HEVC 기술 소개 - Part II: 부호화 성능 향상 툴


김 재 일
한국과학기술원


안 상 수
한국과학기술원


김 대 은
한국과학기술원


김 문 철
한국과학기술원

Ⅰ. 서 론

HEVC는 기존 H.264/MPEG-4 AVC의 동등 주관적 화질 대비 약 2배의 압축 부호화 성능 목표를 달성하였다. 이러한 부호화 성능 개선은 HEVC의 표준 참조 소프트웨어인 HM(HEVC test Model)을 통해 확인 가능하다. 본고에서는 현재 시점에서 HEVC 표준 참조 소프트웨어의 최신 버전인 HM 15.0을 기준으로 부호화기의 설계 및 구현에 대한 툴 단위의 상세 기능 분석과 부호화 성능 분석을 실험 결과와 함께 설명한다.

Ⅱ. HM의 획득 및 실행

HM은 HEVC 표준화 과정에서 제안된 기술의 성능 및 복잡도를 검증하기 위한 용도로 사용되었다. HEVC 표준화를 진행하면서 HEVC 표준안으로 채택된 기술들은 참조 소프트웨어인 HM에 구현되었고, 이미 채택된 다른 제안 기술들과 통합되면서 부·복호화기의 성능 및 복잡도가 지속적으로 개선되었다. HM 소스 코드는 누구에게나 접근 가능하며 다양한 목적으로도 사용 가능하다[1]. HM 소스 코드의 build 폴더 내 visual studio 또는 linux로 컴파일 할 수 있는 프로젝트 파일이 존재 한다. 이를 통해 컴파일을 하면 bin 폴더 내 실행파일인 TAppEncoder와 TAppDecoder 파일이 생성된다(visual studio 인 경우). 이 파일로 -c [configuration.cfg]를 통해 쉽게 실행이 가능하다. HM의 경우 AVC와 비슷하게 configuration (cfg) 파일을 통해 실행이 가능한데, 기본적인 실험을 위한 cfg 파일은 HM의 cfg 폴더 내에 저장되어 있다. 이를 이용하여, 각 입력 영상 및 양자화 파라미터 값에 대해 부호화기 실행이 가능하다. HM의

부호화기와 복호화기 실행을 위한 좀 더 자세한 정보는 HM의 doc 폴더 내에 있는 software-manual.pdf 파일에 상세설명이 되어 있다.

본고에서는 HM의 부호화 성능 향상에 기여하는 여러 툴에 대해 설명한 후 각 툴이 어느 정도의 부호화 성능 개선에 기여하였는지를 확인하기 위하여 임의 접근(Random Access: RA) 부호화 조건에서 부호화 실험을 수행하고 실험 결과와 분석 내용을 제시한다. 또한, 모든 영상에 대한 테스트는 많은 리소스를 필요하기 때문에 Class A 영상인 *Kimono*, *ParkScene*, *Cactus*, *BasketballDrive* 영상 100 프레임에 대하여 실험하였다. 실험을 위하여 HM 15.0 버전을 사용하였으며, HM에서 권고하는 실험조건을 통해 실험을 수행하였고, JCTVC-L1100 문서에 포함된 엑셀파일을 이용하여 Y-BDBR로 결과를 정리하였다[2].

HM(HEVC test Model)은 한 CTU 내의 입력 영상을 효율적으로 부호화하기 위해 최적의 CU, PU 및 TU의 모드 조합 결정하는데, 이는 재귀 구조의 복잡한 수행 과정을 거친다.

Ⅲ. HM 기본 구성 기술

본 장에서는 HM의 기본 구성 기술인 확장 블록 부호화 구조와 화면내 예측, 화면간 예측, 변환 및 양자화, 엔트로피 부호화, 인루프 필터에 대한 기술을 상세히 소개하고, 각 부호화 툴의 성능을 제시한다. 또한, 본 장의 마지막에는 HEVC와 AVC의 부호화 성능을 비교하여 제시한다.

1. 확장 블록 부호화 구조

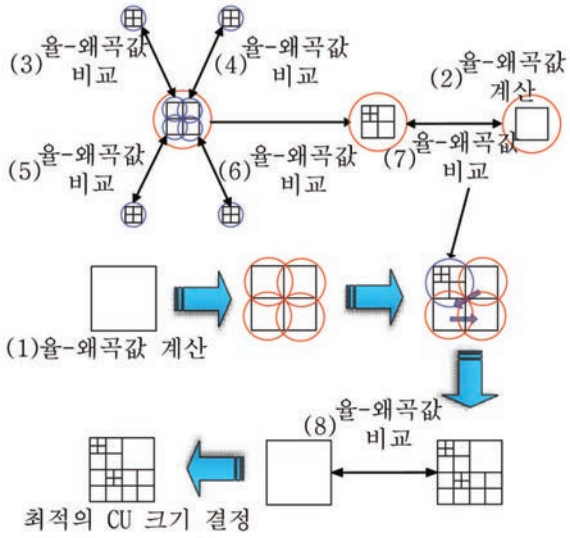
HM은 한 CTU 내의 입력 영상을 효율적으로 부호화하기 위해 최적의 CU, PU 및 TU의 모드 조합을 결정하는데, 이는 재귀 구조의 복잡한 수행 과정을 거친다. 〈그림 1〉은 64×64 CTU내 최적의 CU 분할 과정을 나타낸다. CU가 64×64 크기에서 8×8 크기까지의 분할 과정을 거치면서 율-왜곡 비용을 계산하고, 분할 전후와 비교하여 최적의 분할 모드를 결정하는 과정을 나타낸다. 구체적인 과정은 다음과 같다.

과정 (1) : 64×64 크기의 CU에 대해 예측, 변환/양자화, 역양자화/역변환 및 CABAC 수행을 통해 최소의 율-왜곡 값을 발생하는 최적의 PU와 TU 부호화 모드를 결정한다.

과정 (2) : 64×64 CU를 32×32 크기의 CU 4개로 분할하고 첫 번째 CU에 대하여 64×64 CU와 동일한 과정을 거쳐 율-왜곡값이 최소인 최적의 PU와 TU 부호화 모드를 결정한다.

과정 (3)-(6) : (2)의 32×32 CU를 16×16 크기의 CU 4개로 다시 분할하고, 각 16×16 CU에 대해 율-왜곡값이 최소인 최적의 PU와 TU 부호화 모드를 결정하고, 다시 16×16 CU에 대해 4개의 8×8 CU로 분할하고 각 8×8 CU에 대해 율-왜곡값이 최소인 최적의 PU와 TU 부호화 모드를 결정한다. 4개 8×8 CU의 율-왜곡값의 합과 16×16 CU의 율-왜곡값과 비교하여 16×16 블록의 분할 여부를 결정한다. 이를 나머지 3개의 16×16 CU들에 대해서도 동일한 과정을 수행한다.

과정 (7) : 과정 (2)에서 계산된 첫 번째 32×32 CU의 율-왜곡값과 과정 (3)-(6)에서 얻은 4개



〈그림 1〉 64×64 CTU내 최적의 CU 분할 과정

〈표 1〉 CTU 크기에 따른 Y-BDBR 증가

	32×32 CTU	16×16 CTU
Kimono	3.5%	23.4%
ParkScene	2.3%	10.3%
Cactus	2.3%	13.0%
BasketballDrive	4.1%	23.5%
Average	3.1%	17.6%

16×16 CU의 율-왜곡값의 합과 비교하여 첫 번째 32×32 블록 내에서 최적 조합의 CU 분할을 결정한다. 이를 나머지 3개의 32×32 CU들에 대해서도 동일한 과정을 수행한다.

과정 (8) : 마지막으로 64×64 CU 율-왜곡 값과 4개의 32×32 CU의 율-왜곡값의 합을 비교하여 64×64 블록 내에서 최적 조합의 CU 분할을 결정한다.

HEVC의 기본 부호화 단위 블록인 CTU의 크기는 HM 15.0 부호화기에서 cfg 파일의 파라미터인 MaxCUWidth와 MaxCUHeight 값으로 설정할 수 있다. 〈표 1〉은 HM 15.0에서 64×64 CTU 대비 32×32 CTU와 16×16 CTU의 부호화 성능을 나타내며 HEVC가 AVC의 16×16 크기의 매크로블록 비해 64×64와 32×32 크기 단위의 부호화 블록을 새롭게 도입함으로써 부호화 효율을 크게 개선 되었음을 알 수 있다.

〈표 1〉에서 보듯이 CTU 크기를 32×32로 설정하였을 경우 성능저하(비트율 증가)는 평균 3.1%로 크지 않지만, 16×16 크기로 설정 했을 때 큰 성능 저하가 발생하는 것을 알 수 있다. 이는, HEVC에서 64×64 CU로 인한 부호화 효율 성능 향상은 대체로 작지만, 32×32 CU로 인한 성능 향상이 상대적으로 크다는 것을 알 수 있다.

TU 블록 크기는 HM 15.0 부호화기 cfg 파일의 파라미터인 QuadtreeTULog2MaxSize와 QuadtreeTULog2MinSize를 통해 설정이 가능하다. 실제 TU 크기에 $\log_2$를 취한 값으로 설정하기 때문에 32×32 TU 크기에서 4×4 TU는 5에서 2의 값으로 설정한다. 〈표 2〉는 비교 기준(앵커, anchor)의 TU 크기가 32×32에서 4×4까지 가변될 수 있도록 설정했을 때, 비교 대상의 TU 크기를 16×16에서 4×4까지, 그리고 8×8에서 4×4까지 설정했을 때 얼마나 부호화 성능 저하(비트량 증가)가 발생되는지에 대한 실험 결과를 나타낸다. 〈표 2〉에서 보듯이 32×32 크기의 TU를 사용하지 않았을 때(최대 TU 크기가 16×16 인 경우) 3.3%의 성능저하만 발생하지만, 16×16 크기의 TU를 사용하지 않을 때(최대 TU 크기가 8×8 인 경우)는 11.8%의 큰 성능 저하가 발생됨을 알 수 있다. 결국, 현재 실험한 영상에서 16×16 TU가 성능에 큰 영향을 주는 것을 알 수 있다.

HM 15.0 부호화기 cfg 파일에서 CU를 기준으로 하는 TU의 깊이(depth) 값 설정은 화면간 모드일 경우 QuadtreeTUMaxDepthInter, 화면내 모드일 경우 QuadtreeTUMaxDepthIntra 파라미터를 통해 가능하다. 예를 들어 QuadtreeTUMaxDepthInter = 1 (또는, QuadtreeTUMaxDepthIntra = 1)인 경우 HM에 각 CU마다 변환 및 양자화의 율-왜곡값 비교를 통해 결정되는 최적의 TU 조합은 CU 크기와 동일한 크기의 TU만을 이용하여 결정한다. 〈표 3〉은 HM의 최대 TU 깊이인 3으로 설정하였을 때 대비 TU 깊이를 2와 1로 설정 했을 때의 성능 저하(Y-BDBR 증가)를 나타낸다.

〈표 3〉에서 확인할 수 있듯이 CU와 동일한 크기 TU

〈표 2〉 TU 크기에 따른 Y-BDBR

	16×16~4×4 TU	8×8~4×4 TU
Kimono	6.5%	21.7%
ParkScene	0.4%	2.6%
Cactus	2.1%	8.0%
BasketballDrive	4.3%	14.7%
Average	3.3%	11.8%

〈표 3〉 TU 깊이 조절 실험의 Y-BDBR

	TU 깊이 = 2	TU 깊이 = 1
Kimono	0.4%	0.9%
ParkScene	0.3%	1.0%
Cactus	0.2%	0.9%
BasketballDrive	0.4%	1.2%
Average	0.3%	1.0%

〈표 4〉 PU 모드 선택에 Y-BDBR 성능 평가

	AMP off	2N×2N Only
Kimono	0.5%	1.9%
ParkScene	0.6%	2.0%
Cactus	0.5%	2.7%
BasketballDrive	0.6%	2.3%
Average	0.6%	2.2%

만을 부호화에 사용(TU 깊이가 1인 경우만 사용)한다 하더라도 평균 1.0%의 성능 저하만 발생함을 알 수 있다. 따라서 TU 깊이를 1로 설정해도 큰 성능 저하가 없기 때문에 낮은 복잡도의 부호화기를 설계하는 경우에 합리적인 선택이 될 수 있다.

한편, PU는 본고의 Part I에서 설명했던 것처럼 각 2N×2N CU에 대하여 대칭형 형태의 2N×2N SKIP, 2N×2N Inter, 2N×N Inter, N×2N Inter, 2N×2N Intra, N×N Intra와 화면간 비대칭형 형태의 AMP(Asymmetric Motion Partition) 모드인 2N×nU, 2N×nD, nL×2N, nR×2N 모드가 있다. PU의 각 부호화 모드에 대한 성능을 알아보기 위하여 AMP 모드를 사용하지 않는 경우(AMP 모드 off시)와 대칭형 구조에서 정방형 2N×2N SKIP, 2N×2N Inter, 2N×2N Intra 모드만 사용했을 때의 성능 비교를 〈표 4〉에 제시 하였다.

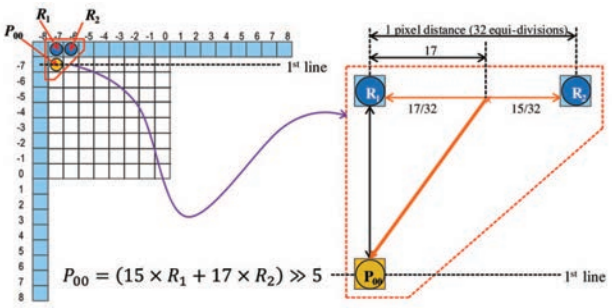
〈표 4〉에서, AMP 모드를 사용하지 않았을 때 평균 0.6% 정도의 성능 저하가 발생한 반면, AMP 모드 뿐만 아니라 2N×N Inter, N×2N Inter 모드까지 사용하지 않았을 경우에는 약 2.2%의 성능 저하가 발생한 것을 알 수 있다. 물론 실험 영상이 달라질 경우 약간은 다른 결과가 예상되지만 전체적인 경향성은 PU의 다양한 모드를 사용하여 율-왜곡 비교 과정을 통한 성능 개선보다는 다양한 크기의 2N×2N SKIP 모드 및 2N×2N Inter 모드, 2N×2N Intra 모드만을 사용하여 부호화 과정을 수행하는 것이 성능에 더 큰 영향을 주는 것으로 볼 수 있다.

화면내 예측 모드에서는 35개 방향성 모드에 대해 간단한 SATD 율-왜곡 값으로 소수의 모드를 선별하고,이에 대해 변환 및 양자화가 필요한율-왜곡 값으로 최종 모드를 결정한다.

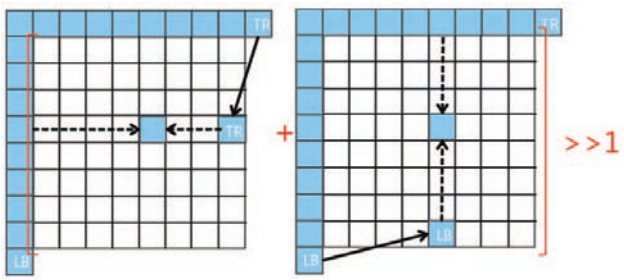
2. 화면내 예측 모드

HEVC의 화면내 예측 모드는 본고의 Part I에서 설명하였던 것처럼 총 33개의 방향성 모드에 모드 0인 planar 모드와 모드 1인 DC 모드를 더하여 총 35개의 모드로 구성되어 있다. 〈그림 2〉는 33개의 방향성 모드 중 31번 방향성 모드를 사용하여 예측 데이터를 생성하는 예를 나타낸다. HEVC 화면내 방향성 예측은 현재 블록 내의 화소값을 예측하기 위해 현재 블록 경계에 인접한 부호화된 주변 화소들에 대해 각 두 화소간 거리를 32 등분하여 선형 예측을 수행하게 된다. 31번 방향성 모드의 경우 한 화소를 32 등분 단위로 나누었을 때 오른쪽 17 만큼의 선형 예측을 통해 예측 데이터를 생성하게 된다.

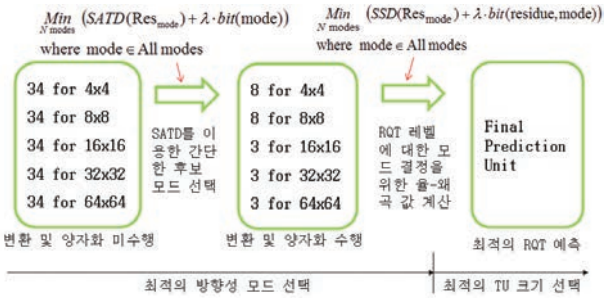
HEVC의 planar 모드는 〈그림 3〉과 같이 위 오른쪽(TR: Top Right)에 있는 인접블록 경계 화소 값을 이용하여 좌측 라인과 선형 예측을 통해 예측 데이터를



〈그림 2〉 화면내 예측 데이터의 생성 예



〈그림 3〉 HEVC planar 화면내 예측 모드의 데이터 생성 과정



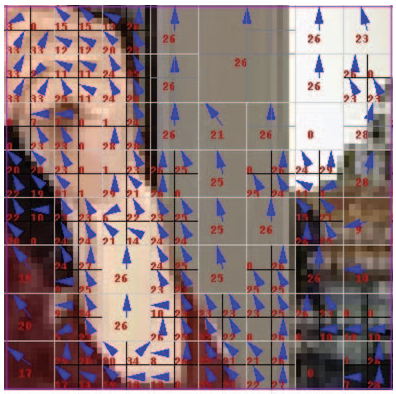
〈그림 4〉 HM에서 화면내 예측 부호화 수행 과정

만들고, 왼쪽 아래(LB: Left Bottom)에 있는 인접블록 경계 화소 값을 이용해 예측 데이터를 만들어서 두 예측 데이터 블록의 평균을 planar 모드의 예측 데이터로 생성한다. 이러한 예측 데이터 생성은 병렬처리가 용이하기 때문에 병렬 모드가 지원되는 하드웨어에서 빠른 속도의 처리가 가능하다.

HEVC의 화면내 예측 부호화 수행 시 총 35개의 부호화 모드로부터 최적의 부호화 모드를 결정하기 위해, 모든 CU 크기의 조합에 대한 변환 및 양자화를 비롯한, 역변환, 역양자화, CABAC를 통한 율-왜곡 값 계산 및 비교는 매우 높은 연산 복잡도를 초래한다. 따라서 현재 HM의 화면내 예측 모드에서는 〈그림 4〉와 같이 35개의 방향성 모드에 대하여 SATD 및 모드 비트 정보만을 이용하여 간단한 SATD 율-왜곡 값을 계산하고, 이때 64×64에서 16×16 화면내 예측모드는 SATD 율-왜곡값이 가장 작은 3개의 모드를, 8×8에서 4×4 화면내 예측모드는 SATD 율-왜곡값이 가장 작은 8개의 모드를 이용해 변환 및 양자화가 필요한 율-왜곡 값을 통하여 최종적인 모드를 결정하게 된다. 방향성 모드가 결정이 되고나서 최종적으로 HEVC의 화면내 예측 모드를 위한 TU 크기를 결정하게 된다.

〈표 5〉 화면내 예측 모드 off시 Y-BDBR 성능

	Y-BDBR
Kimono	2.0%
ParkScene	1.3%
Cactus	4.7%
BasketballDrive	12.2%
Average	5.1%



〈그림 5〉 화면내 슬라이스에서 휘도 신호의 방향성 모드 선택 예[3]

〈표 6〉 B-슬라이스 부호화시 화면내 예측 모드 선택 비율

영상	QP	화면내 예측 모드 선택 비율
ParkScene	*27*	24.5%
ParkScene	*32*	28.0%
BasketballDrive	*27*	38.1%
BasketballDrive	*32*	34.2%

〈그림 5〉는 *BQMall* 영상을 QP = 27로 부호화한 후, 화면내 슬라이스(I-Slice)에서 화면내 예측 모드의 최종 선택된 방향성 모드를 도시하였다. HEVC의 화면내 예측 모드는 이전 AVC에 비해 다양한 방향성 모드 및 다양한 크기의 planar 모드가 추가되어 화면내 블록에 잘 맞는 예측이 가능하다.

HEVC의 화면내 예측 모드가 화면간 예측 부호화를 위한 P 또는 B 슬라이스에서 부호화 성능에 어느 정도의 영향을 주는지 실험하였다. 이를 위하여 화면간 예측 부호화를 수행함에 있어 B 슬라이스만 사용하는 조건에서 화면내 예측 모드를 사용하지 않도록 HM 15.0 소드 코드를 수정하여 실험한 결과를 〈표 5〉에 제시하였다. 〈표 5〉에서 확인할 수 있듯이 HEVC 화면내 예측모드는 영상의 특성에 따라 부호화 성능이 크게 가변적이다.

한편, 화면내 예측 모드를 B 슬라이스 부호화에 적용하였을 경우 화면내 예측 모드가 선택 비율을 〈표 6〉에 제시 하였다. 〈표 6〉의 실험 결과는 B 슬라이스 부호화시 화면내 예측 모드 적용 여부에 따라 성능에 큰 영향이 없는 *ParkScene* 영상과 큰 성능 차이를 보이

는 *BasketballDrive* 영상에 대해 QP = 27과 32를 적용한 결과이다.

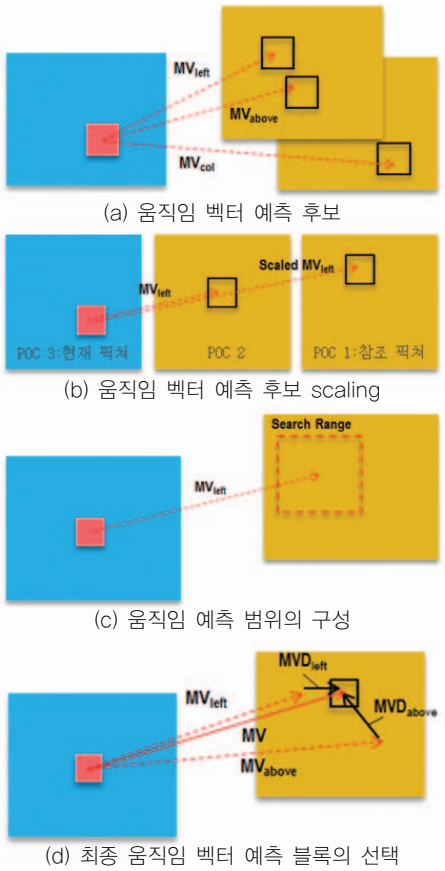
〈표 6〉에서 보듯이 *ParkScene* 영상과 *BasketballDrive* 영상에 대해 서로 다른 QP 값들에 대해 약간의 화면내 예측 모드의 선택 비율에 차이가 있으나 대체로 큰 차이를 보이지는 않는다. 이러한 결과는 입력 영상의 특성상 *ParkScene* 영상과 *Kimono* 영상의 경우에는 패턴이 거의 없는 랜덤한 특징을 보이는 경우로서 화면내 예측이 성능에 거의 영향을 주지 않는 반면, *Cactus* 영상과 *BasketballDrive* 영상의 경우에는 화면 내에 공간적 유사도가 높은 영역들이 많이 존재하여 화면내 예측 성능이 전체 부호화 성능에 크게 영향을 주는 것을 알 수 있다.

3. 화면간 예측 모드

3.1 Advanced Motion Vector Prediction (AMVP)

(1) 움직임 예측 초기점 선택

〈그림 6〉은 HEVC에서 움직임 벡터 예측 방법을 나타낸다. 〈그림 6(a)〉는 현재 블록의 왼쪽과 위쪽 블록의 움직임 벡터(MV)를 움직임 벡터 예측 값으로 모두 사용 가능한 경우로서 공간 움직임 벡터 예측(MVP) 후보(candidate)가 된다[4]. 좌측이나 상단 둘중에 하나라도 사용 가능하지 않은 경우 〈그림 6(a)〉처럼 collocated 블록의 MV가 움직임 벡터 예측(MVP) 후보(candidate)가 된다. 이때 collocated 블록은 바로 직전에 부호화된 참조화면에서 선택된다. 본 논문에서는 편의를 위해 공간적 움직임벡터 예측 후보가 모두 사용 가능하다고 가정한다. 만약 현재 블록의 참조 픽쳐와 공간 MVP 후보의 참조 픽쳐가 다른 경우에는, 현재 픽쳐의 참조 픽쳐와 후보 블록의 참조 픽쳐의 순서(Picture Order Count: POC) 차이만큼 움직임 벡터를 스케일링하고 이 스케일링된 움직임 벡터를 후보 MVP로 사용한다. 따라서 공간 MVP 후보는 좌측 인접 블록의 MV, 좌측 인접 블록의 스케일링된 MV, 상단 인접 블록의 MV, 상단 인접 블록의 스케일링된 MV가 된다. 〈그림 6(c)〉에서 보듯이 공간 MVP 후보가 가리키는 블록과 현재 블록의 왜곡값(Sum of Absolute Difference: SAD)을 이용하여 가장 작은 SAD값을 갖는 공간 MVP 후보를 선택하고 그 MVP가 가리키는 지점을 중심으로 움직임 예측 범위(Search Range: SR)를 구성한다.



〈그림 6〉 움직임 벡터 예측 블록의 선택 및 움직임 예측

Merge 모드는 공간 MVP(Motion Vector Predictor) 후보 4개와 시간 MVP 후보 1개를 참조 후보 블록으로 사용하고, 이중에 가장 작은 율-왜곡 값을 갖는 참조 후보를 참조 블록으로 사용한다.

(2) 움직임 벡터 예측 후보 선택

〈그림 6(d)〉에 보이듯이, SR 내에서 움직임 예측을 수행하고, 최소의 율-왜곡 값을 갖는 위치를 현재 블

록의 최종 MV로 결정한다. 이 MV는 그대로 전송되지 않고 전송되는 정보량을 최소화하기 위해 주변 블록의 MVP 후보와의 차분 벡터(MVD)가 전송된다. 이를 위해 움직임 초기점 선택에 사용된 MVP 후보들 중 가장 작은 MVD를 발생하는 MVP 후보를 최종 MVP로 선택하고 그 MVP 위치(MVP index)값과 움직임 벡터 차분을 전송한다.

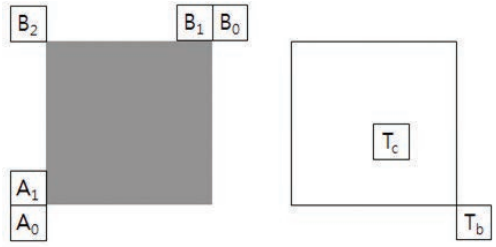
3.2 Merge 모드/Merge SKIP 모드

(1) Merge 모드

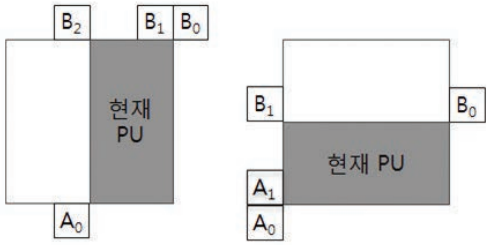
Merge 모드는 2N×2N PU 뿐만 아니라 N×2N, 2N×N, 2N×nU, 2N×nD, nL×2N, 그리고 nR×2N 크기 등 모든 PU 크기에 적용 가능한 모드이다. 〈그림 7〉은 각 PU 크기에 따른 Merge 모드의 공간 MVP 후보 블록을 나타낸다. 흥미로운 것은 2N×2N PU(〈그림 7(a)〉의 어두운 블록)와 여타 비정방형 PU(〈그림 7(b)〉의 어두운 블록)는 참조하는 인접 블록이 다르다. 이는 각 CU 복호화 과정에 병렬처리가 가능하도록 참조 블록을 제한했기 때문이다[5]. 〈그림 7(b)〉와 같이 인접하는 PU를 참조하지 않도록 강제함으로써 동일 CU내의 PU들의 의존성을 차단하여 각 PU의 병렬처리를 가능하게 하여 결국 CU 병렬처리의 속도를 향상 시킬 수 있는 구조를 고안하였다.

Merge 모드는 AMVP와 다르게 공간 MVP후보 4개와 시간 MVP 후보 1개를 포함하여 총 5개의 후보 블록을 참조 후보 블록으로 사용하고, 이중에 가장 작은 율-왜곡 값을 갖는 참조 후보를 참조 블록으로 사용한다. 〈그림 7〉에서 보이듯이 Merge 모드는 A_1, B_1, B_0, A_0, B_2의 순서로 사용 가능 여부를 확인하고 이 중 사용가능한 순서대로 4개의 공간 후보 블록을 선택한다. 또한 시간적 후보 블록도 T_b, T_c 순서로 사용가능 여부를 확인하고, 우선적으로 T_b를 후보 블록으로 사용하며 T_b가 사용가능하지 않을 때 T_c를 후보 블록으로 고려한다. 따라서 AMVP의 참조 블록 정보를 전송하기 위해서는 1 비트만 사용되지만, Merge 모드의 참조 블록 정보를 전송하기 위해서는 2 비트가 사용된다.

Merge 모드와 AMVP와의 또 다른 차이는 MVD, 참조 화면 번호, 참조 방향 정보의 전송 여부이다. Merge 모드는 주변 블록의 MVD, 참조 화면 번호, 참조 방향을 그대로 이용하고 별도의 움직임 예측 과정 없이 인접 블록과 MVD, 참조 화면 번호, 참조 방향이 지시하고 있는 블록을 참조 블록으로 사용하여 움직임 예측을 진행하고 이 참조 블록과 현재 블록간의 잔차 신호의 합으로 예측 블록을 구성한다.



(a) 2N×2N PU의 MVP 생성을 위한 참조 가능 인접 공간/시간 블록



(b) 2N×N, N×2N, AMP PU 의 MVP 생성을 위한 참조 가능 인접 공간 블록

〈그림 7〉 PU 크기에 따른 Merge 모드의 참조 블록

(2) Merge SKIP 모드

Merge SKIP 모드는 2N×2N PU에만 적용 가능한 모드이다. Merge SKIP 모드는 후보 블록의 MV, 참조 화면 번호, 참조 방향을 동일하게 사용하며, 움직임 예측을 추가적으로 진행하지 않고 인접 블록과의 MVD, 참조 화면 번호, 참조 방향으로 유도된 참조 블록을 그대로 예측 블록으로 사용한다. 즉, AMVP나 Merge 모드가 참조 블록과 잔차 신호를 이용하여 예측 블록을 구성하는데 비해, 잔차 신호의 전송 없이 참조 블록을 그대로 예측 블록으로 사용함으로써 잔차 신호 전송에 의해 발생되는 정보량을 감소시켰다. Merge SKIP 모드일 경우 SKIP flag 정보만을 전송하고 SKIP 모드가 아닐 때에만 Merge flag를 전송하기 때문에 전송되는

〈표 7〉 각 예측 모드의 전송 정보

	MVD	Ref idx	Ref List	잔차신호
AMVP	o	o	o	o
Merge	x	x	x	o
Merge SKIP	x	x	x	x

비트량을 절약한다. AMVP, Merge SKIP, Merge 모드들에 대해 각 전송되는 정보를 비교하면 〈표 7〉과 같다.

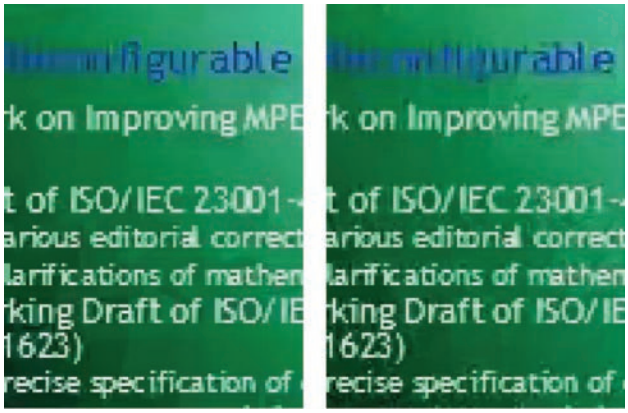
4. 변환 및 양자화

HM 15.0에서는 부호화기의 성능 향상을 위하여 AVC에서부터 개발된 RDOQ(Rate- Distortion Optimized Quantization) 방법을 HM에 맞도록 적용하여 부호화 성능을 개선하였다. 이를 위하여 변환 및 양자화 이후 양자화 계수 값을 올림과 내림, 양자화 계수가 3 보다 작은 경우 0으로 하는 3가지 경우에 대한 율-왜곡 값을 계산하여 비교함으로써 최종 양자화 계수 값이 결정된다. 〈표 8〉은 HM 15.0 부호화기에서 RDOQ를 사용하지 않았을 때 저하된 Y-BDBR 성능이다. 〈표 8〉에서 볼 수 있듯이 HM 15.0 부호화기의 RDOQ는 간단한 연산을 통해 부호화를 수행함에도 불구하고 평균 6%의 큰 부호화 성능 향상을 가져온다. 하지만, RDOQ를 실행하기 위하여 왜곡값 뿐만 아니라 율 값을 계산해야하기 때문에 모드 결정시 CABAC을 사용하지 않는 저복잡도 부호화기의 경우 〈표 8〉과 같은 성능을 기대하기는 힘들다.

또한, HEVC의 화면내 또는 화면간 모드에서는 4×4 블록에서 변환을 수행하지 않는 새롭게 적용된 TSM(Transform Skip Mode) 모드를 지원한다. TSM 툴의 경우 텍스트 영상과 같이 블록 내 텍스처가 복잡한 신호의 경우 변환을 수행하지 않는 것이 부호화 성능을 향상하게 된다. TMS 모드는 텍스트 영상인 HM의 Class F 영상에 대해 부호화 성능 및 주관적 화질을 크게 개선하였다. TSM의 성능을 확인하기 위하여 Class F 영상인 *BasketballDrillText*, *ChinaSpeed*, *SlideEditing*, *SlideShow*에 대하여 TSM을 사용하지 않았을 때의 객관적, 주관적 성능을 확인하기 위한 실험을 수행하였다. 〈표 9〉는 TSM을 사용하지 않았을 때 Y-BDBR 증가를 나타낸다.

RDOQ는 간단한 연산을 통해 부호화를 수행함에도 불구하고 평균 6%의 큰 부호화 성능 향상을 가져온다.

〈표 9〉의 실험 결과에서 확인할 수 있듯이 평균 8.5%의 Y-BDBR 성능 저하(비트율 증가)가 발생하며, 이는 화면내 존재하는 텍스트 데이터가 변환 및 양자화를 통해 데이터가 손실되어 화면 왜곡이 크게 발생



(a) TSM 수행 (b) TSM 미수행

〈그림 8〉 TSM 적용/미적용 시 주관적 화질 비교

〈표 8〉 RDOQ 오프 실험 결과 Y-BDBR 성능

	Y-BDBR
Kimono	5.8%
ParkScene	4.4%
Cactus	6.3%
BasketballDrive	7.5%
Average	6.0%

〈표 9〉 TSM 오프시 Y-BDBR 성능

	Y-BDBR
BasketballDrillText	0.3%
ChinaSpeed	14.5%
SlideEditing	13.6%
SlideShow	5.5%
Average	8.5%

하였으며, 이를 예측데이터로 사용하는 이후 블록들의 화질이 점차적으로 저하되어 전체적인 성능이 크게 저하 되는 결과를 초래 하였다. 〈그림 8〉은 TSM의 수행 여부에 따른 화면의 왜곡을 주관적으로 비교하기 위한 그림이다. 〈그림 8〉에서와 같이 TSM을 사용하지 않는 블록에서는 텍스트에 블러링(blurring)이 크게 발생 하여 주관적 화질 저하가 심함을 확인할 수 있다.

5. 엔트로피 부호화 (Entropy Coding)

HEVC의 엔트로피 부호화 과정에서 변환계수 부호화 부분은 AVC와 유사하지만 HEVC의 변환 블록의 크기가 매우 크기 때문에 이에 적합한 새로운 변화들이 도입되었다[6-7].

(1) 마지막 계수 (last significant coefficient)

HEVC에서는 변환 블록에 대하여 0이 아닌 변환 계수를 알려주는 4×4 크기의 서브블록의 중요도 맵(significance map)을 전송하기 전에 0이 아닌 마지막 계수의 수평 및 수직 좌표가 부호화된다. 좌표 값은 우선 접두사(prefix)와 접미사(suffix)로 각각 이진화 된다. 신택스 요소 *last_significant_coeff_x_prefix*와 *last_significant_coeff_y_prefix*는 *truncated unary*방식으로 이진화 되고 나머지 *last_significant_coeff_x _suffix*와 *last_significant_coeff_y_suffix*는 고정 길이로 이진화된다.

> 디블록킹 필터는 블록 경계강도(boundary strength, Bs)와 경계 주변의 화소 값을 이용하여 계산된 값을 특정 임계치와 비교하여 필터 적용 여부 및 필터 강도를 결정한다.

(2) 중요도 맵 (significance map)

AVC와 마찬가지로 HEVC에서도 CBF (coded block flag)가 해당 변환 블록에 0이 아닌 계수가 남아있는가를 표시한다. 모든 계수가 0이면 CBF값이 0이 된다. 마지막 계수는 이미 알고 있으므로 마지막 계수까지 스캔하여 중요도 맵이 부호화 된다. HEVC에서는 발전된 예측 기술들로 인해 많은 계수들이 0의 값을 갖게 되어 중요도 맵이 매우 성긴(sparse) 형태로 나타난다. 이를 효율적으로 다루기 위해 변환 블록을 두 레벨로 나누어서 부호화 한다.

(3) 계수 레벨 (coefficient level)

HEVC에서 계수 레벨을 부호화 하는 것은 AVC에서 하던 것과 일부 유사하지만 몇 가지 차이가 있다. 계수의 레벨이 1 보다 큰지 (coeff_abs_level_greater1_flag), 2 보다 큰지(coeff_abs_level_greater2_flag)를 부호화하고 나머지 값(coeff_abs_level_remaining)을 부호화한다. 이와 같이 부호화를 하면 스캐닝 횟수를 줄여서 데이터 처리량 및 속도를 향상시킬 수 있다. 1, 2 보다 큰 나머지 값에 대해서는 Golomb-Rice 코드와 Exp-Golomb 코드로 이진화 된다.

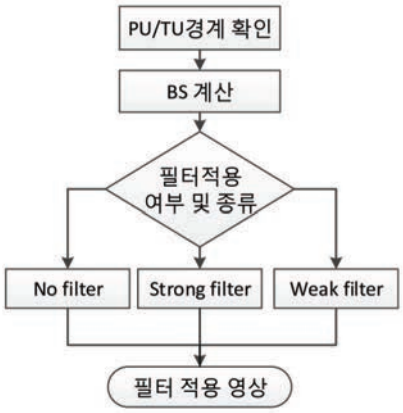
(4) 부호 데이터 (sign data)

HEVC에서 계수의 부호는 각 심볼이 동일한 확률로 발생하며 서로 연관이 없다고 가정하여 부호화된다. 신택스 요소 *coeff_sign_flag*는 전체 비트스트림 양에서 의미 있는 비율을 차지한다. 이에 HEVC에서는 부호 데이터 은폐(sign data hiding, SDH)라고 알려진 방법이 압축 효율을 더 향상시키기 위해 적용되었다. 부호화된 계수의 위치와 개수에 따라 부호비트가 조건적으로 부호화된다. SDH가 이용될 때, 4×4 서브블록에 0이 아닌 계수가 적어도 두 개 있고, 첫 번째 계수와 마지막 계수의 스캐닝 위치가 3 보다 클 때, 첫 번째 계수의 부호 비트는 계수 크기 합의 패리티로부터 추론된다. 그 외의 경우에 부호 비트는 평범하게 부호화 된다.

이 같은 방식으로 HEVC에서 변환 계수 부호화를 효과적으로 적용하게 되었고 이전 표준에 비해 많은 이점을 갖게 되었다.

6. 디블록킹 필터

디블록킹 필터는 블록 경계강도(boundary strength, Bs)와 경계 주변의 화소 값을 이용하여 계산된 값을 특



〈그림 9〉 HEVC 디블록킹 필터 적용 순서

〈표 10〉 Bs값의 결정 조건

조건	Bs
적어도 한 블록이 화면내 예측	2
적어도 한 블록에 0이 아닌 변환계수가 존재	1
두 블록의 MV가 정수 화소 단위로 1이상 차이	1
두 블록이 상이한 참조영상에서 움직임 예측을 하거나 움직임 벡터의 개수가 다름	1
그 외의 경우	0

정 임계치와 비교하여 필터 적용 여부 및 필터 강도를 결정한다. 〈그림 9〉는 디블록킹 필터의 순서도를 나타낸다.

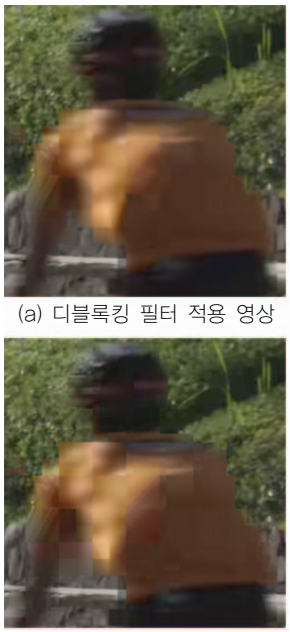
Bs값은 0-2의 값을 갖는데, 경계 양쪽의 블록 특성에 따라 결정된다. 〈표 10〉은 Bs값이 결정되는 조건을 나타낸다[8].

한 경계에 대해 Bs를 판단할 때는 〈표 10〉에 나오는 조건을 위에서부터 아래로 내려오면서 판단한다. 만약 위의 조건이 참으로 판단되면 아래의 조건들은 판단하지 않는다. Bs값이 0인 경계에 대해서는 필터가 적용되지 않으며 Bs 값이 1, 2인 경우는 그 값에 따라서 임계값이 다르게 적용되어 필터의 강도를 선택하는데 영향을 준다. 필터의 강도를 결정하는데 이용되는 임계값은 β와 t_C값으로 이는 QP에 대한 함수로서 참고문헌 [9]에 표로 제시되어 있다.

디블록킹 필터는 실제 부호화 효율에도 많은 영향을 준다. 〈표 11〉은 디블록킹 필터를 적용하지 않았을 경우 성능을 나타낸다. 〈표 11〉은 Random Access 구조로 각 영상을 100 프레임을 부호화하여 비교한 결과이다. 디블록킹 필터를 적용하지 않을 때 평균 4.4%의 성능 저하가 발생하는 것을 표를 통해 확인할 수 있다. 더욱 중요한 사실은 디블록킹 필터가 주관적인 화질 향상에도 큰 기여함을 알 수 있다.

〈표 11〉 디블록킹 필터 미적용시 BDBR 성능

	BD-rate (piecewise cubic)		
Sequence	Y	U	V
Kimono	8.0%	5.0%	3.8%
ParkScene	1.9%	1.0%	-0.2%
Cactus	3.2%	3.4%	3.7%
BasketballDrive	4.4%	4.0%	4.9%
Average	4.4%	3.3%	3.1%



(a) 디블록킹 필터 적용 영상

(b) 디블록킹 필터 미 적용 영상

〈그림 10〉 ParkScene 영상에 대해 QP 37에서 RA 구조로 부호화한 영상의 56번째 프레임의 디블록킹 필터 적용 및 미적용시의 화질 비교

〈그림 10〉은 디블록킹 필터를 적용한 복원영상과 적용하지 않은 복원영상을 비교하여 나타낸 것이다. 그림에서 볼 수 있듯이 디블록킹 필터를 적용하지 않은 영상의 TU와 PU경계에서 발생하는 블록화 현상이 디블록킹 필터를 적용하면 많이 완화됨을 확인할 수 있다.

7. SAO (sample adaptive offset) 필터링

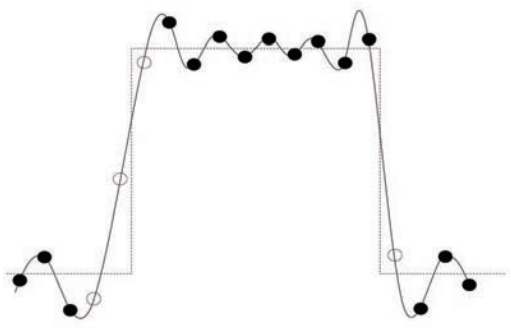
SAO는 모든 화소를 4개의 카테고리로 분류하여 각 카테고리마다 적응적으로 오프셋을 가감하여 원본영상과 복원영상의 차이를 줄이는 필터링이다. 화소를 카테고리로 분류하는 기준에 따라 EO(edge offset, SAO type 1-4)와 BO (band offset, SAO type 5), 두 개의 클래스로 분류할 수 있다. 여기서 EO의 경우, 영상을 압축할 때 변환 및 양자화로 인해 생기는 진동무늬왜곡(ringing) 현상을 효과적으로 완화시킬 수 있다. 〈그림 11〉은 영상의 에지부분에서 발생할 수 있는 진동무늬왜곡 현상을 1차원 단면으로 나타낸 것이다. 그림에서 검은색으로 채워져 있는 화소들은 SAO의 적용으로 원본영상과 유사한 밝기 값을 갖게 된다.

HEVC는 기존의 AVC 대비 고정된 QP를 사용하였을 때 평균 42.2%의 비트율 효율 향상이 있었으며, 영상이 복잡할수록 큰 부호화 성능 개선을 보였다.

한편 BO 클래스의 경우 특정 밝기 구간의 값이 양자화 등으로 인해 동일하게 밝아지거나 어두워진 경우 효과적으로 화질을 향상시킨다.

〈그림 12〉는 영상에 SAO가 적용되었을 때와 적용되지 않았을 때를 비교하여 나타낸 것이다. 그림에서 볼 수 있듯이 SAO가 적용되지 않았을 때는 영상의 에지 주변에서 진동무늬왜곡 현상으로 인해 많은 화질 저하가 발생하지만 SAO를 적용한 경우 에지 주변이 선명하게 복원되는 것을 확인할 수 있다.

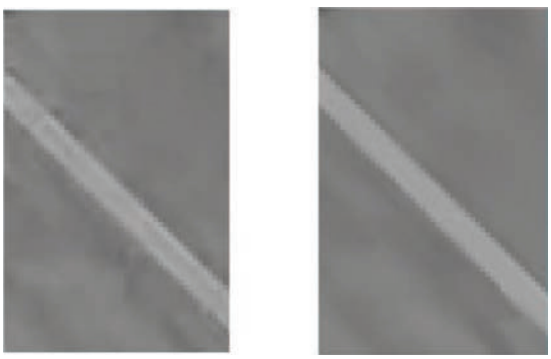
부호화기 관점에서는 현재 처리하는 CTU에 대해 SAO를 통해 어떤 클래스 및 타입으로 화소를 분류할 것인지와 각 카테고리별로 얼마만큼의 오프셋을 적용해야 하는지를 결정해야 한다. 이를 위해 율-왜곡 최적화를 통하여 최적의 SAO 타입과 오프셋 값을 결정하는데 왜곡을 측정하는데 있어서 고속화 알고리즘이 적용되어 실행 속도 역시 향상되었다.



〈그림 11〉 변환 및 양자화로 인해 발생하는 링잉 현상, 점선이 원본이고 실선과 점이 복원영상[10]

8. AVC와 HEVC 부호화 성능 비교

AVC와 HEVC의 부호화 성능을 비교하기 위하여 HEVC 참조 소프트웨어인 HM 15.0과 JM 참조 소프트웨어인 JM 18.6[11]을 사용하였다. M 15.0은 임의 접근 cfg 파일을, JM 18.6은 HM 15.0과 비교를 위해 제공되는 cfg 파일 (encoder_JM_RA_B_HE.cfg)을 이용하여 실험을 하였다. 각 참조소프트웨어의 QP 값은 HM의 기본 실험 조건인 22, 27, 32, 37에 대하여 부호화를 수행 하였다. 〈표 12〉는 HM 15.0과 JM 18.6



(a) SAO 미적용 영상 (b) SAO 적용 영상

〈그림 12〉 SAO 적용시와 미적용시의 주관적 화질 비교 – *BasketballDrillText* 영상, QP 32, 100번째 프레임 (HM 12.0)

〈표 12〉 HM 15.0과 JM 18.6의 비교실험 결과

	Y-BDBR
Kimono	-48.4%
ParkScene	-35.7%
Cactus	-38.3%
BasketballDrive	-46.4%
Average	-42.2%

의 부호화 성능 비교 결과이다.

〈표 12〉에서 볼 수 있듯이 JM18.6 대비 HM 15.0 부호화 결과를 비교하였을 때 HM 15.0이 Y-BDBR에서 평균 42.2%의 비트가 절약 되었고, *Kimono* 영상에서 48.4%의 최대 비트 감소를 보였다. 결국, 고정된 QP로 실험을 하였을 때 거의 50%의 비트가 감소하였으며 영상이 빠르고 복잡한, *Kimono* 와 *BasketballDrive* 영상에서 HM 15.0이 JM 18.6에 비하여 큰 부호화 성능 개선을 보였다.

Ⅳ. 결 론

본고에서는 최근 표준화가 완성된 차세대 비디오 압축 부호화 규격인 HEVC의 기본 요소 기술에 대한 특징 및 성능에 대해 소개 하였다. 특히, 기존의 AVC 대비 HEVC가 고정된 QP를 사용하였을 때 평균 42.2%의 비트율 성능을 개선하였으며, 영상이 복잡할수록 부호화 성능이 큰 개선을 보였다. HEVC는 기존의 비디오 부호화 표준들과 비교하여 압축 성능을 크게 향상시켰으며 향후 도래하는 UHDTV 방송이나 3D 비디오 부호화 응용에 널리 활용될 것으로 예상된다. 뿐만 아니라, 스마트폰 등 모바일 응용분야와 IPTV 등 스트리밍 분야에도 널리 사용될 것으로 예상된다.

참 고 문 헌

[1] HM reference software,[Online]. Available: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware.

[2] JCT-VC, "Common test conditions and software reference configurations," JCTVC-L1100, Geneva, Switzerland, Jan. 2013.

[3] Dual HEVC Bitstream Analyzer, http://www.hanulsoftware.com/site/product-dhba.php

[4] Lin J. L., et al., "Motion Vector Coding in the HEVC Standard," IEEE Trans. Signal Processing, vol. 7, No. 6, pp 957-968, Dec. 2013.

[5] 조현오, 심동규, "고효율 영상 부호화 기술 HEVC 표준 기술의 이해".

[6] Sullivan, G. J., et al., "Overview of the High Efficiency Video Coding (HEVC) Standard," IEEE Trans. Circuits Syst. Video Technol., vol. 22, No. 12, pp. 1649-1668. Dec. 2012.

[7] Sole, J., et al., "Transform Coefficient Coding in HEVC," IEEE Trans. Circuits Syst. Video Technol., vol. 22, No. 12, pp. 1765-1777. Dec. 2012.

[8] Andrey N., et al., "HEVC Deblocking Filter," IEEE Trans. Circuits Syst. Video Technol., vol. 22, No. 12, pp. 1746-1754. Dec. 2012.

[9] JCT-VC, "High Efficiency Video Coding (HEVC) text specification draft 10 (for FDIS & Last Call)," JCTVC-L1003_v34, Geneva, Switzerland, Jan. 2013.

[10] Fu, C., et al., "Sample Adaptive Offset in the HEVC Standard," IEEE Trans. Circuits Syst. Video Technol., vol. 22, No. 12, pp. 1755-1764. Dec. 2012.

[11] JM reference software, [Online]. Available: http://iphome.hhi.de/suehring/tml/ download/jm18.6.zip.

김 재 일

• 2006년 2월 한국기술교육대학교 정보통신공학과 전자공학 전공 학사
• 2006년 3월~현재 한국과학기술원 정보통신공학과 석박사 통합과정

〈관심분야〉
비디오 부호화, 인지 비디오 부호화 기술, 패턴 인식, 방송시스템

안 상 수

- 2006년 8월 충남대학교 전기전자전파전공 학사
- 2007년 3월~현재 한국과학기술원 정보통신공학과 석박사 통합과정

〈관심분야〉
비디오 부호화, 율-왜곡값 예측, 패턴 인식

김 대 은

- 2011년 2월 고려대학교 전기전자전파공학부 학사
- 2014년 2월 한국과학기술원 전기 및 전자공학과 석사
- 2014년 3월~현재 한국과학기술원 전기 및 전자공학화 박사과정

〈관심분야〉
비디오 부호화, 인지 비디오 부호화

김 문 철

- 1989년 2월 경북대학교 전자공학과 학사
- 1992년 12월
  University of Florida, USA, Electrical & Computer Engineering 석사
- 1996년 8월
  University of Florida, USA, Electrical & Computer Engineering 박사
- 1997년 1월~2001년 2월
  한국전자통신연구원 방송미디어연구부 선임연구원
- 2001년 2월~2008년 2월
  한국정보통신대학교 공학부 조교수/부교수
- 2009년 3월~현재 한국과학기술원 전기및전자공학과 부교수/정교수

〈관심분야〉
2D/3D (인지) 비디오 부호화, 통계적 기계 학습, 패턴인식, 영상 분석 및 이해