HEVC에서 인코더 계산 복잡도 개선 및 분할 정보 부호화 방법

이 한 수^{a)}, 김 경 용^{a)}, 김 태 룡^{a)}, 박 광 훈^{a)*}, 김 휘 용^{b)}, 임 성 창^{b)}, 이 진 호^{b)}

Improving Encoder Complexity and Coding Method of the Split Information in HEVC

Han Soo Lee^{a)}, Kyung Yong Kim^{a)}, Tae Ryong Kim^{a)}, Gwang Hoon Park^{a)*}, Hui Yong Kim^{b)},
Sung Chang Lim^{b)}, and Jin Ho Lee^{b)}

요 약

본 논문에서는 참조 프레임 혹은 시간적으로 이전에 부호화한 프레임을 통해 현재 프레임의 LCU 분할구조를 예측하여 부호화하는 방법을 제안한다. HEVC에서는 CU로 부호화 및 복호화를 수행하는데, CU의 기본이 되는 LCU 단위로 영상의 특성에 따라 분할구조를 결정하여 영상을 적응적으로 부호화한다. 이 때, 현재 부호화하려는 LCU의 분할구조와 참조 프레임 및 시간적으로 이전에 부호화한 프레임 내의 동일한 위치에 대응되는 LCU(Co-located LCU)의 분할구조는 매우 유사한 특성이 있다. 따라서 본 논문에서는 인코더의 복잡도를 낮추기 위하여 현재 LCU의 분할구조를 결정할 때, Co-located LCU의 복잡성을 통해 현재 부호화하는 LCU의 분할구조 정보를 예측하고 분할구조에 포함될 확률이 높은 CU만 부호화하는 방법을 제안한다. 제안 방법의 시뮬레이션 결과로서, 인코더만을 변경하여 인코더 복잡도를 낮추는 방법이 기존 대비 인코더 복잡도가 평균 21.3% 감소하였고, 디코더 복잡도는 거의 비슷했으며, BD-Bitrate는 최대 0.6% 증가하였다. 또한 인코더에서 분할구조를 결정할 때 LCU의 분할 정보를 예측하여 부호화하고, CU 분할 정보를 부호화 및 복호화하는 과정을 변경하는 방법을 통해 BD-Bitrate를 감소시키는 방법을 제안하였다. 제안 방법의 시뮬레이션 결과는 인코더 복잡도가 평균 22% 감소하였고, 디코더 복잡도는 거의 비슷했으며, BD-Bitrate는 최대 0.3% 정도만 증가하여 제안하는 방법의 우수함을 확인할 수 있었다.

Abstract

This paper proposes the coding method to predict the split structure of LCU in the current frame on the basis of the reference frame or temporally-previous frame. HEVC encoder determines split structure according to image characteristics in LCU which is an basic element of CU. The split structure of the current LCU is very similar to the split structure of collocated LCU in the reference frame or temporally-previous frame. Thus, this paper proposes the method to reduce the encoder computational complexity by predicting split structure of the current LCU on the basis of that of collocated LCU in the reference frame or temporally-previous frame. And it also proposes the method to reduce the BD-Bitrate by coding after the prediction of the CU split information. The simulation results of changing only encoder showed that the mean of encoder computational complexity was lower by 21.3%, the decoder computational complexity was negligible change and the BD-Bitrate increase by the maximum of 0.6%. Also, the method changing encoder, bitstream, and decoder improves the mean of encoder computational complexity was lower by 22%, the decoder computational complexity was negligible change and the BD-Bitrate is improved to the maximum of 0.3%. When compared with the conventional method, indicating that the proposed method is superior.

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

1. 서론

오늘날 멀티미디어 분야에서 동영상상을 부호화하는 기술은 ISO/IEC SC29 WG11 (MPEG)과 ITU-T SG16/Q.6 (VCEG)의 주도로 JVT(Joint Video Team)를 구성하여 제정한 표준인 H.264/AVC(H.264/MPEG-4 Part 10 Advanced Video Coding)^[1]가 대표적이다. H.264/AVC는 높은 효율을 보이는 다수의 알고리즘을 적용하여 기존의 코덱보다 높은 부호화 성능을 달성함으로써 다양한 분야에 널리 쓰이고 있다. 그리고 최근에는 현재 상용화되고 있는 HDTV와 같이 고해상도, 고화질 영상에 효율적인 압축 표준을 제정하기 위해서 MPEG과 VCEG으로 구성된 JCT-VC(Joint Collaborative Team on Video Coding)를 구성하였고, 해당 그룹에서 차세대 비디오 코덱 기술인 HEVC(High Efficiency Video Coding)^[2]라는 표준을 진행하고 있다.

HEVC 국제표준화는 기존의 H.264/AVC의 기술을 기반으로 하여 고해상도, 고화질 영상에 보다 효율적인 새로운 알고리즘 등을 중심으로 진행되고 있다. 특히 화소수가 가로세로 16x16의 고정된 크기인 MB(Macroblock) 단위를 기본 부호화 단위로 사용하는 H.264/AVC에 반해, HEVC에서는 다양한 크기의 정사각형 모양으로 정의된 CU(Coding Unit)를 기본 부호화 단위로 사용한다^[3]. 이러한 CU의 크기는 가장 큰 단위인 LCU(Largest Coding Unit)에서부터 가장 작은 단위의 SCU(Smallest Coding Unit)까지 존재한다. LCU 및 SCU에서 사용하는 크기(화소수)는 SPS(Sequence Parameter Set)에 포함되어 전송된다^[2]. JCT-VC에서 제안되는 알고리즘의 성능평가를 위해서 사용하는

공통실험조건^[4]에서 LCU는 화소수가 가로세로 64x64 크기를 사용하고, SCU는 8x8 크기를 사용한다. CU는 가로 및 세로가 각각 절반 크기인 CU 4개로 분할할 수 있는데, 분할된 CU는 또한 재귀적으로 가로 및 세로가 절반 크기인 CU 4개로 분할될 수 있다. 위와 같은 과정을 인코더에서 미리 정의한 횟수까지 수행할 수 있는데, 다양한 크기의 CU는 그 크기에 따라 깊이(depth) 정보^[2]로 표현된다. LCU의 경우에는 가장 큰 단위이므로 깊이가 '0'으로 가장 낮고 SCU의 경우에는 가장 작은 단위이므로 가장 큰 깊이를 사용한다. 예를 들어 LCU 크기가 64x64이고 SCU가 8x8인 경우, 64x64 CU(LCU)는 깊이 '0'으로, 32x32 CU는 깊이 '1'로, 16x16 CU는 깊이 '2'로, 8x8 CU(SCU)는 깊이 '3'으로 표현된다. 또한 각각의 CU는 가로와 세로의 크기가 각각 절반인 CU 4개로 분할할지에 대한 정보를 분할 정보(Split information)^[2]를 통해 표현하고 해당 정보를 인코더에서 디코더로 전송하는데, 이 정보는 SCU를 제외한 모든 CU에 동일하게 포함되어 있다.

HEVC에서는 CU로 부호화 수행 시 영상을 LCU로 나누어서 영상의 특성에 맞게 분할구조를 결정한다. 분할구조는 LCU 내에서 영상을 효율적으로 부호화하기 위해 분할되는 CU의 구조를 의미하며, LCU 내의 CU 분할 정보를 통해 결정된다. 이 알고리즘은 기존의 저해상도 영상뿐만 아니라 초고해상도, 초고화질 영상을 부호화하는데 효율적으로 사용된다. 영상 내에서 복잡하거나 움직임이 많은 영역에서는 CU가 치밀하게 작게 분할되어 부호화되며, 반대로 움직임이 완만하거나 단순한 영역에서는 큰 CU단위로 성글게 부호화된다. 즉, 영상의 특성에 따라 LCU의 분할구조가 적응적으로 부호화됨으로써 코딩효율 측면에서 전송 비트량을 감소시키는 효과가 있다. 하지만 LCU 단위로 분할구조를 결정할 때, 인코더의 복잡도가 매우 높은 문제점이 있다. 이는 LCU 단위로 모든 깊이의 CU를 부호화하여 후보에 저장한 뒤, 코딩효율이 가장 높은 분할구조를 결정하기 때문이다. 이러한 문제점을 해결하기 위해 현재 부호화하려는 LCU를 예측하여 분할구조에 포함될 확률이 높은 CU만 부호화를 수행하는 방법을 통해 인코더의 복잡도를 감소시킬 수 있다. CU의 깊이 및 분할 정보의 특성을 이용해서 인코더의 복잡도를 감소시키고 분할 정보를 효율적으

a) 경희대학교 전자정보대학 컴퓨터공학과
Dept. of Computer Engineering, College of Electronics and Information,
Kyung Hee University

b) 한국전자통신연구원
ETRI

✉ 교신저자 : 박광훈(ghpark@khu.ac.kr)

※ 본 연구는 방송통신위원회의 ETRI 연구지원 사업[KCA-2011-(11921-02001), 무안정 다시점 3D지원 UHD TV 방송 기술개발]과 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 지원사업(NIPA-2011-(C1090-1111-0001))의 연구결과로 수행되었음.

· 접수일(2011년10월18일), 수정일(1차:2012년2월3일, 2차:3월5일), 게재확정일(2012년3월5일)

로 부호화하는 방법을 본 논문에서 제안하였다.

본 논문의 구성은 다음과 같다. II장에서는 기존의 CU 부호화 방법에 대해서 설명한다. III장에서는 CU 깊이 정보의 특성에 대하여 분석하고, IV장에서는 III에서 분석한 내용을 바탕으로 현재 부호화하려는 LCU를 예측하여 영상을 효율적으로 부호화 및 복호화하는 방법에 대하여 설명한다. V장에서 제안하는 방법에 대한 성능을 평가한다. 마지막으로 VI장에서는 결론을 기술한다.

II. HEVC의 CU 부호화 방법

HEVC에서는 영상을 효율적으로 부호화하기 위해서 CU로 부호화를 수행한다. <그림 1>은 영상을 부호화할 때 CU를 분할하는 방법을 나타낸 그림이다.

<그림 1>과 같이 HEVC 인코더에서는 영상을 LCU 단위로 순차적으로 분할한 후, LCU 단위로 분할구조를 결정하여 부호화한다. 분할구조는 LCU 내에서 영상을 효율적으

로 부호화하기 위한 CU의 분포를 의미하는데, 이 분포는 각각의 CU를 크기가 가로 세로 절반씩 감소한 4개의 CU로 분할할지를 결정함으로써 결정할 수 있다. 분할된 CU는 같은 방식으로 가로 세로 절반씩 감소한 4개의 CU로 재귀적으로 분할할 수 있는데, CU의 분할은 미리 정의된 깊이까지 분할을 수행할 수 있다^[2]. 기본이 되는 LCU의 깊이는 0이고, SCU의 깊이는 미리 정의된 최대 깊이이다. LCU로부터 가로 세로 절반으로 분할을 수행할 때마다 CU의 깊이가 1씩 증가한다. 각각의 깊이별로 분할을 하지 않는 CU의 경우에는 $2N \times 2N$ 의 크기로, 분할을 수행할 경우에는 $N \times N$ 크기의 CU 4개로 분할된다. N의 크기는 깊이가 1씩 증가할 때마다 절반으로 감소한다. <그림 1>은 최소 깊이가 0인 LCU의 크기가 64×64 화소이고, 최대 깊이가 3인 SCU의 크기가 8×8 화소인 예이다. 64×64 화소 CU(LCU)는 깊이 '0'으로 32×32 화소 CU는 깊이 '1'로 16×16 화소 CU는 깊이 '2'로 8×8 CU(SCU)는 깊이 '3'으로 표현된다. 또한 특정 CU를 분할할지에 대한 정보는 1비트 정보인 분할 정보를 통해 표현한다. 이 분할 정보는 SCU를 제외한 모든 CU

그림 1. LCU 단위로 CU를 분할하는 방법

Fig. 1. CU splitting for each LCU in HEVC

에 포함되어 있으며, 해당 깊이의 CU를 분할하지 않을 경우에는 분할 정보에 '0'을 저장하고, 분할할 경우에는 '1'을 저장한다. <그림 2>는 HEVC에서 LCU의 크기가 64x64 화소이고, 최대 깊이가 3인 경우 LCU 내에서 CU의 분할에 대한 예를 나타낸 그림이다.

그림 2. LCU 단위로 분할하는 분할구조의 예
Fig. 2. Example of the split structure in an LCU

<그림 2>와 같이 깊이가 0인 64x64 화소 LCU(CU₀)는

분할을 수행하여 깊이가 1인 32x32 화소 CU₁, CU₂, CU₁₁, CU₁₂로 분할하며 CU₀의 분할 정보에는 분할이 된다는 것을 의미하는 '1'이 저장된다. 따라서 해당 LCU는 깊이가 0인 CU 크기로 부호화를 수행하지 않고, 깊이가 1인 32x32 화소 CU 4개로 분할되어 동일한 과정을 거친다. CU₁, CU₁₁, CU₁₂는 더 이상 분할되지 않고 해당 크기의 CU로 부호화를 수행하므로 각각의 CU의 분할 정보에는 분할되지 않는다는 것을 의미하는 '0'이 저장되고, 이 단위로 부호화를 수행한다. 다음 CU₂는 깊이가 2인 16x16 화소 CU로 분할되어 부호화를 수행하므로 CU₂의 분할 정보에는 '1'이 저장되고, 깊이가 2인 CU₃, CU₄, CU₅, CU₆으로 분할한다. CU₃, CU₄, CU₅은 더 작은 CU로 분할되지 않고 해당 크기의 CU로 부호화하므로 각각의 CU의 분할 정보에는 '0'이 저장되고, 이 단위로 부호화를 수행한다. CU₆는 깊이가 3인 8x8 화소 CU로 분할되어 부호화를 수행하므로 CU₆의 분할 정보에는 '1'이 저장되고, 깊이가 3인 CU₇, CU₈, CU₉, CU₁₀으로 분할한다. CU₇, CU₈, CU₉, CU₁₀는 SCU이므로 분할 정보를 포함하지 않고, 이 단위로 부호화를 수행한다. <그림 3>은 <그림 2>의 LCU에 대한 CU 분할 정보의 구조를 나타낸 그림이다.

<그림 3>에서 확인할 수 있듯이, 분할 정보가 '0'일 경우에는 CU를 분할하지 않고 해당 깊이의 CU로 부호화를 수행하고, 분할 정보가 '1'일 경우에는 깊이가 1 증가한 CU

그림 3. 분할 정보의 구조
Fig. 3. Example structure of the split information in an LCU

4개로 동일한 과정을 거친다. 최대 깊이에 도달하면 더 이상 분할할 수 없으므로 최대 깊이의 CU는 분할 정보를 가지지 않고 해당 깊이의 CU로 부호화를 수행한다. 위에서 설명한 분할 정보를 통해 LCU 단위로 분할구조를 구성할 수 있다.

인코더에서 LCU 단위로 분할구조를 결정할 때, CU를 분할하여 부호화할지에 대한 판단은 LCU 내의 영상의 특성에 따라 결정된다. 인코더에서는 영상을 부호화할 때, LCU 단위로 모든 깊이의 CU로 부호화를 수행한 뒤, 부호화 정보를 CU 후보에 저장하고, 코딩효율이 가장 높은 분할구조를 결정한다. <그림 4>는 인코더에서 영상을 부호화하고 코딩효율이 가장 높은 분할구조를 결정하는 방법을 나타낸 순서도이다.

<그림 4>와 같이 인코더에서는, 최소 깊이인 LCU부터 부호화를 수행하여 분할구조 후보에 저장하고, 깊이가 1 증가한 4개의 CU로 분할하여 부호화를 수행하고 분할구조 후보에 저장한다. 이렇게 분할된 CU들은 다시 깊이가 1 증가한 4개의 CU로 분할하여 부호화를 수행하고 분할구조 후보에 저장한다. 이 과정은 최대 깊이인 SCU를 부호화할 때까지 수행된다. LCU 단위로 모든 깊이의 CU를 부호화한 후에는 부호화된 결과를 토대로 최적의 분할구조를 결정한다. 이 과정은 인코더에서 분할 정보를 포함한 CU의

정보를 율-왜곡 최적화 방법(Rate-distortion Optimization)^[5]을 통하여 선택하는데, 코딩효율과 화질 측면에서 가

그림 4. LCU 단위로 분할구조를 결정하는 방법

Fig. 4. Method to determine the split structure for each LCU

그림 5. LCU 단위로 분할구조를 결정하는 예

Fig. 5. Example of determining the split structure for each LCU in an image frame

장 효율적인 분할구조를 결정한다. <그림 5>는 HEVC의 증명모델인 HEVC HM 3.0 참조 소프트웨어(reference software)^[6]에서 앞에서 설명한 방법으로 부호화를 수행한 영상의 예를 보여준 그림이다. 실험 영상은 HEVC 테스트 시퀀스^[4](Test Sequence)인 ‘Blowingbubbles’이다.

<그림 5>에서 검정색 사각형은 LCU이고, LCU 내의 회색 구조는 LCU의 분할구조이다. 그림에서 보다시피 인코더는 영상을 미리 정의된 크기의 LCU 단위로 순차적으로 분할하고, 각각의 LCU 내에서 코딩효율이 가장 높은 분할구조를 결정하여 해당 CU로 부호화한다. 이 때, 분할구조는 LCU 단위로 독립적으로 부호화되는데, 복잡하거나 움직임이 많은 영역에서는 대체로 깊이가 큰 CU로 부호화를 수행하고, 완만하거나 움직임이 적은 영역에서는 대체로 깊이가 작은 CU로 부호화를 수행한다.

III. CU 깊이 정보의 특성

<그림 6>에 ‘BasketballPass’ 영상을 HEVC HM 3.0 참조 소프트웨어(reference software)^[6]를 통해 부호화를 수행한 일부영상 결과를 도시하였다.

<그림 6>에서 보다시피 <그림 6>의 05번 프레임과 <그림 6>의 06번 프레임의 동일한 위치에 있는 LCU(Co-lo-

cated LCU)의 분할구조는 서로 유사한 경향을 보이고 있는 것을 알 수 있다(<그림 6>의 ①, <그림 6>의 ②). <그림 6>의 ①과 같은 경우에는 상대적으로 큰 깊이의 CU로 부호화하고, <그림 6>의 ②와 같은 경우에는 상대적으로 작은 깊이의 CU로 부호화를 수행한다. 이와 같이 현재 프레임을 부호화할 때, 이전에 부호화를 수행한 참조 프레임 혹은 시간적으로 이전 프레임의 분할구조를 적극적으로 이용하여 인코딩시 계산 복잡도를 크게 줄일 수 있다. 즉, 현재 프레임 내의 LCU를 부호화할 경우, Co-located LCU의 분할구조를 통해 현재 LCU를 예측할 수 있다.

LCU의 분할구조에 대한 시간적 상관도를 알아보기 위하여 HEVC HM 3.0의 공통실험조건^[4]을 사용하여 몇 가지 실험을 진행하였다. HEVC HM 3.0의 공통실험조건^[4]에는 세 가지 constraint가 있는데, 각 constraint의 설명은 다음과 같다.

- 1) Intra-only: 고화질 콘텐츠 제작 환경을 고려한 constraint.
- 2) Low-delay: 저지연 환경을 고려한 constraint.
- 3) Random-access: 임의접근 환경을 고려한 constraint.

intra-only constraint는 모든 CU가 화면내 예측을 수행하는 인트라 모드로만 부호화하는데, 모든 프레임은 IDR

그림 6. 시간적으로 인접한 프레임간 분할구조의 예

Fig. 6. Example of split structures of the LCUs between temporally subsequent frames

(Instantaneous Decoding Refresh)^[2] 픽처이며, 프레임간의 양자화 변수(QP; Quantization Parameter)가 모두 동일한 값을 가진다. low-delay constraint는 첫 번째 프레임만 인트라 프레임이며 다음 프레임부터는 GPB(Generalized P-picture and B-picture)^[2]를 이용하여 화면간 예측을 수행하는 인터 프레임으로 부호화한다. 화면간 예측을 통해 부호화하는 프레임은 시간적 레이어(layer)에 따라 처음 인트라 프레임의 QP에 특정 오프셋(offset) 값을 추가한 QP로 부호화를 수행한다. 따라서 프레임 별로 QP의 값이 다르다. random-access constraint는 1초 단위로 한 프레임씩 인트라 프

레이미 존재하고, 계층적(Hierarchical) 구조로 과거 및 미래 프레임을 통해 예측하는 인터 프레임으로 부호화한다. 프레임들의 QP 값은 low-delay constraint와 유사하게 시간적 레이어(layer)에 따라 달라진다. 또한 각각의 constraint에 CABAC(Context-based Adaptive Binary Arithmetic Coding)^{[2][7]}과 같이 코딩효율이 우수하지만 복잡도가 높은 알고리즘이 포함되어 있는 HE(High-efficiency) 부호화 조건과 CAVLC(Context-Adaptive Variable Length Coding)^{[2][8]}와 같이 코딩효율은 상대적으로 떨어지지만 복잡도가 낮은 알고리즘이 포함되어 있는 LC(Low-complexity)

표 1. HEVC의 테스트 시퀀스
Table. 1. The HEVC Test Sequences

Class	Seq No.	Seq Name	Class	Seq No.	Seq Name
Class A (2560x1600)	S01	Traffic	Class C (832x480)	S10	PartyScene
	S02	PeopleOnStreet		S11	RaceHorses
Class B (1920x1080)	S03	Kimono	Class D (416x240)	S12	BasketballPass
	S04	ParkScene		S13	BQSquare
	S05	Cactus		S14	BlowingBubbles
	S06	BasketballDrive		S15	RaceHorses
	S07	BQTerrace		S16	Vidyo1
Class C (832x480)	S08	BasketballDrill	Class E (1280x720)	S17	Vidyo3
	S09	BQMall		S18	Vidyo4

깊이의 차의 절대값 평균

그림 7. 현재 LCU의 깊이와 Co-located LCU의 깊이의 차의 절대값 평균

Fig. 7. The absolute average value of subtraction of the CU depth between the current LCU and the co-located LCU

부호화 조건이 있다^[4].

실험에는 <표 1>과 같이 HEVC HM 3.0의 공통실험조건^[4]에서 사용하는 실험 영상들을 이용하였다.

<그림 7>은 현재 프레임의 LCU의 CU 깊이와 Co-located LCU의 CU 깊이의 차의 절대값 평균을 나타낸 그래프이다. intra-only constraint의 경우에는 시간적으로 이전에 부호화된 프레임들, low-delay constraint와 random-access constraint는 참조 프레임을 참조하였다.

<그림 7>에서 보듯이 CU 깊이의 차의 절대값 평균이 constraint별로, 테스트 시퀀스 별로 다르다. intra-only constraint의 경우에는 모든 프레임을 동일한 QP로 부호화하고 시간적으로 인접한 프레임을 참조하기 때문에 CU의 평균 깊이도 유사하다. low-delay constraint의 경우에는 인터 프레임을 부호화할 때, 처음의 인트라 프레임에서 특정 offset 값을 추가한 QP 값으로 부호화를 수행하기 때문에 프레임 간에 다른 QP로 부호화를 수행한다. 따라서 시간적으로 인접한 프레임을 참조하지만, 프레임간 CU의 평균 깊이의 차이가 intra-only constraint보다 큰 것을 알 수 있다. random-access constraint의 경우에는 low-delay constraint와 마찬가지로 1초 단위로 인터 프레임을 부호화할 때, 인트라 프레임에서 특정 offset 값을 추가한 QP 값으로 부호화를 수행한다. 또한 1초 단위로 계층적 구조를 가지기 때문에 low-delay constraint보다는 시간적으로 인접하지 않은 프

레이를 통해 예측하는 경우가 많으므로, CU의 평균 깊이의 차이가 low-delay constraint보다 큰 것을 알 수 있다. 결국, <그림 7>에서 보듯이, 정도의 차이는 있지만 모든 constraint에서 깊이의 차의 절대값 평균이 0에 가까운 것을 볼 수 있다. 깊이의 차의 절대값이 0에 가까울수록 현재 LCU와 Co-located LCU의 CU의 깊이가 유사하다는 것을 의미한다. 이를 통해 모든 constraint에서 현재 LCU와 Co-located LCU의 분할구조가 유사하다는 것을 알 수 있다. 따라서 현재 LCU를 부호화할 때, Co-located LCU를 통해 현재 LCU의 분할구조를 예측하여 부호화할 수 있다.

IV. 제안 방법

HEVC에서는 영상의 특성에 따라 LCU 단위로 분할구조를 결정하는데, 본 논문에서는 예측을 위해 LCU를 단순한 LCU와 복잡한 LCU, 그리고 일반 LCU의 3가지로 구분하는 방법을 제안한다. 이러한 LCU들은 CU의 깊이 정보를 통해서 정의하는데, LCU내의 모든 CU의 깊이가 0 또는 1일 경우에는 단순한 LCU로 정의하고, LCU 내에 최대 깊이의 CU가 1개 이상 존재하면 복잡한 LCU로 정의한다. 위의 조건에 모두 해당되지 않으면 일반 LCU로 정의한다.

<그림 8>은 HEVC HM 3.0 참조 소프트웨어(reference

그림 8. 일반 LCU, 복잡한 LCU, 단순한 LCU의 비율

Fig. 8. The ratios of Normal LCU, Complex LCU, Simple LCU in each constraints

software)^[6]로 영상을 부호화하여 constraint 별로 위에서 정의한 LCU의 비율을 나타낸 그래프이다.

앞서 정의한 LCU의 비율이 constraint별로 다른 것을 <그림 8>에서 발견할 수 있다. 이는 각 constraint의 예측모드, 예측구조가 다르기 때문이다. intra-only constraint의 경우 화면간 예측을 수행하지 않으므로, 큰 깊이의 CU로 정밀하게 화면내 예측을 수행해야하기 때문에 복잡한 LCU의 비율이 크고, low-delay constraint의 경우에는 시간적으로 인접한 과거 프레임들 통해 예측하므로, 예측 성능이 높기 때문에 비교적 작은 깊이의 CU가 많이 존재하여 단순한 LCU의 비율이 크다. random-access constraint의 경우에는 1초마다 화면내 예측을 수행하는 프레임이 존재하고, 화면간 예측을 수행할 때 시간적으로 인접하지 않은 프레임들 통해 예측할 경우가 많다. 따라서 큰 깊이의 CU가 intra-only constraint보다는 적게 존재하고, low-delay constraint보다는 많이 존재한다. 또한 모든 constraint에서 일반 LCU는 10% 내외로 존재하는데, 이는 현재 LCU를 부호화할 때, Co-located LCU가 복잡한지, 단순한지를 참조하여 예측할 수 있는 확률이 90%를 차지한다는 것을 의미한다.

<그림 9>는 현재 LCU를 부호화할 때, Co-located LCU가 복잡한 LCU 혹은 단순한 LCU인 경우, 현재 LCU와의 관계를 나타낸 그래프이다. intra-only constraint의 경우에는 시간적으로 이전에 부호화된 프레임들, low-delay constraint와 random-access constraint는 참조 프레임들 참조하

였다.

<그림 9>의 왼쪽과 같이 Co-located LCU가 복잡한 LCU일 경우, 현재 LCU에 최소 깊이 CU가 존재하지 않을 확률이 모든 constraint에서 약 99%이다. 또한 <그림 9>의 오른쪽과 같이 Co-located LCU가 단순한 LCU일 경우, 현재 LCU의 최대 깊이 CU가 존재하지 않을 확률이 약 95%이다. 이는 Co-located LCU가 단순한 LCU일 경우에는 현재 LCU도 단순할 확률이 매우 높기 때문에 최대 깊이의 CU로 부호화할 확률이 매우 작고, 복잡한 LCU일 경우에는 현재 LCU도 복잡할 확률이 매우 높기 때문에 최소 깊이의 CU로 부호화할 확률이 매우 작다는 것을 의미한다.

HEVC HM 3.0 인코더에서는 다양한 영상을 효율적으로 부호화하기 위하여 모든 깊이를 부호화를 수행한 뒤 최적의 분할구조를 결정한다. 하지만 모든 깊이의 CU를 부호화하면 인코더에서는 높은 계산 복잡도(Computational complexity)를 필요로 한다. 따라서 Co-located LCU를 통해 현재 LCU가 복잡한지 혹은 단순한지를 예측하고, 분할구조에 포함될 확률이 높은 CU만 부호화함으로써 계산 복잡도를 낮추는 과정이 빠른 속도를 요하는 부호화 시 절실히 필요하다.

1. 인코더만을 변경하는 성능개선 방법

본 논문에서는, 인코더에서 현재 LCU의 분할구조를 결

그림 9. Co-located LCU와 현재 LCU의 상관관계

Fig. 9. The correlation between the co-located LCU and the current LCU

그림 10. Co-located LCU를 통해 현재 LCU를 부호화하는 방법

Fig. 10. Proposed method to perform encoding of the current LCU using the LCU information of the co-located LCU

정할 때, Co-located LCU 정보를 통해 현재 LCU의 분할구조를 예측하고, 분할구조에 포함될 확률이 높은 CU만 부호화하는 방법을 제안한다. Co-located LCU가 단순한 LCU이거나 복잡한 LCU일 경우, 현재 LCU를 부호화하는 방법을 <그림 10>에 도시하였다.

<그림 10>에 도시한 방법은 intra-only constraint, low-delay constraint, random-access constraint에 모두 적용할 수 있다. 제안하는 방법은 Co-located LCU가 단순한 LCU인지 혹은 복잡한 LCU인지 혹은 일반 LCU인지에 대한 복잡도 정보를 이용한다. 이러한 복잡도 정보는 앞서 정의한

방법을 통해 구분하고, 이전 프레임의 부호화가 완료되었을 때 메모리에 저장되어 다음 프레임의 부호화 시 사용된다. 이 정보는 메모리에 저장되어 있어야 하므로 메모리 사용량이 증가할 수 있지만, LCU 하나당 2 비트이므로 메모리 사용량은 크게 증가하지 않는다.

intra-only constraint의 경우에는 현재 프레임을 부호화할 때 기본적으로 참조 프레임을 가지지 않는다. 따라서 제안하는 방법에서는 현재 프레임의 LCU를 부호화할 때 참조 프레임을 참조하는 것이 아니라, 시간적으로 이전에 부호화된 프레임의 Co-located LCU의 복잡도 정보만을 이용한다. 또한 현재 프레임의 LCU를 부호화할 때, 시간적으로 이전에 부호화된 프레임이 존재하지 않거나, Co-located LCU가 일반 LCU일 경우에는 제안하는 방법이 적용되지 않고 기존 HEVC HM 3.0의 방법이 수행된다. low-delay constraint와 random-access constraint의 경우에는 현재 프레임의 LCU를 부호화할 때, 참조 프레임이 존재하지 않거나, 참조 프레임 내의 Co-located LCU가 일반 LCU일 경우, 현재 프레임이 인트라 프레임일 경우에는 제안하는 방법이 적용되지 않고 기존 HEVC HM 3.0의 방법이 수행된다.

인코더에서 분할구조를 결정할 때, Co-located LCU 정보를 통해 현재 LCU를 부호화하는 방법의 자세한 수행과정은 다음과 같다.

- (1 단계) low-delay 혹은 random-access일 경우에는 참조 프레임 내에서, intra-only일 경우에는 시간적으로 이전에 부호화된 프레임 내에서 현재 부호화할 LCU와 동일한 위치에 대응되는 Co-located LCU 정보를 참조한다.
- (2 단계) Co-located LCU가 복잡한 LCU이고, 현재 LCU의 부호화하려는 CU의 깊이가 최소 깊이인 경우에는 현재 깊이의 CU를 부호화하지 않고 5단계를 수행한다. 이 경우 Co-located LCU가 복잡한 LCU이므로 현재 LCU도 복잡한 확률이 높다고 판단하여 단순한 영역을 부호화하는 최소 깊이의 CU를 부호화하지 않는다. 해당 조건에 만족하지 않을 경우, 3단계를 수행한다.

(3 단계) Co-located LCU가 단순한 LCU이고, 현재 LCU의 부호화하려는 CU의 깊이가 최대 깊이인 경우에는 현재 깊이의 CU를 부호화하지 않고 5단계를 수행한다. 이 경우 Co-located LCU가 단순한 LCU이므로 현재 LCU도 단순할 확률이 높다고 판단하여 복잡한 영역을 부호화하는 최대 깊이의 CU를 부호화하지 않는다. 해당 조건에 만족하지 않을 경우, 4단계를 수행한다.

(4 단계) 현재 깊이의 CU를 부호화하여 CU 후보에 저장한 후, 5단계를 수행한다.

(5 단계) 만일 현재 부호화하려는 CU의 깊이가 최대 깊이가 아닌 경우에는 6 단계를 수행한다. 그렇지 않다면 7단계를 수행한다.

(6 단계) 현재 깊이의 CU를 깊이가 1이 큰 CU 4개로 분할하여 1단계를 수행한다.

(7 단계) 현재 LCU에 저장되어 있는 CU 후보들 중 왜곡 최적화 방법(Rate-distortion Optimization)^[5]을 통하여 화질 및 비트량 측면에서 가장 효율적인 분할구조를 선택하고 종료한다.

예를 들어, Co-located LCU가 복잡한 LCU일 경우에는 현재 LCU를 부호화할 때 최소 깊이의 CU는 부호화하지 않고, 깊이가 '1'부터 최대 깊이까지의 CU를 부호화하여 그 중에 가장 효율적인 분할구조를 결정한다. Co-located LCU가 단순한 LCU일 경우에는 현재 LCU를 부호화할 때 최대 깊이인 CU를 부호화하지 않고, 최소 깊이부터 '최대 깊이-1'의 CU를 부호화하여 그 중에 가장 효율적인 분할구조를 결정한다. 이 방법은 모든 깊이를 부호화하는 방법보다는 비트 효율이 좋지 않지만, 적은 비트량 손실로 인코더 복잡도를 현저히 낮출 수 있다.

앞서 제안한 방법 외에 Co-located LCU가 단순한 LCU일 경우만 고려하는 방법을 <그림 11>에 도시하였다.

<그림 11>에서는 <그림 10>과는 달리 인코더에서 현재 LCU의 분할구조를 결정할 때, Co-located LCU가 단순한 LCU이고, 현재 LCU의 부호화하려는 CU의 깊이가 최대 깊이의 경우에만 부호화를 수행하지 않는다. random-

그림 11. Co-located LCU가 단순한 LCU일 때만 예측하여 부호화하는 방법

Fig. 11. Proposed method to perform encoding of the current LCU using the LCU information of the co-located LCU when the co-located LCU is simple LCU

access constraint의 경우에는 1초 단위로 계층적인 구조로 예측을 수행하기 때문에, 시간적으로 인접하지 않은 프레임들을 통해 예측하여 높은 QP로 부호화를 수행한다. 따라서 Co-located LCU가 복잡한 LCU일 경우에도 최소 깊이의 CU가 선택될 확률이 높다. 위와 같은 경우에는 Co-located LCU가 단순한 LCU일 경우만 고려해서 예측을 수행하는 것이 코딩효율 측면에서 효과적이다.

2. 인코더와 비트스트림 구조 및 디코더를 변경하는 성능 개선 방법

HEVC HM 3.0의 방법은 인코더에서 분할구조를 결정할

때 모든 깊이의 CU를 부호화하여 CU 후보에 저장하므로 모든 깊이의 CU가 실제 부호화되는 CU일 가능성이 있다. 따라서 모든 깊이의 CU를 부호화 및 복호화하기 위해서는 기본적으로 SCU를 제외한 모든 CU에 분할 정보를 포함시켜야 한다.

앞에서 제시한 방법은 인코더에서 분할구조를 결정할 때, 현재 LCU의 분할구조에 포함될 확률이 높은 CU만 부호화하여 인코더의 계산 복잡도를 낮출 수 있다. 하지만, 이와 같은 방법은 인코더의 LCU 분할구조를 결정하는 과정만 변경하므로, 이후 CU 분할 정보를 부호화 및 복호화하는 과정에서 부호화를 수행하지 않은 깊이의 불필요한 CU 분할 정보도 부호화 및 복호화하는 단점이 있다. <그림

그림 12. CU 분할 정보를 부호화 및 복호화하는 방법

Fig. 12. Proposed method to encode and decode the CU split information

12>는 <그림 10>에서 제안한 방법을 통해 분할구조를 결정 후, 적응적으로 CU 분할 정보를 부호화 및 복호화하는 방법을 나타낸 그림이다.

<그림 12>에서 보듯이, CU 분할 정보를 부호화 및 복호화할 때, 앞에서 제안한 방법을 동일하게 적용하여 CU 분할 정보를 참조하지 않고도 해당 깊이의 CU로 분할할지 여부를 추론할 수 있다. Co-located LCU의 복잡도 여부를 참조하는 방법은 앞서 제안한 방법과 동일하다. 이와 같은 과정은 intra-only constraint를 제외한 low-delay constraint 및 random-access constraint에 적용할 수 있는데, intra-only constraint는 복호화 시 현재 프레임 외에 어떤 프레임으로

부터도 예측하는 것이 불가능하기 때문에 위에서 제시한 방법을 적용할 수 없다. low-delay constraint와 random-access constraint의 경우에는 현재 프레임의 LCU를 부호화할 때, 참조 프레임이 존재하지 않거나, 참조 프레임 내의 Co-located LCU가 일반 LCU일 경우, 현재 프레임이 인트라 프레임일 경우에는 제안하는 방법이 적용되지 않고 기존 HEVC HM 3.0의 방법이 수행된다.

제안하는 방법에 대한 일례를 Co-located LCU가 복잡한 LCU일 경우와 단순한 LCU일 경우로 나누어 설명하였다. <그림 13>은 <그림 12>에서 Co-located LCU가 복잡한 LCU일 경우, 현재 LCU의 분할구조 및 비트스트림의 구조

그림 13. Co-located LCU가 복잡한 LCU일 경우의 현재 LCU 분할 정보 구조

Fig. 13. The CU split information structure if the co-located LCU is complex LCU

를 나타낸 그림이다.

<그림 13>에서 보듯이, Co-located LCU가 복잡한 LCU이고, 현재 LCU의 분할구조가 왼쪽 그림과 같을 경우, 부호화 및 복호화하는 현재 LCU의 CU 분할 정보 구조는 오른쪽 그림과 같다. 최소 깊이의 CU인 경우에는 인코더에서 분할 구조를 결정할 때의 제안하는 방법과 동일하게 Co-located LCU를 통해 분할한다고 추론할 수 있으므로, 해당 깊이의 분할 정보를 부호화 및 복호화 하지 않고 무조건 분할한다. SCU를 제외한 모든 깊이의 CU는 기존 방법

대로 분할 정보를 부호화 및 복호화한다. 다음으로 <그림 14>는 Co-located LCU가 단순한 LCU일 경우, 현재 LCU의 분할구조 및 비트스트림의 구조를 나타낸 그림이다.

<그림 14>에서 보듯이, Co-located LCU가 단순한 LCU이고, 현재 LCU의 분할구조가 왼쪽 그림과 같을 경우, 부호화 및 복호화하는 현재 LCU의 CU 분할 정보 구조는 오른쪽 그림과 같다. 깊이가 '최대 깊이-1'인 CU인 경우에는 인코더에서 분할 구조를 결정할 때의 제안하는 방법과 동일하게 Co-located LCU를 통해 분할하지 않는다고 추론

그림 14. Co-located LCU가 단순한 LCU일 경우의 현재 LCU 분할 정보 구조

Fig. 14. The CU split information structure if the co-located LCU is simple LCU

할 수 있으므로, 해당 깊이의 분할 정보를 부호화 및 복호화하지 않고 무조건 분할하지 않는다.

V. 성능 평가

본 논문에서 제안하는 방법의 우수성을 검증하기 위해 HEVC HM 3.0 참조 소프트웨어(reference software)^[6]에 제안하는 방법을 구현하였다. 실험 영상은 <표 1>의 HEVC의 테스트 시퀀스^[4]이며 기타 실험 조건은 HEVC 공통실험조건^[4]을 준수하였다. <표 2>에 제안하는 방법 중 인코더만 변경한 경우의 시뮬레이션 결과를 도시하였다. 실험은 Intel(R) Xeon(R) CPU E5630 @ 2.53GHz, 16.0GB

메모리, Windows Server SP2 64비트 O/S환경에서 진행하였다. 제안하는 방법의 계산 복잡도를 측정하기 위해 <표 1>에서 도시한 테스트 시퀀스를 인코더로 부호화하여 비트스트림을 생성하는 과정의 평균 시간(Encoding Time)을 측정하였고, 비트스트림을 디코더로 복호화하면서 소모되는 평균 시간(Decoding Time)을 측정하였다. 또한 부호화 효율을 측정하기 위해 평균적인 bit-rate 감소량을 나타내는 BD-Bitrate 방법^[9]을 사용하였다.

<표 2>는 인코더만 변경하는 방법 중에서 <그림 10>의 방법에 대한 시뮬레이션 결과이다.

<표 2>와 같이 인코더의 복잡도는 intra-only constraint에서 HE의 경우에는 15% 감소하였고, LC의 경우에는 인

표 2. 인코더만 변경하는 방법의 실험 결과

Table. 2. Simulation results of modifying only the encoder

	Intra Only HE			Intra Only LC		
	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate
Class A	0.0	0.0	0.0	0.0	0.0	0.0
Class B	0.0	0.0	0.0	0.0	0.0	0.0
Class C	0.0	0.0	0.0	0.0	0.0	0.0
Class D	0.0	0.0	0.0	0.0	0.0	0.0
Class E	0.1	0.0	0.1	0.0	0.0	0.0
All	0.0	0.0	0.0	0.0	0.0	0.0
Enc Time[%]	85%			84%		
Dec Time[%]	98%			101%		

	Low-delay HE			Low-delay LC		
	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate
Class A						
Class B	0.3	-0.4	-0.8	0.4	0.3	0.3
Class C	0.3	-0.1	0.0	0.4	0.4	0.2
Class D	0.2	-0.4	-0.4	0.3	0.2	0.2
Class E	0.6	-0.2	-1.1	0.5	0.5	0.7
All	0.3	-0.3	-0.6	0.4	0.3	0.3
Enc Time[%]	74%			70%		
Dec Time[%]	98%			98%		

	Random Access HE			Random Access LC		
	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate
Class A	0.6	0.4	0.6	0.7	0.7	0.7
Class B	0.7	0.5	0.6	0.8	0.9	0.9
Class C	0.5	0.5	0.5	0.5	0.6	0.5
Class D	0.3	0.2	0.2	0.3	0.3	0.3
Class E						
All	0.5	0.4	0.4	0.6	0.6	0.6
Enc Time[%]	80%			79%		
Dec Time[%]	100%			99%		

코더의 복잡도가 16% 감소하였다. low-delay constraint에서 HE의 경우에는 26% 감소하였고, LC의 경우에는 30% 감소하였다. random-access constraint에서 HE의 경우에는 20% 감소하였고, LC의 경우에는 21% 감소하였다. BD-Bitrate(Y)는 intra-only constraint에서 HE와 LC 모두 0%로 변화가 없었다. low-delay constraint에서 HE의 경우에는 0.3% 증가하였고, LC의 경우에는 0.4% 증가하였다. random-access constraint에서 HE의 경우에는 0.5% 증가하였고, LC의 경우에는 0.6% 증가하였다. 디코더의 복잡도는 기존의 방법과 큰 차이가 없었다.

인코더 복잡도의 감소를 살펴보면, intra-only constraint의 경우에는 다른 constraint에 비해 상대적으로 인코더의 복잡도가 적게 감소한 것을 볼 수 있는데, 이는 <그림 8>에서 보듯이 low-delay constraint와 random-access constraint에 비해 단순한 LCU 비율이 상대적으로 작기 때문이다. 제한하는 방법의 인코더의 복잡도는, 참조 혹은 시간적으로 이전 프레임 내의 LCU 비율 중 단순한 LCU가 많을수록 많이 감소한다. 이는 HEVC HM 3.0의 인코더의 복잡도를 차지하는 비중은 CU의 깊이가 높아질수록 증가하는데, Co-located LCU가 단순한 LCU일 경우, 현재 LCU를 부호화할 때 LCU 내의 모든 최대 깊이 CU를 부호화하지 않기 때문이다. 반면에 low-delay constraint의 경우에는 단순한 LCU의 비율이 상대적으로 크므로 인코더의 복잡도가 많이 감소한 것을 볼 수 있다. random-access constraint의 경우에는 단순한 LCU의 비율이 intra-only constraint보다는 크고, low-delay constraint보다는 작기 때문에 인코더의 복잡도 또한 in-

tra-only constraint보다는 크고, low-delay constraint보다는 작게 감소하였다. 결국 인코더의 복잡도 감소는 <그림 8>의 단순한 LCU의 비율의 영향을 받는 것을 알 수 있다.

또한 BD-Bitrate(Y)의 차이를 살펴보면, intra-only constraint에서는 BD-Bitrate의 차이가 없는 것을 알 수 있는데, 이는 <그림 7>에서 보듯이 Co-located LCU의 깊이 정보와 현재 LCU의 깊이 정보가 매우 유사하기 때문이다. low-delay constraint에서는 BD-Bitrate(Y)가 0.3%에서 0.4%로 증가하였는데, Co-located LCU의 깊이 정보와 현재 LCU의 깊이 정보가 다른 경우가 있기 때문이다. 즉, 제한하는 방법으로 인해 부호화를 수행하지 않는 깊이의 CU가 코딩효율과 화질 측면에서 가장 효율적인 CU일 경우는 있는 것을 의미한다. 또한 random-access constraint에서는 BD-Bitrate(Y)가 0.5%에서 0.6%로 증가하였는데, 이는 random-access constraint의 경우 low-delay constraint보다 시간적으로 멀리 있는 참조 프레임을 통해 예측하는 경우가 많으므로, Co-located LCU의 깊이 정보와 현재 LCU의 깊이 정보의 차이가 low-delay constraint보다 크기 때문이다. 결국 BD-Bitrate는 Co-located LCU의 CU 깊이 정보에 대한 예측 효율이 좋을수록 적게 증가한 것을 볼 수 있다. <그림 7>에서 보듯이 참조 프레임간의 평균 깊이의 차가 가장 작은 intra-only constraint의 BD-Bitrate 증가가 가장 적은 것을 볼 수 있고, low-delay constraint, random-access constraint 순으로 BD-Bitrate 증가가 많아지는 것을 볼 수 있다.

<표 3>은 인코더만 변경하는 방법 중에서 <그림 11>의 방법에 대한 시뮬레이션 결과이다. 이 방법은 random-

표 3. Co-located LCU가 단순한 LCU일 때만 예측하여 부호화하는 방법의 실험 결과

Table. 3. Simulation results of modifying only the encoder when the co-located LCU is simple LCU

	Random Access HE			Random Access LC		
	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate
Class A	0.1	0.0	0.0	0.0	0.0	0.1
Class B	0.1	0.0	0.0	0.1	0.2	0.1
Class C	0.2	0.2	0.2	0.2	0.2	0.2
Class D	0.0	0.1	0.0	0.1	0.1	0.0
Class E						
All	0.1	0.1	0.1	0.1	0.1	0.1
Enc Time[%]	86%			85%		
Dec Time[%]	99%			98%		

access constraint에만 적용하였다.

<표 3>과 같이 인코더의 복잡도는 HE의 경우에는 14% 감소하였고, LC의 경우에는 15% 감소하였다. BD-Bitrate(Y)는 HE의 경우에는 0.1% 증가하였고, LC의 경우에는 0.1% 증가하였다. 디코더의 복잡도는 기존의 방법과 큰 차이가 없었다. 인코더 복잡도는 <표 2>의 random-access constraint보다 적게 감소한 것을 볼 수 있는데, 이는 Co-located LCU가 복잡한 LCU일 때는 적용하지 않았기 때문이다. 즉, <표 2>와 비교할 때 인코더 복잡도 차이 6%는, Co-located LCU가 복잡한 LCU일 때 현재 LCU의 최소 깊이의 CU를 부호화하는 차이라고 할 수 있다. 또한 BD-Bitrate(Y)의 차이를 살펴보면, <표 2>와는 달리 HE와 LC에서 0.1%로 BD-Bitrate의 손실이 줄어든 것을 알 수 있는데, 이는 Co-located LCU가 복잡한 LCU일 경우에 현재 LCU의 분할구조에 최소 깊이의 CU가 코딩효율과 화질 측면에서 가장 효율적인 CU일 경우가 많다는 것을 의미한다.

<표 4>에 제안하는 방법 중 <그림 10>, <그림 12>의 인코더와 비트스트림, 그리고 디코더를 변경한 경우의 시뮬레이션 결과를 도시하였다.

<표 4>와 같이 인코더의 복잡도는 low-delay constraint

에서 HE의 경우에는 24% 감소하였고, LC의 경우에는 27% 감소하였다. random-access constraint에서 HE의 경우에는 18% 감소하였고, LC의 경우에는 19% 감소하였다. BD-Bitrate(Y)는 low-delay constraint에서 HE와 LC 모두 0.1% 증가하였다. random-access constraint에서는 HE와 LC 모두 0.3% 증가하였다. 디코더의 복잡도는 기존의 방법과 유사하게 거의 변화가 없었다. 시뮬레이션 결과를 살펴보면, 인코더의 복잡도 감소는 <표 2>와 유사하며, 동시에 BD-Bitrate 증가는 작도록 개선됨을 확인할 수 있었다.

VI. 결 론

본 논문에서는 고효율 비디오 부호화(HEVC) 표준에서 Co-located LCU 정보를 통해, 현재 LCU를 예측하여 부호화 및 복호화를 적응적으로 수행하는 방법에 대한 연구를 수행하였다. 참조 프레임 혹은 시간적으로 이전 프레임의 깊이 정보는 현재 프레임의 깊이 정보가 유사하기 때문에, 인코더에서 현재 LCU의 분할구조를 결정할 때, Co-located LCU를 통해 현재 LCU를 예측하여 부호화할 수 있다. 이와

표 4. 인코더와 비트스트림 구조 및 디코더를 변경한 방법의 실험 결과
Table. 4. Simulation results of modifying both the encoder and decoder

	Low-delay HE			Low-delay LC		
	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate
Class A						
Class B	0.1	-0.6	-0.6	0.1	-0.1	0.1
Class C	0.1	-0.2	-0.2	0.1	0.1	0.1
Class D	0.1	-0.4	0.1	0.2	0.1	0.2
Class E	0.1	-0.5	-0.7	0.0	-0.1	-0.6
All	0.1	-0.4	-0.4	0.1	0.0	0.0
Enc Time[%]	76%			73%		
Dec Time[%]	98%			98%		

	Random Access HE			Random Access LC		
	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate
Class A	0.3	0.2	0.2	0.3	0.3	0.4
Class B	0.5	0.3	0.3	0.5	0.6	0.5
Class C	0.4	0.4	0.3	0.4	0.4	0.4
Class D	0.2	0.1	0.2	0.2	0.2	0.2
Class E						
All	0.3	0.3	0.3	0.3	0.4	0.4
Enc Time[%]	82%			81%		
Dec Time[%]	101%			101%		

같이 LCU 단위로 분할구조를 결정할 때, 분할구조에 포함될 확률이 높은 CU만 부호화를 수행함으로써 인코더의 복잡도를 15%에서 30%까지 감소시킬 수 있다. 하지만 부호화를 수행하지 않는 CU가 화질 및 비트량 측면에서 가장 효율적인 CU인 경우가 존재하기 때문에 최대 0.6%까지 BD-Bitrate(Y) 증가가 있었다. 이에 대한 해결책으로 위 방법으로 분할구조를 결정하고, 부호화하지 않는 CU의 분할 정보를 부호화 및 복호화하지 않음으로써 BD-Bitrate의 이득을 볼 수 있었다. 이 방법을 통해 18%에서 27%까지 인코더의 복잡도가 감소하였고, BD-Bitrate(Y) 증가는 최대 0.3%로 개선할 수 있었다. 결국, 참조 프레임 혹은 시간적으로 이전 프레임을 통해 현재 CU의 깊이를 예측하여 부호화하는 방법은 BD-Bitrate의 변화 없이 인코더의 복잡도를 크게 감소시킴으로써, 현재 국제표준화가 진행 중인 HEVC HM 3.0 방법보다 우수함을 확인할 수 있었다.

참 고 문 헌

- [1] ITU-T Recommendation H.264 and ISO/IEC 14496-10 (MPEG-4 Part 10 AVC), "Advanced Video Coding for Generic Audiovisual Services," Version 1: March 2003, Version 2: May 2004, Version 3: March 2005, Version 4: September 2005, Version 5 and Version 6: June 2006, Version 7: April 2007, Version 8: July 2007.
- [2] Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, "WD3: Working Draft 3 of High-Efficiency Video Coding", JCTVC-E603, Geneva, Switzerland, March 2011.
- [3] Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, "HEVC Test Model 3 (HM 3) Encoder Description", JCTVC-E602, Geneva, Switzerland, March 2011.
- [4] Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, "Common test conditions and software reference configurations", JCTVC-E700, Geneva, Switzerland, March 2011.
- [5] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G.J. Sullivan, "Rate-Constrained Coder Control and Comparison of Video Coding Standards," IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, No. 7, pp. 688-703, July 2003.
- [6] https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/
- [7] D. Marpe, H. Schwarz, and T. Wiegand, "Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard," IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, No. 7, pp. 620-636, July 2003.
- [8] K. Ugur, K. Andersson, A. Fuldseth, G. Bjøntegaard, L. P. Endresen, J. Lainema, A. Hallapuro, J. Ridge, D. Rusanovskyy, C. Zhang, A. Norkin, C. Priddle, T. Rusert, J. Samuelsson, R. Sjöberg, and Z. Wu, "High Performance, Low Complexity Video Coding and the Emerging HEVC Standard," IEEE Transactions on Circuits and Systems for Video Technology, Vol. 20, No. 12, pp. 1688-1697, December 2010.
- [9] G. Bjøntegaard, "Calculation of average PSNR differences between RD-curves," ITU-T SG16 Q.6, VCEG-M33, Texas, USA, April 2001.

저 자 소 개

이 한 수

- 2010년 8월 : 경희대학교 컴퓨터공학과 학사
- 2012년 2월 : 경희대학교 컴퓨터공학과 석사
- 주관심분야 : HEVC(High Efficiency Video Coding)

김 경 용

- 2007년 2월 : 경희대학교 컴퓨터공학과 학사
- 2009년 2월 : 경희대학교 컴퓨터공학과 석사
- 2009년 3월 ~ 현재 : 경희대학교 컴퓨터공학과 박사과정
- 주관심분야 : 비디오 신호처리, 영상 압축, 영상처리, 멀티미디어

저 자 소 개

김 태 룡

- 2010년 2월 : 경북대학교 컴퓨터공학과 학사
- 2012년 2월 : 경북대학교 컴퓨터공학과 석사
- 주관심분야 : HEVC(High Efficiency Video Coding)

박 광 훈

- 1985년 2월 : 연세대학교 전자공학과 학사
- 1987년 7월 : 연세대학교 전자공학과 석사
- 1991년 1월 : Case Western Reserve University, Dept. of EEAP 석사
- 1995년 1월 : Case Western Reserve University, Dept. of EEAP 박사
- 1995년 3월 ~ 1997년 2월 : 현대전자 멀티미디어연구소 책임연구원
- 1997년 3월 ~ 2001년 2월 : 연세대학교 문리대학 전산학과 부교수
- 2001년 3월 ~ 현재 : 경북대학교 컴퓨터공학과 교수
- 주관심분야 : 멀티미디어, 비디오 신호처리, 패턴인식, 영상처리, 계산지능

김 휘 용

- 1994년 8월 : 한국과학기술원(KAIST) 전기및전자공학과 (공학사)
- 1998년 2월 : 한국과학기술원(KAIST) 전기및전자공학과 (공학석사)
- 2004년 2월 : 한국과학기술원(KAIST) 전기및전자공학과 (공학박사)
- 2003년 8월 - 2005년 10월 : (주)애드팩테크놀로지 기술연구소 멀티미디어팀장
- 2005년 11월 - 현재 : 한국전자통신연구원(ETRI) 방통융합미디어연구부 선임연구원
- 2006년 9월 - 2010년 8월 : 과학기술연합대학원대학교(UST) 이동통신및디지털방송공학과 겸임교수
- 주관심분야 : 영상 압축, 컴퓨터 비전, 멀티미디어 시스템

임 성 창

- 2006년 2월 : 세종대학교 컴퓨터공학과 (공학사)
- 2008년 2월 : 세종대학교 컴퓨터공학과 (공학석사)
- 2008년 3월 - 현재 : 한국전자통신연구원(ETRI) 방통융합미디어연구부 연구원
- 주관심분야 : 영상 압축, 컴퓨터 비전, 멀티미디어 시스템

이 진 호

- 2007년 2월 : 고려대학교 전자및정보공학부 (공학사)
- 2009년 2월 : 과학기술연합대학원대학교(UST) 이동통신 및 디지털방송공학과 (공학석사)
- 2009년 3월 - 현재 : 한국전자통신연구원(ETRI) 방통융합미디어연구부 연구원
- 주관심분야 : 영상 압축, HEVC