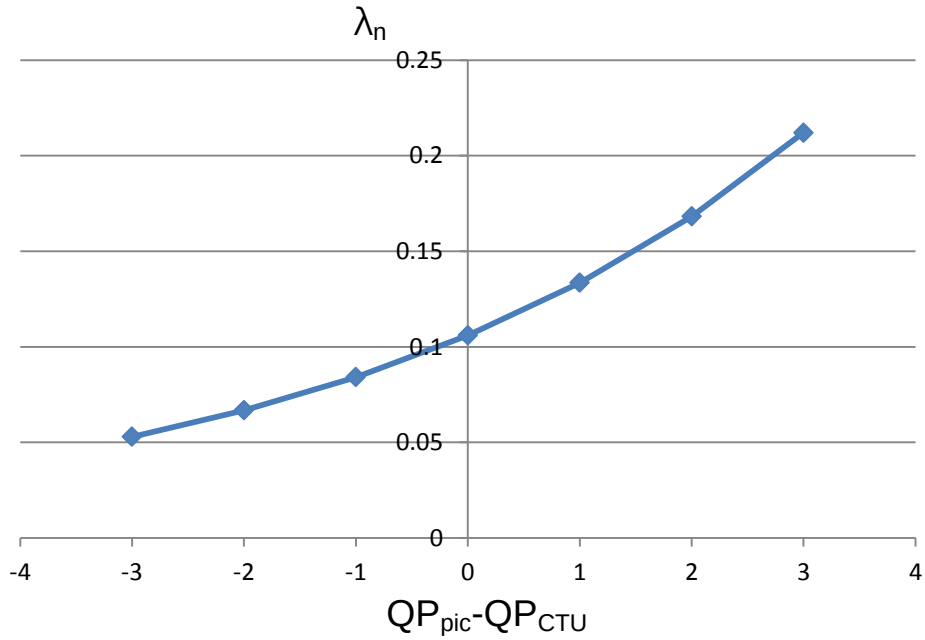


그림 4-7 Picture의 평균 λ 값을 이용할 경우의 QPpic-QP_{CTU}에 따른 normalized λ 값

제 5 장 HEVC의 CTU-level Rate control의 HW 구현

4K 30 fps의 이미지 시퀀스를 HW로 400 MHz에서 인코딩한다고 가정하였을 경우 4K 이미지 크기는 2048개의 64x64 크기의 CTU로 구성되기 때문에 CTU 당 cycle budget은 식 (5-1)과 같이 약 6510 cycle이 된다.

$$400\ \text{MHz} / (2048 \times 30) = 6510\ \text{cycles} \qquad (5-1)$$

H.264의 reference SW인 JM 인코더에 적용된 rate control [17]의 SW의 연산량과 HW 복잡도를 분석하고 있다. H.264의 reference SW 구현된 rate control [17]를 ARM에서 SW로 수행하였을 경우 macroblock 당 35k cycle의 수행 cycle을 가지고 HW로 구현하였을 경우에 144k gate의 복잡도를 가진다. [15]논문에서는 frame-level 이상의 rate control은 software로 수행하고 macroblock-level의 rate control은 HW로 수행하였다. HEVC HW 인코더에서도 CTU-level rate control을 수행은 4K 급 이상이 throughput을 만족시키기 위해서는 rate control의 HW 구현이 필요하다.

이 장에서는 HM 인코더의 R-λ model의 rate control을 HW 구현에 적합하게 log를 취한 log R-log λ model을 이용하도록 rate control 알고리듬을 변경한다. log R-log λ model을 이용한 rate control은 계산의 간단화와 더불어 코딩 효율의 상승도 보였다. Log를 취한 rate control의 계산식은 4장에서 설명한 HW 구현에 적합한

normalized RD cost의 계산과 부합하도록 λ_{mode}값을 이용하지 않고 normalized λ값만을 이용하도록 수정하였다. 또한 HW 구현을 위해 floating point를 사용하는 변수를 fixed point를 사용하도록 수정하고 log, anti-log, division의 식을 LUT를 이용하여 수행하도록 하여 HW 구현에 적합하게 수정하였다. 수정한 rate control 알고리듬을 HW로 구현하였고 구현한 rate control HW는 가정하는 HEVC HW 인코더 수행에서 CTU당 4 cycle의 overhead를 보여 무시할 수 있는 수준이다.

5.1 HW 구현을 위한 log를 취한 log R-log λ model

R-λ model을 이용한 rate control은 식 (5-2)을 이용한다. 식 (5-2)에서 α, β는 R-λ model의 model parameter를 나타내고 R은 target rate을 의미하며 bpp를 단위로 이용한다.

$$\lambda = \alpha \cdot R^{\beta} \tag{5-2}$$

식 (5-2)는 지수함수를 이용하고 α, β는 HM 인코더에서 floating point형식을 이용하기 때문에 HW 구현에 어려움이 있다. 식 (5-3)은 식 (5-2)를 HW 구현에 적합하게 지수함수를 없애기 위해 log함수를 취하여 α와 R이 $\log_2\alpha$와 $\log_2$R의 사용으로 변형하였다. 그리고 R에 β의 지수함수를 취하는 것이 $\log_2$R과 β의 곱으로 변형되었다.

$$\log_2 \lambda = \log_2 \alpha + \beta \log_2 R \tag{5-3}$$

표 5-1은 이미지 시퀀스에 따른 model parameter α, β값의 평균, 최대값, 최소값을 보여준다. B picture에서 이용되는 α는 2~3

범위에서 대체로 값이 발생하며 최대 30정도의 큰 값을 가지는 경우도 있다. β는 −2~0 범위에서 대체로 값이 발생하며 −3까지 같을 가지는 경우도 있다. β값에 비해서 α값이 넓은 범위의 값을 가지기 때문에 α값을 log를 취하여 $\log_2\alpha$을 이용하는 것은 model parameter의 범위를 비슷하게 하는 효과가 있다. 반면 rate도 다양한 target rate을 지원해야 하기 때문에 $\log_2$R을 이용하는 것은 변수의 dynamic range를 줄인다.

표 5−1 이미지 시퀀스에 따른 model parameter α, β값의 평균, 최대값, 최소값

1080p	Target rate (kbps)	α			β		
		μ	max	min	μ	max	min
Kimono	4721.3	2.266	5.907	0.328	−0.703	−0.100	−1.754
	2160.5	2.568	5.309	0.873	−0.808	−0.142	−1.756
	1051.1	2.716	3.938	1.811	−0.948	−0.457	−1.591
	531.5	2.726	3.971	1.807	−1.014	−0.625	−1.643
ParkScene	7381.9	2.400	13.810	0.310	−0.653	−0.100	−1.841
	3167.1	2.560	12.279	0.786	−0.754	−0.220	−1.764
	1444.8	2.638	4.258	1.833	−0.907	−0.535	−1.511
	667.7	2.638	4.114	1.706	−0.981	−0.654	−1.583
Cactus	18975.0	2.914	24.776	0.050	−0.678	−0.100	−1.776
	6110.9	2.828	26.802	0.159	−0.753	−0.113	−1.742
	2881.1	2.676	7.324	1.096	−0.899	−0.428	−1.666
	1486.6	2.664	6.286	0.932	−0.989	−0.540	−1.846
Basketball Drive	17804.1	3.690	23.147	0.079	−0.593	−0.100	−1.961
	6193.6	3.412	21.797	0.457	−0.735	−0.135	−1.875
	2897.4	2.806	6.589	1.439	−0.904	−0.492	−1.727
	1521.0	2.771	5.631	1.337	−1.003	−0.595	−1.772
BQTerrace	40939.3	2.862	**30.395**	**0.050**	−1.167	**−0.100**	**−3.000**
	8380.9	2.364	16.865	0.231	−0.730	−0.200	−1.973
	2826.3	2.281	5.097	0.972	−0.812	−0.425	−1.647
	1297.6	2.177	5.012	0.845	−0.920	−0.571	−1.684

식 (5−4)는 λ와 quantization step size와의 관계를 보여준다. 식 (5−5)는 QP와 quantization step size의 관계를 나타낸다. 식 (5−2), (5−4), (5−5)로부터 $\log_2$R로부터 QP를 구하는 식을 정리하면 식 (5−6)과 같다. 식 (5−6)의 $\log_2\lambda_n$은 상수 값이고 R에 log를 취한

$\log_2$R값과 $\log_2\alpha$, β 값을 이용하여 λ_{mode}값의 계산 없이 QP값을 계산한다.

$$\lambda = \lambda_n \cdot Q_{step}^2 \tag{5-4}$$

$$Q_{step} = \frac{QP-4}{6} \tag{5-5}$$

$$\begin{aligned} QP &= 3\cdot\log_2 Q_{step}^2 + 4 \\ &= 3\cdot\log_2\left(\frac{\alpha\cdot R^{\beta}}{\lambda_n}\right) + 4 \\ &= 3\beta\cdot\log_2 R + 3\cdot\log_2\alpha - 3\cdot\log_2\lambda_n + 4 \end{aligned} \tag{5-6}$$

식 (5-7)(5-8)(5-9)는 HM 인코더에서 적용된 R-λ model의 model parameter α, β의 update 방법을 보여준다. 식 (5-7)에서 실제 발생한 rate인 R_{real}을 이용하여 실제 rate에 해당하는 λ_{real}을 계산하고 R-λ model로부터 인코딩을 위해 계산한 λ_{target}과의 차이를 이용하여 식 (5-8)과 식 (5-9)과 같이 model parameter α, β를 update한다.

$$\lambda_{real} = \alpha_{old}\cdot R_{real}^{\beta_{old}} \tag{5-7}$$

$$\alpha_{new} = \alpha_{old} + \delta_{\alpha}\cdot(\log\lambda_{target} - \log\lambda_{real})\cdot\alpha_{old} \tag{5-8}$$

$$\beta_{new} = \beta_{old} + \delta_{\beta}\cdot(\log\lambda_{target} - \log\lambda_{real})\cdot\log R_{real} \tag{5-9}$$

식 (5-10)는 식 (5-7)을 이용한 $\log\lambda_{target}$과 $\log\lambda_{real}$값의 차이를 정리한 식이다. $\log_2\alpha$에 해당하는 항이 없어지고 $\log R_{target}$과 $\log R_{real}$값의 차와 β의 곱으로 나타나게 된다.

$$\log\lambda_{target} - \log\lambda_{real} = \log\alpha_{old} \cdot R_{target}^{\beta_{old}} - \log\alpha_{old} \cdot R_{real}^{\beta_{old}}$$

$$= \log\alpha_{old} + \beta_{old} \cdot \log R_{target} - (\log\alpha_{old} + \beta_{old} \cdot \log R_{real})$$

$$= \beta_{old} \cdot (\log R_{target} - \log R_{real}) \quad (5-10)$$

식 (5-11)은 log를 취한 R-λ model의 model parameter $\log\alpha$의 update식을 보여준다. α의 update는 [12]논문의 유도에서 식 (5-11)의 $\log\alpha$의 update식에서 유도한 것으로 식 (5-11)을 이용하는 것이 원래 update식을 이용하는 것으로 더 정확하다. 식 (5-10)을 식 (5-11)에 대입하면 식 (5-12)과 같이 λ의 계산 없이 target rate과 실제 rate으부터 $\log\alpha$를 바로 update할 수 있다.

$$\log\alpha_{new} = \log\alpha_{old} + \delta_{\alpha} \cdot (\log\lambda_{target} - \log\lambda_{real}) \quad (5-11)$$

$$= \log\alpha_{old} + \delta_{\alpha} \cdot \beta_{old} \cdot (\log R_{target} - \log R_{real}) \quad (5-12)$$

식 (5-13)은 log를 취한 R-λ model의 model parameter β의 update식을 보여준다. 식 (5-10)를 식 (5-13)에 대입하면 식 (5-14)와 같이 λ의 계산 없이 target rate과 실제 rate으로부터 β를 바로 update할 수 있다.

$$\beta_{new} = \beta_{old} + \delta_{\beta} \cdot (\log\lambda_{target} - \log\lambda_{real}) \cdot \log R_{real} \quad (5-13)$$

$$= \beta_{old} + \delta_{\beta} \cdot \beta_{old} \cdot (\log R_{target} - \log R_{real}) \cdot \log R_{real} \quad (5-14)$$

표 5-2는 HM 인코더의 rate control과 log를 적용한 R-λ model을 이용하였을 경우의 rate control의 HM 인코더에서 rate control을 이용하지 않았을 경우에 대한 BD rate을 보여준다. 평균 1.5 %p의 BD rate 향상을 보였다. Bit error는 평균 0.25%p가 증가하였다. 이는 α의 update를 원래의 log α의 update를

이용함으로써 model parameter update의 오차가 작아져 코딩 효율이 좋아진 것으로 판단된다.

표 5-2 log를 취한 log R-log λ model 이용하였을 경우의 BD rate (%)

1080p	CTU-level rate control (HM)				log를 취한 R-λ model			
	BD rate (%)			Err (%)	BD rate (%)			Err (%)
	Y	UV	6:1:1		Y	UV	6:1:1	
Kimono	6.38	-6.94	4.16	0.47	4.44	-6.40	2.65	0.75
ParkScene	2.27	0.06	1.90	0.04	0.62	2.21	0.89	0.80
Cactus	2.98	-0.56	2.38	0.01	1.12	-3.24	0.38	0.12
BasketballDrive	4.72	-2.95	3.10	0.01	3.65	-4.46	1.91	0.03
BQTerrace	4.36	1.23	3.84	0.57	1.58	0.75	1.49	0.75
Average	**4.14**	-1.84	**3.08**	**0.22**	**2.28**	-2.23	**1.46**	**0.49**

5.2 HW 구현을 위한 GOP의 picture별 target rate 계산 방법

HM 인코더에는 GOP 구조에 따라 GOP를 구성하는 picture의 model parameter의 값을 반영 [14]하여 GOP내의 picture의 bit 할당을 수행한다. 식 (5-15)는 GOP의 picture별 model parameter값을 반영한 bit 할당을 위한 식을 보여준다. λ_{ratio}의 값은 GOP구조에 따라서 미리 정해진 값으로 표 3-2에서와 같은 값을 가진다. 식 (5-15)에서 i는 GOP를 구성하는 picture를 가리키는 index이다. 그림 5-1은 식 (5-15)에서 model parameter를 반영하게 위해 구해야 하는 값인 λ_{basic}의 해를 찾는 과정을 보여준다.

$$bpp_{target} \approx \frac{1}{GOPsize} \cdot \sum_{GOP} \left(\frac{\lambda_{ratio}[i]}{\alpha[i]}\right)^{\frac{1}{\beta[i]}} \cdot (\lambda_{basic})^{\frac{1}{\beta[i]}} = \frac{1}{GOPsize} \cdot \sum_{GOP} \left(\frac{\lambda_{ratio}[i] \cdot \lambda_{basic}}{\alpha[i]}\right)^{\frac{1}{\beta[i]}} \quad (5-15)$$

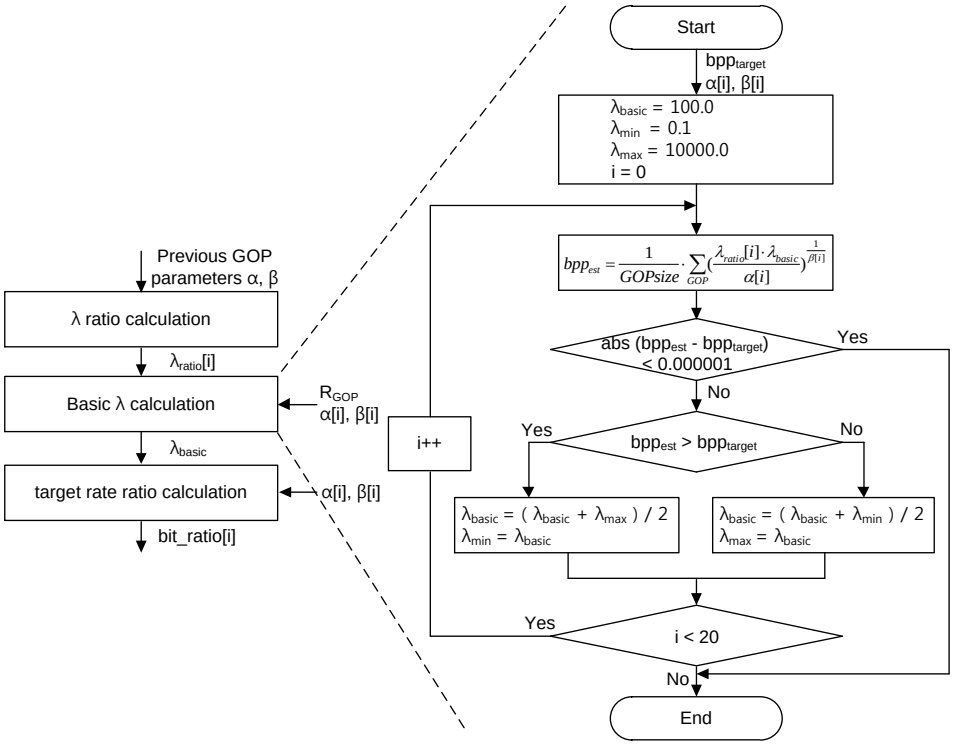


그림 5-1 HM 인코더에 적용된 GOP의 picture별 bit 할당을 위해 λ_{basic}의 해를 찾는 과정

λ_{basic}값과 rate는 그림 5-2와 같이 감소하는 함수 관계를 가진다. HM 인코더에서 λ_{basic}값을 찾는 방법은 그림 5-2에서와 같이 target 값과의 크기 비교를 통해서 현재 값과 min 또는 max와의 평균값을 새로운 solution으로 설정하고 solution을 update하는 방법을 이용한다. 이전 solution은 min값과 평균을 하여서 새로운 solution을 구하였을 경우에 max가 되고 max값과 평균을 하여서 새로운 solution을 구하였을 경우에는 min값이 된다. HM 인코더에 적용된 방법은 그림 5-2의 Sol 2에서 Sol 3으로 바뀌는 것과 같이 iteration이 수행될수록 반드시 해에 가까워 지지는 않기 때문에 최대 20 번까지의 iteration을 지원하는 것으로 보인다. Newton method를 이용하여 해를 찾으면 더 적은 iteration에 해를 찾을 수 있지만 기울기를 구하기 위해서 미분

함수가 필요하고 x절편을 구하는 과정이 필요하다. 따라서 HM 인코더에서는 iteration의 수는 많이 필요하지만 간단한 방법을 이용한 것으로 보인다.

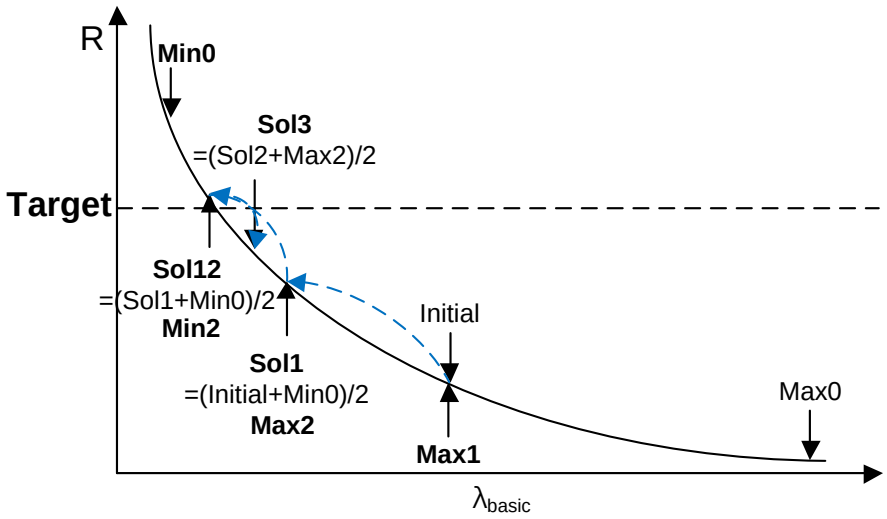


그림 5-2 HM 인코더에 적용된 GOP의 picture별 target rate을 구하기 위한 λ_{basic}의 해를 찾는 과정

HM 인코더에 적용되어 있는 식 (5-15)의 λ_{basic}값을 iteration을 수행하여 해를 구하는 것은 HW구현에서는 적합하지 않다. 제안하는 방법은 picture들의 λ_{ratio}, α, β의 평균값을 이용하여 iteration을 수행하지 않고 값을 구한다. 식 (5-16) (5-17) (5-18)에서와 같이 λ_{ratio}와 α, 는 기하 평균, β는 산술 평균을 이용한다.

$$\log_2 \lambda_{ratio,avg} = \frac{1}{GOPsize} \cdot \sum_{GOP} \log_2 \lambda_{ratio}[i] \qquad (5-16)$$

$$\log_2 \alpha_{avg} = \frac{1}{GOPsize} \cdot \sum_{GOP} \log_2 \alpha[i] \qquad (5-17)$$

$$\beta_{avg} = \frac{1}{GOPsize} \cdot \sum_{GOP} \beta[i] \qquad (5-18)$$

식 (5-19)은 식 (5-16) (5-17) (5-18)의 평균 λ_{ratio}, α, β를 이용한 target rate과 λ_{basic}값의 관계식이다.

$$R_{target} = (\frac{\lambda_{ratio,avg} \cdot \lambda_{basic}}{\alpha_{avg}})^{\frac{1}{\beta_{avg}}} \qquad (5-19)$$

식 (5-19)의 양변이 log를 취하면 식 (5-20)과 같고 이를 $\log \lambda_{basic}$값에 대해 정리하면 식 (5-21)과 같다. 식 (5-21)은 식 (5-16) (5-17) (5-18)을 이용하여 구한 평균 값들과 log를 취한 target rate으로부터 log를 취한 λ_{basic}값을 구할 수 있다.

$$\log_2 R_{target} = \frac{1}{\beta_{avg}} \log_2 (\frac{\lambda_{ratio,avg} \cdot \lambda_{basic}}{\alpha_{avg}}) \qquad (5-20)$$

$$\log_2 \lambda_{basic} = \beta_{avg} \cdot \log_2 R_{target} + \log_2 \alpha_{avg} - \log_2 \lambda_{ratio,avg} \qquad (5-21)$$

식 (5-22)는 picture별 target rate을 구하기 위한 picture별 weight를 λ_{basic}값으로부터 구하는 식이다.

$$\omega[i] = (\frac{\lambda_{ratio}[i] \cdot \lambda_{basic}}{\alpha[i]})^{\frac{1}{\beta[i]}} \qquad (5-22)$$

식 (5-22)에 log를 취하면 식 (5-23)와 같고 식 (5-16) (5-17) (5-18)을 이용하여 구한 평균 값들과 식 (5-21)에서 구한 log를 취한

λ_{basic}값으로부터 log를 취한 picture별 weight의 값을 구할 수 있다. 구해진 log를 취한 weight값에 anti-log를 취하여 최종적인 picture의 weight값을 계산한다.

$$\log_2 \omega[i] = \frac{1}{\beta[i]}(\log \lambda_{ratio}[i] + \log \lambda_{basic} - \log \alpha[i])$$

$$= \frac{1}{\beta[i]}(\beta_{avg} \cdot \log_2 R_{target} + \log_2 \alpha_{avg} - \log_2 \alpha[i] - (\log_2 \lambda_{ratio,avg} - \log_2 \lambda_{ratio}[i])) \quad (5-23)$$

표 5-3은 Target bpp = 0.668229일 경우의 λ_{ratio}, α, β의 평균을 이용하였을 λ_{basic}을 구하였을 경우의 λ_{basic}값 비교이다. HM 인코더의 알고리듬으로 구한 λ_{basic}의 해 보다 평균으로 구한 해가 작음을 볼 수 있다. 평균 λ_{ratio}, α, β를 이용하여 구한 해의 error를 보상해 줘서 더 정확한 값의 해를 구할 수도 있지만 평균으로 λ_{basic}의 값을 계산하여도 표 5-4에서와 같이 코딩 효율의 저하가 발생하지 않기 때문에 평균으로 계산한 λ_{basic}의 값을 그대로 이용하여 HW 구현에 적합하게 식을 수정하였다.

표 5-3 Target bpp = 0.668229일 경우의 λ_{ratio}, α, β의 평균을 이용한 λ_{basic}을 구하였을 경우의 λ_{basic}값 비교

λ_{basic}	HM의 Target bpp	평균을 이용한 Target bpp	
		산술평균 (λ_{ratio}) 산술평균 (α) 산술평균 (β)	기하평균 (λ_{ratio}) 기하평균 (α) 산술평균 (β)
…	…	…	…
10	1.046978	0.910269	0.968081
11	0.976026	0.847733	0.901573
12	0.915464	0.794402	0.844855
13	0.863081	0.748309	0.795835
14	0.817261	0.708022	0.752989
15	0.776796	0.672468	0.715177
16	0.740761	0.640827	0.681526
17	0.708438	0.612462	0.65136
18	0.679258	0.58687	0.624143
19	0.652763	0.563647	0.599445
20	0.628586	0.542465	0.576918
21	0.606421	0.523057	0.556276
…	…	…	…

표 5-4 평균 λ_{ratio}, α, β를 이용하여서 GOP의 λ_{basic}를 계산할 경우의 BD rate (%)

1080p	CTU-level rate control (HM)				평균 λ_{ratio}, α, β를 이용한 λ_{basic} 계산			
	BD rate (%)			Err (%)	BD rate (%)			Err (%)
	Y	UV	6:1:1		Y	UV	6:1:1	
Kimono	6.38	-6.94	4.16	0.47	6.36	-5.52	4.38	0.62
ParkScene	2.27	0.06	1.90	0.04	2.01	1.51	1.93	0.31
Cactus	2.98	-0.56	2.38	0.01	2.42	-0.87	1.86	0.01
BasketballDrive	4.72	-2.95	3.10	0.01	4.42	-2.82	2.90	0.01
BQTerrace	4.36	1.23	3.84	0.57	3.21	2.20	3.06	0.65
Average	**4.14**	-1.84	**3.08**	**0.22**	**3.68**	-1.10	**2.83**	**0.32**

표 5-5는 이번 장에서 제안한 λ_{ratio}를 구하는 방법에 5.1장에서 제안한 log R-log λ를 같이 적용하고 4.3장에서 코딩 효율을 높이기 위해서 제안한 picture의 평균 λ값을 이용하였을 경우의 BD rate을

보여준다. log R-log λ에 평균 평균 λ_{ratio}, α, β를 이용한 λ_{basic} 계산을 적용하였을 때 log R-log λ만을 적용하였을 경우보다 평균 Y BD rate이 0.08 % 증가로 코딩 효율의 약간의 저하가 있다. 4.3장에서 제안한 picture의 평균 λ값을 이용하였을 경우에는 약 0.4 %의 코딩 효율이 추가로 생기는 것을 확인할 수 있고 제안하는 방법들이 각각 코딩 효율을 높이는데 효과가 있음을 보여준다.

표 5-5 log R-log λ를 이용하고 평균 λ_{ratio}, α, β를 이용하여서 GOP의 λ_{basic}를 계산할 경우와 picture의 평균 λ값을 이용하였을 경우의 BD rate (%)

1080p	log R – log λ 이용 ① + 평균 λ_{ratio}, α, β를 이용한 λ_{basic} 계산 ②				① + ② + Picture의 평균 λ값 이용 ③			
	BD rate (%)			Err (%)	BD rate (%)			Err (%)
	Y	UV	6:1:1		Y	UV	6:1:1	
Kimono	4.53	-6.13	2.77	0.85	6.11	-7.65	3.80	1.29
ParkScene	1.16	2.72	1.43	0.68	0.57	-1.22	0.31	0.65
Cactus	1.13	-3.95	0.26	0.18	0.75	-0.73	0.54	0.15
BasketballDrive	3.43	-4.03	1.82	0.02	2.25	-0.95	1.56	0.03
BQTerrace	1.53	2.77	1.79	0.8	0.27	-1.48	0.07	0.98
Average	**2.36**	-1.72	1.61	**0.51**	**1.99**	-2.41	1.25	**0.62**

5.3 HW 구현을 위한 model parameter update

식 (5-24) (5-25)는 각각 log를 취하고 target rate과 실제 발생 rate으로부터 model parameter인 $\log\alpha$와 β의 update 식이다. δ_α와 δ_β는 sequence의 target bpp에 따라서 표 5-6과 같은 값을 가지는 update parameter이다. 이 값들은 1보다 작은 값들로 나눗셈 연산이나 곱셈과 오른쪽 shift로 구현할 수 있다. 코딩 효율의 저하가 없다면 HW 구현을 간단하게 하기 위해 오른쪽 shift만으로 구현할 수 있도록 δ_α와 δ_β의 값을 바꾸는 것이 효율적이다.

$$\log\alpha_{new} = \log\alpha_{old} + \delta_{\alpha} \cdot \beta_{old} \cdot (\log R_{target} - \log R_{real}) \quad (5-24)$$

$$\beta_{new} = \beta_{old} + \delta_{\beta} \cdot \beta_{old} \cdot (\log R_{target} - \log R_{real}) \cdot \log R_{real} \quad (5-25)$$

표 5-6 HM 인코더에서 사용하는 sequence의 target bpp에 따른 update parameter의 값

Sequence의 target bpp	δ_{α}	δ_{β}
bpp < 0.03	0.01	0.005
0.03 ≤ bpp < 0.08	0.05	0.025
0.08 ≤ bpp < 0.2	0.1	0.05
0.2 ≤ bpp < 0.5	0.2	0.1
0.5 ≤ bpp	0.4	0.2

표 5-7은 δ_{α} 와 δ_{β} 값을 HW 구현에 적합하도록 오른쪽 shift만으로 나눗셈이 구현 가능하도록 수정한 두 가지 set을 보여준다. A set은 원래의 값보다 가까운 작은 값으로 바꾼 값이고 B set은 원래의 값보다 가까운 큰 값으로 바꾼 값이다. 표 5-8은 표 5-7의 δ_{α} 와 δ_{β} 값의 두 가지 set의 값을 이용하여 인코딩 하였을 경우의 BD rate을 보여준다. 이 논문에서는 A set을 이용하였을 때 B set을 이용하였을 경우보다 Y BD rate이 더 작기 때문에 A set의 δ_{α} 와 δ_{β} 값을 이용하였다.

표 5-7 HW 구현에 적합한 sequence의 target bpp에 따른 update parameter의 값의 두 가지 set

Sequence의 target bpp	A set		B set	
	δ_{α}	δ_{β}	δ_{α}	δ_{β}
bpp < 0.03	1/128	1/256	1/64	1/128
0.03 ≤ bpp < 0.08	1/32	1/64	1/16	1/32
0.08 ≤ bpp < 0.2	1/16	1/32	1/8	1/16
0.2 ≤ bpp < 0.5	1/8	1/16	1/4	1/8
0.5 ≤ bpp	1/4	1/8	1/2	1/4

표 5-8 제안하는 두 가지 update parameter set을 이용하여 rate control을 하였을 경우의 BD rate (%)

1080p	A set				B set			
	BD rate (%)			Err (%)	BD rate (%)			Err (%)
	Y	UV	6:1:1		Y	UV	6:1:1	
Kimono	6.56	−6.67	4.33	0.51	6.56	−6.55	4.37	0.50
ParkScene	1.98	0.76	1.79	0.01	2.09	−0.84	1.59	0.02
Cactus	3.17	−0.44	2.56	0.02	3.81	0.72	3.29	0.02
BasketballDrive	4.83	−2.57	3.27	0.01	4.73	−3.13	3.08	0.01
BQTerrace	4.56	2.04	4.16	0.59	4.46	0.19	3.74	0.55
Average	**4.22**	−1.38	**3.22**	**0.23**	**4.33**	−1.92	**3.21**	**0.22**

5.4 Rate control의 HW 구현을 위한 fixed point 연산과 LUT 사용

HM 인코더에 구현된 rate control 알고리듬은 floating point를 이용하는 변수를 이용하고 HW 구현에 적합하지 않는 지수 연산을 이용한다. 수행 cycle이 빠르고 HW 복잡도가 작은 rate control HW를 구현하기 위해서는 rate control에서 사용하는 변수들을 fixed point로 사용하고 bit width 최적화를 시켜야 한다. 그리고 5.1장에서 log를 취한 log R−log λ model을 사용함으로써 log와 anti−log연산으로 변환한 지수연산을 LUT를 이용하여서 수행한다. 또한 target bit을 구할 때 사용하는 division 연산도 LUT를 이용하여 HW 복잡도를 줄인다.

표 5-9는 rate control에서 이용하는 변수들의 fixed point로 표현하였을 때 지원한 bit width를 보여준다. Total target rate은 32 bit 정수형을 이용하였고 total target rate로부터 CTU 단위로 bit allocation을 수행할 때 사용하는 weight ω와 rate R은 16 bit 정수형을 이용하였다. log R−log λ model에서 사용하는 model parameter인 $\log\alpha$와 β는 정수 부분과 소수 부분의 bit을 지원한다. $\log\alpha$는 정수 부분 4 bit과 소수 부분 3 bit을 포함하고 β는 정수 부분

3 bit과 소수 부분 6 bit을 포함한다. Rate control 과정에서 사용하는 log R인 log bpp값은 식 (5-26)의 bpp의 계산식에 log를 취하여 식 (5-27)의 log R과 log (# of pixels)의 차로 계산한다. log bpp는 정수 6 bit과 소수 3 bit를 포함한다. CTU 별 target bit 할당을 위해 CTU 별 weight ω를 구할 때 log ω를 구하고 anti-log를 취하여 weight를 계산한다. 계산에서 사용한 log ω는 정수 부분 6 bit과 소수 부분 7 bit을 포함한다.

표 5-9 Rate control에서 이용하는 변수들의 fixed point로 표현하였을 때 지원한 bit width

	정수 부분 bit width	소수 부분 bit width	총 bit width
Total R	32	0	32
$\log_2 \alpha$	4	3	7
β	3	6	9
rate R	16	0	16
$\log_2$bpp	6	3	9
weight ω	16	0	16
$\log_2 \omega$	6	7	13

$$bpp = \frac{R}{\#of\ pixels} \tag{5-26}$$

$$\log_2 bpp = \log_2 R - \log_2(\#of\ pixels) \tag{5-27}$$

Rate control에서 log R-log λ model을 이용하여 CTU 별 bit 할당을 수행할 때 log와 anti-log의 함수가 필요하다. $\log_2$와 anti-$\log_2$는 [29] 논문에서와 같이 LUT을 이용하고 leading zero detector를 이용하여 구현하였다. 그림 5-3 (a)는 정수 domain의 값을 $\log_2$ LUT를 이용하여 log를 취하는 과정을 보여주고 그림 5-3 (b)는 log domain의 값을 anti-log LUT를 이용하여 다시 정수 domain으로 바꾸어 주는 과정을 보여준다.

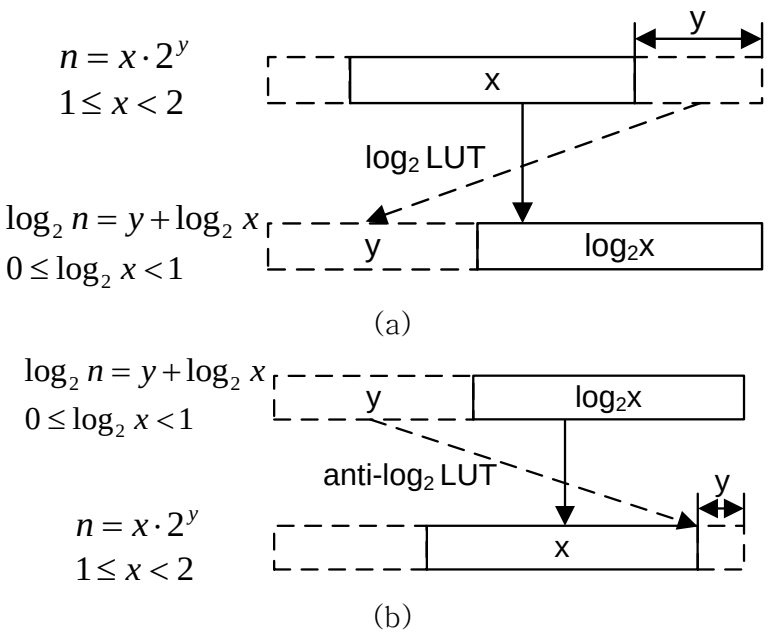
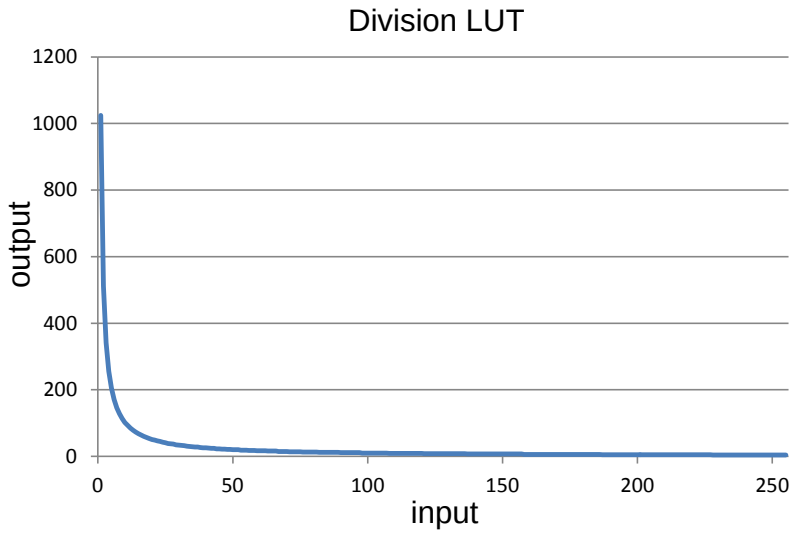


그림 5-3 LUT와 leading zero detector를 이용한 log와 anti-log 연산(a) log LUT (b) anti-log LUT

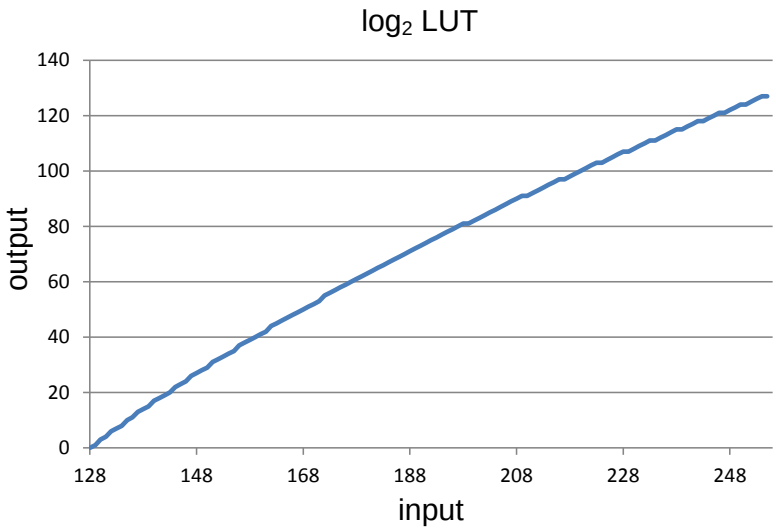
표 5-10은 rate control에서 사용한 division, log, anti-log LUT의 각 LUT의 element의 수와 element의 bit width를 보여준다. 그림 5-4는 각 LUT의 input과 output의 값의 그래프를 보여준다. 그림 5-4 (a)의 division에 대한 LUT의 값은 input값이 증가함에 따라서 급격히 작아지는 것을 볼 수 있다. 그림 5-4 (b) (c)의 log, anti-log LUT의 값은 input값이 증가함에 따라서 증가하는 정도가 급격하지 않음을 볼 수 있다. 따라서 division에 대한 LUT가 log와 anti-log LUT 보다 급격한 값의 변화를 가지기 때문에 log와 anti-log LUT 보다 LUT의 element의 더 많고 element의 bit width를 더 지원하여야 비슷한 정확도를 보일 수 있다.

표 5-10 Rate control에서 사용하는 LUT들의 element의 수와 element의 bit width

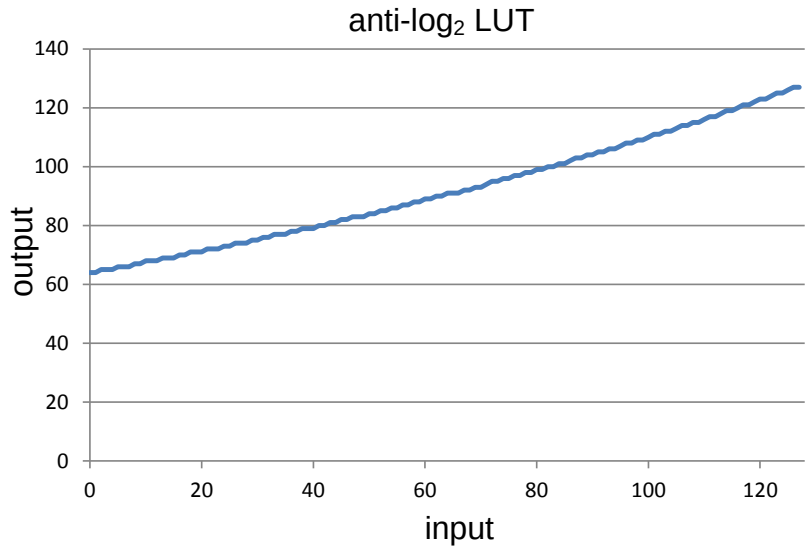
	# of elements	bit width of elements
Division	256	11
$\log_2$ LUT	128	7
anti-$\log_2$ LUT	128	7



(a)



(b)



(c)

그림 5-4 Rate control에서 사용하는 LUT의 input과 output 값
(a) division LUT (b) log LUT (c) anti-log LUT

표 5-11은 rate control에서 사용하는 변수들을 fixed point로 바꾸고 division, log, anti-log를 LUT를 이용하여 수행하였을 경우의 BD rate을 보여준다. 평균적으로 코딩 효율을 높이고 HW를 고려한 5.1장, 5.2장과 4.3장에서 적용한 방법의 코딩 효율보다 알고리듬을 fixed point로 바꾸고 LUT를 적용하면서 floating point를 이용한 경우보다 약 1.7 %의 Y-BD rate의 증가를 보였다.

표 5-11 Rate control에서 사용하는 변수들을 fixed point로 바꾸고 division, log, anti-log를 LUT를 이용하여 수행하였을 경우의 BD rate (%)

1080p	① log R – log λ 이용 + ② 평균 λ_{ratio}, α, β를 이용한 λ_{basic} 계산 + ③ Picture의 평균 λ값 이용				① + ② + ③ + 변수들의 fixed point 사용 division, log, anti-log의 LUT 사용			
	BD rate (%)			Err (%)	BD rate (%)			Err (%)
	Y	UV	6:1:1		Y	UV	6:1:1	
Kimono	6.11	-7.65	3.80	1.29	9.06	-4.10	6.83	1.36
ParkScene	0.57	-1.22	0.31	0.65	1.99	-3.78	0.97	0.50
Cactus	0.75	-0.73	0.54	0.15	1.92	-6.90	0.32	0.21
BasketballDrive	2.25	-0.95	1.56	0.03	3.18	-5.71	1.27	0.15
BQTerrace	0.27	-1.48	0.07	0.98	2.27	-4.42	1.14	0.11
Average	**1.99**	-2.41	1.25	**0.62**	**3.68**	-4.98	2.11	**0.47**

5.5 Rate control의 HW 구현과 HW 인코더에서의 rate control의 동작

표 5-12는 구현한 rate control HW의 complexity를 보여준다. Rate control을 4개로 구분하여 구현하였으며 각 부분은 GOP, picture, CTU에 해당하는 연산을 모두 처리할 수 있도록 구현하였다. 총 27.5 Kgate의 복잡도를 보이며 target rate을 구하는 부분이 12.1 Kgate로 가장 복잡도가 높았으며 이는 total rate을 저장하고 weight의 합을 계산한 bit width가 큰 변수에 대해 나눗셈 연산을 수행하기 위해 LUT을 이용한 곱셈을 수행하기 때문이다. 이는 H.264의 reference SW인 JM 인코더의 rate control 구현 [18]의 144 Kgate의 20 %의 복잡도에 해당한다. JM 인코더의 rate control 방법은 이전 인코딩된 정보들로부터 선형 회귀를 수행하여 quadratic model의 model parameter를 계산하기 때문에 곱셈과 나눗셈 연산을 많이 수행하여 복잡도가 높다.

표 5-12 구현한 rate control HW의 complexity

Category	Complexity (gates)	
	log R – log λ	JM [18]
Weight calculation	5.8 K	–
Target Rate calculation	10.1 K	–
QP calculation	3.8 K	–
Model parameter Update	7.8 K	–
Total	**27.5 K**	**144 K**

식 (5-28)은 rate control HW가 4K 크기의 이미지의 picture level별로 CTU 별로 model parameter와 계산된 weight를 저장하기 위한 memory 크기를 보여준다. RA configuration을 인코딩할 경우에 4개의 picture level을 가지고 2048개의 CTU를 가지고 있다. log R-log λ model에서 사용하는 model parameter인 logα와 β는 각각 7 bit과 9 bit을 가지고 weight는 16 bit을 가져서 하나의 CTU당 4 byte의 memory를 필요로 한다. 따라서 4K 크기의 이미지를 RA configuration으로 인코딩할 경우 CTU-level rate control을 수행하기 위해서 32 KB의 memory가 필요하다. 32 KB의 memory는 SRAM으로 저장하거나 외부 SDRAM에 저장할 수 있다. SDRAM에 저장할 경우에는 버퍼를 두어 한 번에 SDRAM의 burst transaction을 효율적으로 이용하여 데이터를 읽고 쓸 때 발생하는 latency를 줄일 수 있다.

필요한 Memory 크기 =
Picture level (4) x CTU 수 (2048)
x 4 byte (logα (7 bit), β (9 bit), weight (16 bit))
= 32 KB (5-28)

표 5-13은 구현한 rate control HW의 GOP, Picture, CTU의 각 level별 수행 cycle과 사용 수식 정리한 것을 보여준다. Weight

calculation을 수행은 weight를 구하는 picture 수와 CTU 수에 비례하여 cycle이 소모된다. 나머지 연산은 2~4 cycle의 수행 cycle이 걸린다.

표 5-13 구현한 rate control HW의 GOP, Picture, CTU의 각 level별 수행 cycle과 사용 수식 정리

Level	Category	Cycles	사용 수식
GOP	Target rate calculation	4	$R_{GOP} = R_{Avg}^{Pic} \cdot N_{GOP}$ $R_{Avg}^{Pic} = \frac{R_{Total}}{N_{Total}^{Pic}} + \frac{1}{SW} \cdot (R_{Left} - N_{Left}^{Pic} \cdot \frac{R_{Total}}{N_{Total}^{Pic}}) - M \cdot \frac{N_{Left}^{Pic}}{N_{Total}^{Pic}} \cdot \frac{R_{Total}}{N_{Total}^{Pic}}$
Picture	Weight calculation for each picture	GOPsize + 6	$\log_2 \lambda_{ratio,avg} = \frac{1}{GOPsize} \cdot \sum_{GOP} \log_2 \lambda_{ratio}[i]$ $\log_2 \alpha_{avg} = \frac{1}{GOPsize} \cdot \sum_{GOP} \log_2 \alpha[i]$ $\beta_{avg} = \frac{1}{GOPsize} \cdot \sum_{GOP} \beta[i]$ $\log_2 \omega[i] = \frac{1}{\beta[i]} (\log_2 \lambda_{ratio}[i] + \log_2 \lambda_{basic} - \log_2 \alpha[i])$ $= \frac{1}{\beta[i]} (\beta_{avg} \cdot \log_2 R_{target} + \log_2 \alpha_{avg} - \log_2 \alpha[i]$ $- (\log_2 \lambda_{ratio,avg} - \log_2 \lambda_{ratio}[i]))$ $\omega_{total} = \sum_{Pic} \omega[i]$
	Target rate calculation	2	$R_{Init}^{Pic}[i] = R_{GOP} \cdot \frac{\omega[i]}{\sum_{GOP} \omega[i]}$ $R_{Left}^{Pic}[i] = R_{Left} \cdot \frac{\omega[i]}{\sum_{Left} \omega[i]}$ $R_{Left} = R_{GOP} - R_{Coded}$ $R_{Target}^{Pic}[i] = (1 - \frac{1}{SW}) \cdot R_{Init}^{Pic}[i] + \frac{1}{SW} \cdot R_{Left}^{Pic}[i]$
	QP calculation	2	$QP = 3\beta \cdot \log_2 R + 3 \cdot \log_2 \alpha - 3 \cdot \log_2 \lambda_n + 4$
	Model Parameter Update	3	$\log_2 \alpha_{new} = \log_2 \alpha_{old} + \delta_\alpha \cdot \beta_{old} \cdot (\log_2 R_{target} - \log_2 R_{real})$ $\beta_{new} = \beta_{old} + \delta_\beta \cdot \beta_{old} \cdot (\log_2 R_{target} - \log R_{real}) \cdot \log_2 R_{real}$
CTU	Weight calculation for each CTU	# of CTU + 6	$\log_2 \omega[i] = \frac{1}{\beta[i]} (\beta_{Pic} \log_2 R_{Pic} + \log_2 \alpha_{Pic} - \log_2 \alpha[i])$ $\omega_{total} = \sum_{Pic} \omega[i]$

	Target rate calculation	2	$R_{Init}^{CTU}[i] = R_{Pic} \cdot \frac{\omega[i]}{\sum_{Pic}\omega[i]}$ $R_{Left}^{CTU}[i] = R_{Left} \cdot \frac{\omega[i]}{\sum_{Left}\omega[i]}$ $R_{Left} = R_{Pic} - R_{Coded}$ $R_{Target}^{CTU}[i] = (1-\frac{1}{SW}) \cdot R_{Init}^{CTU}[i] + \frac{1}{SW} \cdot R_{Left}^{CTU}[i]$
	QP calculation	2	$QP = 3\beta \cdot \log_2 R + 3 \cdot \log_2 \alpha - 3 \cdot \log_2 \lambda_n + 4$
	Model parameter update	3	$\log_2 \alpha_{new} = \log_2 \alpha_{old} + \delta_\alpha \cdot \beta_{old} \cdot (\log_2 R_{target} - \log_2 R_{real})$ $\beta_{new} = \beta_{old} + \delta_\beta \cdot \beta_{old} \cdot (\log_2 R_{target} - \log R_{real}) \cdot \log_2 R_{real}$

그림 5-5는 HW 인코더의 인코딩 과정에서 rate control HW의 수행과정을 보여준다. CTU의 target rate을 구하는 과정과 QP를 구하는 과정은 이전 CTU의 발생 rate을 반영하여 구해야 하기 때문에 CTU 인코딩과 병렬적으로 수행될 수 없어서 인코딩 과정의 overhead가 된다. Model parameter update는 update한 model parameter를 다음 같은 level의 picture의 같은 위치의 CTU가 이용하기 때문에 수행 cycle을 CTU encoding 과정과 hiding 시켜 전체 수행 cycle에 영향을 주지 않는다.

표 5-14는 RA configuration의 GOPsize = 8인 하나의 GOP의 구현한 rate control HW의 수행 cycle을 보여준다. 8장의 picture를 수행하는데 총 82016 cycle의 overhead가 발생한다. GOP와 picture level의 rate control의 연산을 무시하고 CTU level의 연산만을 고려하면 CTU 당 4 cycle의 overhead가 발생하고 4K 30fps를 400 MHz로 수행할 경우 CTU당 cycle budget인 6510 cycle의 0.06 %에 해당하는 값으로 무시할 수 있는 수준이다.

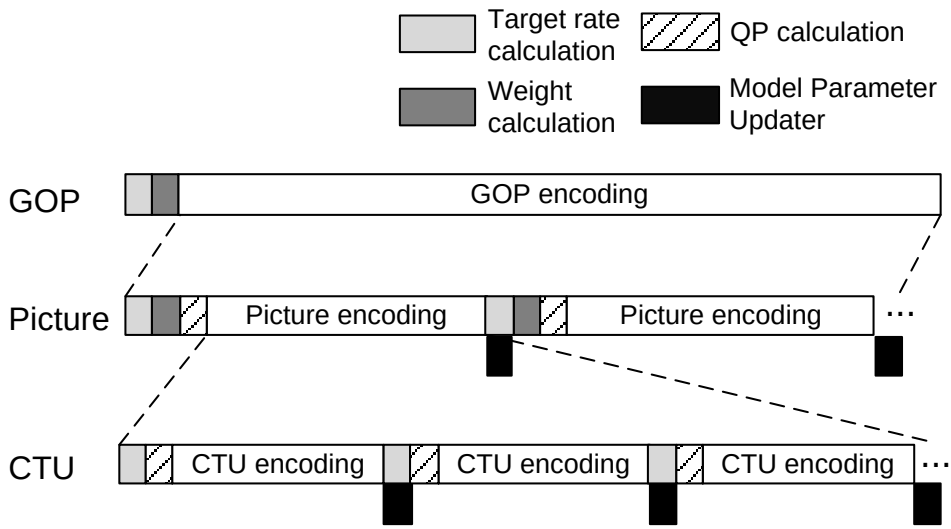


그림 5-5 HW 인코더의 인코딩 과정에서 rate control HW의 수행

표 5-14. RA configuration의 GOPsize = 8인 하나의 GOP의 rate control을 위한 수행 cycle

Level	Category	Cycles for a GOP of RA configuration
GOP	Target Rate Calculation	4
Picture	Weight calculation for each picture	8 + 6
	Target Rate Calculation	8 x 2
	QP calculation	8 x 2
	Model Parameter Update	0 (Hiding)
CTU	Weight calculation for each CTU	(2048 + 6) x 8
	Target Rate Calculation	2048 x 8 x 2
	QP calculation	2048 x 8 x 2
	Model Parameter Update	0 (Hiding)
Total (cycles)		82016 cycles / GOP

제 6 장 결 론

HEVC 표준은 2013년 4월에 확정되었고 기존의 비디오 표준에 비해 두 배 정도의 압축률을 보이기 때문에 시장에서 이미 많은 제품들이 지원하고 있고 앞으로 더 널리 사용될 것으로 예상된다. HEVC reference SW인 HM 인코더에는 R-λ model을 이용한 새로운 rate control 알고리듬이 적용되어 rate control을 이용하여 인코딩을 수행할 때 기존의 rate control 알고리듬에 비해 코딩 효율을 크게 향상 시켰다. 이 논문에서는 HM 인코더에 적용되어 있는 rate control 알고리듬의 코딩 효율을 개선하고 실시간 HW 인코더에서 이용할 수 있도록 HW 구현에 적합하게 수정한 후, 수정한 rate control 알고리듬을 HW로 구현하였다.

HM 인코더의 rate control 알고리듬은 이미지 시퀀스에 따라서 이미지 시퀀스 후반부에 bit rate이 부족한 현상으로 PSNR이 급격히 떨어지는 현상이 발생할 수 있다. 이 논문에서는 이를 개선하고자 rate control 알고리듬의 bit 할당을 전체 이미지 시퀀스에서 target bit allocation을 이미지 초반부에 bit을 덜 할당하였다. 이는 이미지 시퀀스 후반부에 bit rate이 부족하여 PSNR이 떨어지는 현상을 완화시켜 이미지 시퀀스 전체의 picture에 대한 minimum PSNR을 평균 0.11 dB 향상 시켰고 ParkScene 이미지 시퀀스에서 최대 1.58 dB까지 PSNR 향상이 있었다.

HM 인코더의 CTU-level rate control의 코딩 효율을 향상 시키기 위해서 CTU의 계산된 λ가 아닌 picture의 평균 λ를 이용하여 인코딩을 수행하여 4.14 %의 Y-BD rate을 3.11 %로 감소시켰다. 그리고 가정한 HW 인코더 구조의 transform & full RDO &

reconstruction stage의 RDO를 위해서 사용하는 full RD cost 계산의 HW 복잡도를 줄이기 위해서 quantization step (Q_{step})의 제곱으로 나눈 normalized full RD cost를 제안하고 사용하였다.

HM 인코더의 $R-\lambda$ model을 이용한 rate control의 floating point와 지수 연산을 이용하여 HW 복잡도를 줄여 구현하기 적합하지 않다. 이 논문에서는 $R-\lambda$ model에 log를 취한 $\log R - \log\lambda$ model을 이용하여 지수 연산을 없앴다. $\log R - \log\lambda$ model을 이용하며 model parameter의 update가 더 정확해져 평균 Y-BD rate이 1.99 %로 감소하여 코딩 효율의 개선이 있었다. 그리고 floating point를 이용한 변수들을 fixed point로 수정하고 LUT을 이용하여 log, anti-log, division 연산을 HW 구현이 간단하게 수정하였다. 구현한 rate control의 HW는 GOP, picture, CTU level을 모두 지원하고 27.5 kgate의 복잡도를 가지며 4K 이미지 시퀀스 RA configuration를 지원하기 위해 CTU 별 model parameter와 계산한 weight를 저장하기 위한 32 KB의 메모리가 필요하다. 구현한 rate control HW의 수행 cycle은 CTU당 4 cycle로 이는 400 MHz에서 4K 30 fps를 수행하였을 경우에 약 0.06 %의 overhead이다.

참고 문헌

[1] G. J. Sullivan, J.-R. Ohm, W.-J. Han and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," IEEE Trans. Circuits Syst. Video Technol., vol. 22, no. 12, pp.1649 -1668, 2012

[2] K. McCann, B. Bross, I. K. Kim, K. Sugimoto, and W. J. Han, "HM6:High Efficiency Video Coding (HEVC) Test Model 6 Encoder Description," Doc. JCTVC-H1002, 8th JCTVC Meeting, San Jose, CA, Feb. 2012.

[3] T.-C. Chen, Y.-W. Huang, and L.-G. Chen, "Analysis and design of macroblock pipelining for H. 264/AVC VLSI architecture," in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'04), pp. II273-II276, 2004.

[4] S.-F. Tsai, C.-T. Li, H.-H. Chen, P.-K. Tsung, K.-Y. Chen, and L.-G. Chen, "A 1062Mpixels/s 8192× 4320p High Efficiency Video Coding (H. 265) encoder chip," Symp. VLSI Circuits (VLSIC'13), pp. C188-C189, 2013.

[5] X. Bao, D. Zhou, P. Liu, and S. Goto, "An advanced hierarchical motion estimation scheme with lossless frame recompression and early-level termination for beyond high-definition video coding," IEEE Trans. Multimedia, vol. 14, no. 2, pp. 237-249, 2012.

[6] S. Lee, S. Park, and J. Park, "270 MHz Full HD H. 264/AVC High Profile Encoder with Shared Multibank Memory-Based Fast Motion Estimation," ETRI journal, vol. 31, no. 6, pp. 784-794, 2009.

[7] Z. Liu, Y. Song, M. Shao, S. Li, L. Li, S. Ishiwata, et al., " H. 264/AVC encoder chip design and performance analysis," IEEE journal Solid-

State Circuits, vol. 44, no. 2, pp. 594-608, 2009.

[8] X. Tian, T. M. Le, X. Jiang, and Y. Lian, "Full RDO-support power-aware CABAC encoder with efficient context access," IEEE Trans. Circuits Syst. Video Technol., vol. 19, no. 9, pp. 1262-1273, 2009

[9] S. Deshpande, M. M. Hannuksela, K. Kazui, and T. Schierl, "An improved hypothetical reference decoder for HEVC," in IS&T/SPIE Electronic Imaging, pp. 866608-866608-9, 2013

[10] H. Choi, J. Nam, J. Yoo, D. Sim, and I.V. Bajić, "Rate Control Based on Unified RQ Model for HEVC," Doc. JCTVC-H0213, 8th JCTVC Meeting, San Jose, CA, USA, Feb. 2012.

[11] T. Chiang and Y.-Q. Zhang, "A new rate control scheme using quadratic rate distortion model," IEEE Trans. Circuits Syst. Video Technol., vol. 7, pp. 246-250, 1997.

[12] B. Li, H. Li, L. Li, and J. Zhang, "λ Domain Rate Control Algorithm for High Efficiency Video Coding," IEEE Trans. Image Process., Sep. 2014.

[13] G. J. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," IEEE Signal Process. Mag., vol. 15, no. 6, pp. 74-90, Nov. 1998.

[14] B. Li, H. Li, and L. Li, "Adaptive bit allocation for R-Lambda model rate control in HM," Doc. JCTVC-M0036, 13th JCTVC Meeting, Incheon, Korea, Apr. 2013

[15] C.-H. Kuo, L.-C. Chang, K.-W. Fan, and B.-D. Liu, "Hardware/Software codesign of a low-cost rate control scheme for

H. 264/AVC," IEEE Trans. Circuits Syst. Video Technol., vol. 20, pp. 250-261, 2010.

[16] J. C. Tsai, "Rate Control for Low-Delay Video Using a Dynamic Rate Table," IEEE Trans. Circuits Syst. Video Technol., 2005.

[17] Pan. F, K. P. Lim, "Adaptive basic unit layer rate control for JVT, JVT-G012," JVT of ISO/IEC MPEG & ITU-T, 7th Meeting, Pattaya II, Tailand, 2003

[18] J. W. Gu, "A high quality hardwired rate controller for H.264/AVC real-time video encoding," M.S. thesis, Nat. Tsing Hua Univ., Hsinchu, Taiwan, Jul. 2007.

[19] F. Bossen, Common Test Conditions and Software Reference Configurations, document JCTVC-H1100, JCT-VC, San Jose, CA, Feb. 2012.

[20] V. Sze, M. Budagavi, and G. J. Sullivan, "High Efficiency Video Coding (HEVC)," Springer, 2014.

[21] C. Zhu, H. Jia, S. Zhang, X. Huang, D. Xie, and W. Gao, "On a Highly Efficient RDO-Based Mode Decision Pipeline Design for AVS," IEEE Transactions on Multimedia, vol. 15, no. 8, pp. 1815-1829, 2013

[22] T.-C. Chen, C.-J. Lian, and L.-G. Chen, "Hardware architecture design of an H. 264/AVC video codec," Asia and South Pacific Conference on Design Automation, pp. 750-757, 2006

[23] G. Bjøntegaard, "Calculation of average PSNR differences between RD curves," Doc. VCEG-M33, 13th ITU-T VCEG Meeting, Austin, TX, USA, Apr. 2001

[24] M. Karczewicz, and X. Wang, “Intra Frame Rate Control Based on SATD,” JCTVC-M0257, 13th JCTVC Meeting, Incheon, Korea, Apr. 2013

[25] S. Milani, G. A. Mian, D. Alfonso, and L. Celetto, "A (ρ, Eq)-Domain Based Low-Cost Rate-Control Algorithm for the H. 264 Video Coder," Proc. WPMC 2004, pp. 137-142, 2004.

[26] F. Xiao, "DCT-based video quality evaluation," Final Project for EE392J, vol. 769, 2000.

[27] H.-M. Hu, B. Li, W. Lin, W. Li, and M.-T. Sun, “Region-based rate control for H. 264/AVC for low bit-rate applications,” IEEE Trans. Circuits Syst. Video Technol., vol. 22, pp. 1564-1576, 2012.

[28] L.-M. Po and K. Guo, “Transform-domain fast sum of the squared difference computation for H. 264/AVC rate-distortion optimization,” IEEE Trans. Circuits Syst. Video Technol., vol. 17, no. 6, pp. 765-773, 2007.

[29] T. A. Brubaker and J. C. Becker, "Multiplication using logarithms implemented with read-only memory," IEEE Transactions on Computers, vol. 100, pp. 761-765, 1975.

Abstract

In the HM encoder, the rate control algorithm in the coding tree unit (CTU)-level using a R-λ model shows a loss of the coding efficiency, corresponding to about 4.14 % Bjøntegaard-delta rate (BD rate) increase, compared to that when rate control is off. And the rate control algorithm employed in the HM encoder is not suitable to HW implementation because of its usage of floating point variables. In this paper, therefore an improved rate control algorithm is presented that increases the coding efficiency of the HEVC encoder. In addition, the rate control algorithm is modified for easy HW implementation. There are four contributions in this thesis: improved picture-level bit allocation, normalized full RD cost suitable for HW implementation and rate control model using log R-log λ model, and HW implementation of the proposed algorithms.

Peak signal-to-noise ratios (PSNRs) for the frames in the later part of the image sequence sometimes dramatically decrease due to bit rate starvation when the rate control is enabled in the HM encoder. To alleviate this problem, a rate control algorithm that allocates fewer bits for the frames in the earlier part of the sequence and more bits for those in the later part.

In this thesis, we assume that rate distortion optimization (RDO) is performed in a pipeline stage for transform & full RDO & reconstruction. In the pipeline stage, two methods for evaluating the full RD costs are adopted for easy HW implementation. First, for the RDO search of each CTU, the Lagrangian multiplier λ averaged for the current picture is employed instead of that estimated for the CTU, which also a little bit increases the coding efficiency of the rate control. Second, the normalized full RD cost, that is, the original full RD cost divided by the square of quantization step (Q_{step}), is employed, which substantially reduces the dynamic range of full RD cost.

The R-λ model used in the HM encoder is not suitable for HW implementation because it requires exponent operations. Therefore, another model using logR-logλ relation, which converts exponent operations into linear operations. Consequently, the coding efficiency increases a little because model parameter updates used of the HM encoder is an approximation of log-domain model parameter updates. Because

the variables related to rates should support both log-domain and real-domain, $\log_2$ and anti-$\log_2$ operations using look-up tables (LUTs) are also implemented for supporting transformation between log-domain and real-domain. Furthermore, division operation is also replaced with LUT and right shift for reducing the HW complexity.

Five 1080p image sequences such as Kimono, ParkScene, Cactus, BasketballDrive, and BQTerrace are used to show the efficiency of the proposed rate control for the Random access (RA) configuration of common test condition, assuming that maximum transform unit (TU) depth is equal to 1. Target rates for rate control algorithms are determined from the rates encoded with QPs equal to 22, 27, 32, and 37 without the rate control. The proposed rate control algorithm reduces the Y-BD rate from 4.14 % to 1.99 % compared to the CTU-level rate control algorithm in the HM encoder. Furthermore, the PSNR decrease of the frames in the later part of the input sequence is alleviated so that the minimum PSNR is increased to 0.11 dB on the average. The rate control algorithm implemented in HW supports the rate control in the GOP, picture, and CTU levels, which include 27.5 Kgates and 32 KB memory. The operation cycles of the rate control HW is 4 cycles per CTU, which is 0.06 % overhead, assuming that 4K 30 fps is operated at 400 MHz.

Keywords: high efficiency video coding (HEVC), rate control, hardware implementation, full rate distortion (RD) cost
Student Number: 2007-23058