HEVC screen content coding extensions 표준화 동향


양 승 하
성균관대학교

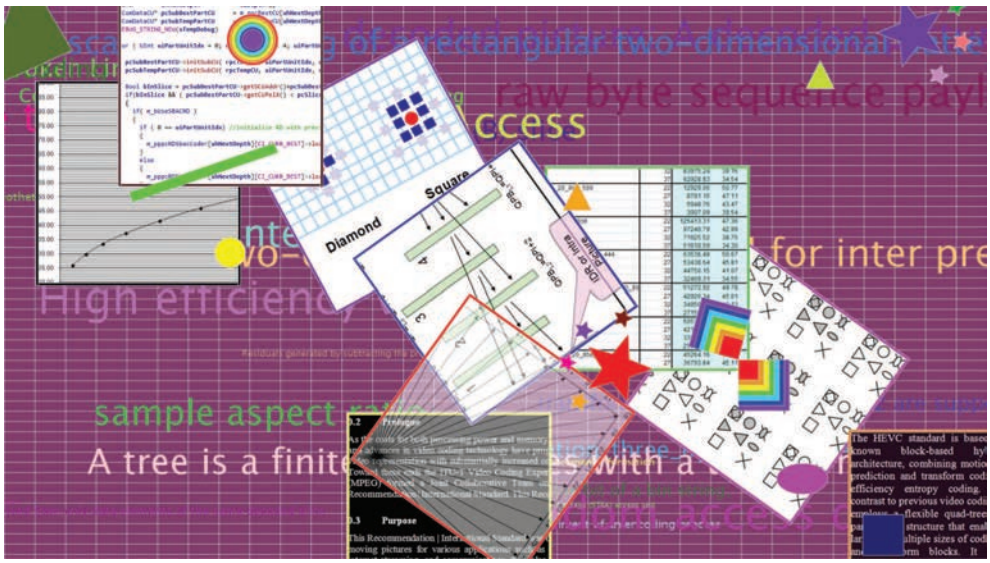

심 혁 재
성균관대학교


전 병 우
성균관대학교

Ⅰ. 서 론

종래의 영상 압축 표준은 주로 카메라로 촬영된 자연 영상의 효율적인 부호화에 중점을 두어 왔다. 하지만, HEVC (High Efficiency Video Coding) 버전 1[1-2]의 표준화 도중에 스크린 콘텐츠(screen content)라는 새로운 콘텐츠의 효율적인 부호화 방법의 필요성을 인지하게 되었다. 스크린 콘텐츠란 컴퓨터를 사용해 생성된 인공적인 영상을 말하는데, 좀 더 구체적으로 텍스트, 컴퓨터 그래픽스, 그래프 등을 포함하는 영상 (〈그림 1〉 참고)을 예로 들 수 있다. HEVC 버전 1의 표준화 과정의 마지막 단계 진행 중에, 향후 스크린 콘텐츠가 많이 사용될 것으로 예상되면서 스크린 콘텐츠를 고려한 새로운 기술들이 제안되기 시작하였으나, 버전 1의 기술 표준 확정을 늦추는 대신, 4×4 변환 크기 블록에 대하여만 변환 생략(transform skip)을 선택적으로 적용하는 기술만을 채택하는 것으로 버전 1이 마무리 되었다. 한편 변환 생략 기술만으로는 스크린 콘텐츠의 특수성을 충분히 고려하지 못하기 때문에, 2014년 3월 스페인 발렌시아에서 개최된 제 17회 JCT-VC (Joint Collaborative Team on Video Coding) 표준화 회의부터 HEVC SCC(screen content coding)[3] 표준화 프로젝트가 본격적으로 시작되었다. 한편, HEVC SCC는 앞서 표준화가 시작된 HEVC REXt(range extensions)[4]를 토대로 개발되고 있기 때문에, HEVC REXt에 채택된 기술은 현재의 HEVC SCC에서도 지원되는 특징을 가진다.

스크린 콘텐츠란 컴퓨터를 사용해 생성된 인공적인 영상을 말하는데, 예로 텍스트, 컴퓨터 그래픽스, 그래프 등이 있다.



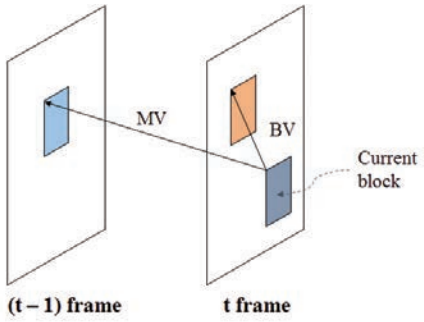
〈그림 1〉 HEVC SCC에서 사용하는 스크린 콘텐츠 실험영상의 예 (sc_flyingGraphics, 1920x1080, 첫 번째 프레임)

종래의 비디오 부호화 표준은 큰 범주에서 두 예측 방법을 통해 영상의 중복성(redundancy)을 제거한다. 첫 번째는 화면 내 예측 (intra prediction)을 통한 중복성 제거이다. 이 방법은 인접블록의 복원 화소 값을 사용하여 현재 블록의 화소 값을 예측한 후 현재 블록에서 감산하도록 하여 영상의 공간적 중복성 제거를 그 목적으로 한다. 두 번째는 화면 간 예측 (inter prediction)을 통한 중복성 제거이다. 이 방법은 현재 프레임(frame)과 시간적으로 인접한 프레임간의 움직임 예측(motion estimation)과 보상(motion compensation)을 수행하여 영상간에 존재하는 시간적 중복성을 제거하는 것이다. SCC의 표준화과정을 위한 다양한 실험을 통하여 이러한 두 가지 예측 방법은 일반적인 영상(자연 영상) 내에 존재하는 중복성을 제거하기에는 매우 효율적이지만, 스크린 콘텐츠(screen content)에서 발생하는 또 다른 형태의 중복성에 대해서는 효율성이 떨어지는 것이 많은 연구팀에 의하여 관찰되었다. 이는 카메라를 사용하여 얻어지는 일반적인 자연 영상과 스크린 콘텐츠가 상호 차별되는 특성이 있기 때문인데, 그 두드러진 차이를 살펴보면 다음과 같다. 즉, 스크린 콘텐츠의 경우, 1) 강한 에지 성분이 많이 발생한다는 것과; 2) 동일 프레임 내에 반복적으로 발생하는 패턴이 많다는 것으로 요약할 수 있다. 따라서 현재 진행 중인 HEVC SCC 표준화를 위한 기술 제안들은 주로 이러한 두 가지의 새로운 특성을 어떻게 잘 활용하는가에 초점이 맞추어져 이루어지고 있다.

본 고에서는 HEVC SCC 표준 개발을 위한 테스트 모델인 SCM (Screen Content Coding Test Model) 2.0 소프트웨어[5]에 채택된 기술 중 기존 HEVC REXt와 차별화되는 3가지 기술을 설명한다. 우선, 화면 내 예측 부호화 기술의 일종인 IBC(Intra Block Copy) 모드와 팔레트(palette) 모드에 대해 설명하고, 마지막으로 예측을 통해 발생한 잔차 신호에 적용되는 적응적 컬러 공간 변환(adaptive color space transform) 기술에 대해 소개한다. 한편, 위에서 열거한 세가지 기술 중 앞의 두 기술(IBC 모드와 palette 모드)은 SCC 영상의 독특한 특성을 주로 반영하기 위한 기술인 반면, 적응적 컬러 공간 변환 기술은 SCC의 특성과는 관련성이 적으며, SCC 영상 혹은 자연 영상의 구분 없이 성능을 갖는 기술이라 할 수 있다.

Ⅱ. IBC(Intra Block Copy) 모드

서론에서 언급한 것과 같이 스크린 콘텐츠(screen content)에서는 일반적인 영상과는 달리 한 화면 내에서 유사한 패턴이 빈번하게 발생하는 특징이 있다. 이



〈그림 2〉 화면 간 예측과 IBC와의 개념적 차이

것을 공간적 중복성(spatial redundancy)이라고 표현할 수도 있지만, 기존 HEVC 버전 1까지의 화면 내 예측이 주 대상으로 하는 공간적 중복성이 갖는 의미와는 상당한 차이를 가진다. 그 차이점 중 하나는 그 대상과의 거리(distance) 이다. 즉, 화면 내 예측에서 말하는 공간적 중복성은 한 영역의 화소 값들이 공간적으로 바로 '인접'한 영역의 화소 값들과 유사성(혹은 연속성)을 갖기 때문에 발생한다. 반면, IBC 기술에서 말하는 공간적 중복성은 한 영역의 화소들이 '다른' 영역의 화소들과 그 형태 혹은 구성에 있어서 유사성을 가지는 것을 의미한다. 따라서 기존의 화면 내 예측의 경우에는 거리의 개념을 포함하고 있지 않는 반면, IBC 기술은 필연적으로 벡터(거리)의 개념을 내포하고 있다. 이런 의미에서 기존의 움직임예측(좀 더 구체적으로는 HEVC 버전 1의 화면 간 예측 부호화 기술인 AMVP (Advanced Motion Vector Prediction)[6])와 매우 유사한 특징을 가진다. 즉, 블록 단위의 탐색을 통해 예측 값을 찾은 후, 원본 값과 예측 값의 차분 값인 잔차 신호를 부호화한다는 점과 예측 값까지의 위치를 부호화한다는 점에서 유사점을 찾을 수 있다. 이때 IBC 기술과 화면 간 예측과의 가장 큰 차이점은 예측 값의 탐색이 수행되는 참조 영역의 시간적 위치가 다르다는 점이다. 즉, 〈그림 2〉에서 볼 수 있듯이 종래의 화면 간 예측은 현재 프레임(t frame)을 제외한 다른 프레임(예를 들어, 이를 (t-1) frame으로 표시하였다)을 참조 영역으로 설정하는데 반해(〈그림 2〉의 MV, motion vector), IBC 기술은 현재 프레임 내의 복호화가 완료된 영역만을 참조 영역으로 설정한다(〈그림 2〉의 BV, block vector).

Intra Block Copy 기술은 한 화면 내의 유사한 패턴을 참조하여 화면내 부호화 하는 기술이며,이를 위해 벡터(거리)의 개념을 내포하고 있다.

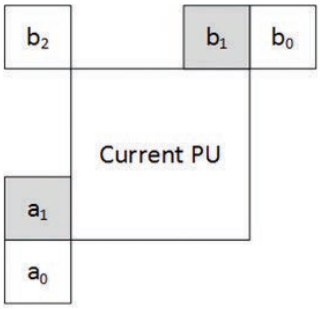
1. Block Vector prediction

IBC 기술에서 현재 블록을 기준으로 예측 블록까지의 위치를 지시하는 2차원 벡터를 블록 벡터(block vector, BV)라 한다. 화면 간 예측의 움직임 벡터(motion vector, MV)와 유사한 개념이지만 IBC와 화면 간 예측을 구분하기 위하여 별도의 명칭인 "블록 벡터"라는 용어를 사용한다. 움직임 벡터와 블록 벡터는 정확도(pel precision)에 있어서 그 차이를 갖는데, 움직임 벡터는 1/4 화소 단위의 정확도(quarter-pel accuracy)를 갖지만, 블록 벡터는 1 화소 단위의 정확도(full pel)로 표현된다. 블록 벡터는 움직임 벡터와 마찬가지로 예측 부호화를 수행한다. 즉, 탐색을 통해 결정한 블록 벡터를 블록 벡터 예측 값(block vector predictor, BVP)과 블록 벡터 차분 값(block vector difference, BVD)으로 나누어 부호화한다. 이를 수식으로 표현하면 다음과 같다.

$$\overrightarrow{BV} = \overrightarrow{BVP} + \overrightarrow{BVD} \quad (1)$$

SCM 2.0을 기준으로 설명하자면 BVP는 2개의 후보벡터(entry)를 가지는 BVP 후보 리스트에서 선택되며, BVP 후보 리스트는 1) 현재 블록에 인접한 바로 이전 블록의 블록 벡터와, 2) 현재 블록과 인접하지는 않으나 과거에 IBC로 부호화가 완료된 블록의 블록 벡터를 사용하여 생성된다. 구체적인 BVP 후보 리스트 생성 과정은 다음과 같다.

Step 1: 현재 PU(Prediction Unit)의 좌측 인접 블록인 〈그림 3〉의 a1 위치에 IBC 모드가 사용된 경우, a1 위치의 BV를 BVP 후보 리스트에 저장한다. 참고로 〈그림 3〉은 AMVP의 spatial MVP(motion vector predictor) 후보와 IBC의 BVP 후보 리스트 생성 과정에서 사용되는 참조 위치를 함께 표현하고 있다. 앞서



〈그림 3〉 Spatial MVP 후보와 BVP 후보

언급하였듯이 IBC의 BVP 후보 리스트 생성과정은 AMVP와 유사한 특징을 지니지만, 상호간의 차이점은 AMVP의 spatial MVP 후보 생성 과정에서는 〈그림 3〉의 {a0, a1, b0, b1, b2} 위치 블록의 정보를 사용하지만, BVP 후보 리스트 생성 과정에서는 {a1, b1} 위치의 블록만을 사용한다는 것이다.

Step 2: 상단 위치의 참조 여부는 부/복호화기의 메모리 사용을 고려하여 동일 CTU(Coding Tree Unit) 내부로 한정된다. 즉, b1 위치가 현재 PU가 속한 CTU 범위를 벗어나는 경우에는 b1 위치는 참조하지 않는다. 그 외의 경우, b1 위치에서 IBC 모드가 사용되었다면 BVP 후보 리스트에 해당 BV를 추가한다. 지금까지의 과정을 통해 2개의 서로 다른 BVP 후보가 생성된 경우 BVP 후보 리스트 생성과정을 종료하고, 그렇지 않을 경우에는 다음 step으로 넘어간다.

Step 3: 과거에 IBC 모드가 사용된 PU 중, 부호화 순서를 기준으로 현재 PU와 가장 가까운 과거 2개의 PU가 가지는 BV를 BVP 후보 리스트에 추가한다. 이 때 고려하는 '과거의 PU'도 현재 PU가 속한 CTU 내부의 PU로 한정된다. 이 과정에서 BVP 후보 리스트에 이미 존재하는 BVP와 과거의 BV가 상호 중복된다면 해당 BV는 BVP 후보 리스트에 추가되지 않는다.

Step 4: 상기 과정을 통해서도 서로 다른 2개의 BVP 후보가 다 생성되지 않은 경우 (즉, 2개의 BVP 후보 리스트가 채워지지 않았을 경우), BVP 후보는 현재 CU의 넓이를 참조하여 디폴트 값인 $(-2\times CU$ 넓이, 0)와 $(-CU$ 넓이, 0)으로 결정된다. 예를 들어, 앞선 과정을 통해서도 BVP 후보가 결정되지 못한 8×8 크기의 CU의 경우, CU넓이는 8 이므로 최종 BVP 후보는 각각 (−16, 0)과 (−8, 0)으로 결정한 후 이를 BVP 후보 리스트에 추가하고 과정을 종료한다. 만일, 디폴트 벡터중 하나만 필요한 경우에는 $(-2\times CU$ 넓이, 0)가 우선적으로 사용된다.

Palette 부호화는 블록 내에 빈번히 발생하는 화소 값을 특정 인덱스로 표현한 후, 비트 스트림에는 인덱스 정보를 부호화 하는 것이다.

2. Intra Block Copy 탐색 영역

IBC 모드는 HEVC REXt의 표준화 과정에서 이미 논의된바 있는 기술이다 (그러나, HEVC REXt 최종 버전 (즉, 버전 2기술)에는 채택되지 않았다). 당시의 IBC 기술과 현재 SCC에 제안되고 있는 IBC 기술 간의 가장 큰 차이점은 탐색 영역의 크기에 있다. HEVC REXt 표준화 과정에서 사용되던 IBC 탐색 영역은 현재 CU가 속한 CTU와 인접한 좌측 CTU로 한정되었다. 이와 달리, 현재 HEVC SCC 표준화 과정에서 논의되는 IBC 탐색 영역은 현재 프레임 내부의 영역 중 현재 CU 이전에 부호화가 완료된 모든 영역으로 확장하였다. 탐색 영역의 확장은 부호화 효율의 향상으로 이어지는데, 실제로 증가된 부호화 효율을 살펴보면, all intra 환경을 기준으로 최대 69.39%의 BD-rate[7] 성능 향상을 보이며, 전체 실험 영상 평균적으로는 19.06%의 BD-rate 성능 향상이 보고 되었다[8]. 또한 자연 영상에서는 IBC의 탐색 영역의 크기를 증가시켜도 성능 향상을 기대할 수 없다는 실험결과 보고[8] 또한 살펴보면, IBC가 스크린 콘텐츠에 대하여 주로 좋은 성능을 보이는 기술이라는 것을 알 수 있다. 이는 자연 영상에 비해 스크린 콘텐츠에서는 유사한 형태나 패턴이 더 빈번하게 발생한다는 사실을 방증하는 것이라 할 수 있다.

Ⅲ. Palette 모드

Palette라는 용어는 이전에도 일반적으로 비트맵을

사용하는 다양한 영상 포맷(BMP, GIF, TIFF, PNG 등)에서 사용 되어 오던 것이다. 예를 들어 16 color 혹은 256 color bitmap의 경우에는 16 혹은 256 개의 컬러 정보를 인덱스(index)로 만든 palette를 가지고 있다고 표현하고 있다. 최근의 HEVC SCC 표준화를 위한 SCM 2.0에 palette 부호화라는 새로운 부호화 방법이 채택되었는데, 그 구체적인 부호화 방식에는 비록 차이가 있지만, palette이라는 기본 개념에 있어서는 기존의 bitmap에서 의미 하는 바와 상당히 유사하다.

palette 부호화는 블록 내에 빈번히 발생하는 화소 값을 특정 인덱스로 표현한 후, 비트 스트림에는 인덱스 정보를 부호화 하는 것이다. 이를 위한 화소 값과 인덱스 간의 맵핑 정보를 palette table으로 정의하고, 각 화소 값을 가리키는 인덱스를 color index라 한다.

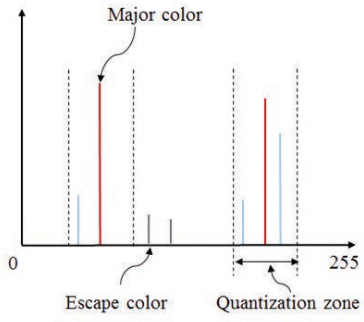
palette 부호화 기술과 기존 압축 부호화 방식을 비교해보면 확연한 차이를 알 수 있다. 지금까지의 전통적인 영상 압축 기술은 예측(prediction), 양자화(quantization), 엔트로피 부호화(entropy coding)의 3가지 주요 과정을 거치고 있다. 전 장에서 설명한 IBC 기술이나 기존의 화면 내 예측(intra prediction) 기술 역시 위의 3가지 주요 과정을 포함하고 있다. 하지만 예측으로 이득을 볼 수 없는 영역(혹은 블록)을 부호화하는 경우에는 부득이하게 양자화와 엔트로피 부호화 과정만으로 압축을 수행해야 하는데, 이것이 palette 부호화가 가지는 가장 큰 특징으로 볼 수 있다. 결과론적으로 IBC나 화면 내 예측 기술이 활용하는 공간적 중복성을 찾기 어려운 경우에 palette 부호화가 선택된다고 볼 수 있다. palette 부호화는 현재 블록을 구성하는 컬러 정보(화소 값의 가짓수)가 상대적으로 적은 경우에도 효과적이다. 여기서 컬러 정보가 적다는 것은 각 컬러를 인덱스로 표현했을 때(즉, palette으로 생성) 더 적은 수의 비트를 할당할 수 있다는 의미로 생각할 수 있다. 따라서 스크린 콘텐츠 중 그래프나 도형 등이 많은, 예를 들어, ppt 스크린을 생각해보면, 사용되는 컬러의 수는 제한적이며(적으며), 그래프나 도형 내부에서는 유사한 색이 빈번히 발생하기 때문에 이런 경우에는 특히 palette 부호화의 성능을 기대할 수 있다.

palette 부호화 과정은 크게 palette table을 부호화하는 과정과 color index map(컬러 대신 인덱스로 맵핑된 영상 또는 블록)을 부호화하는 2개의 과정으로 나눌 수 있다.

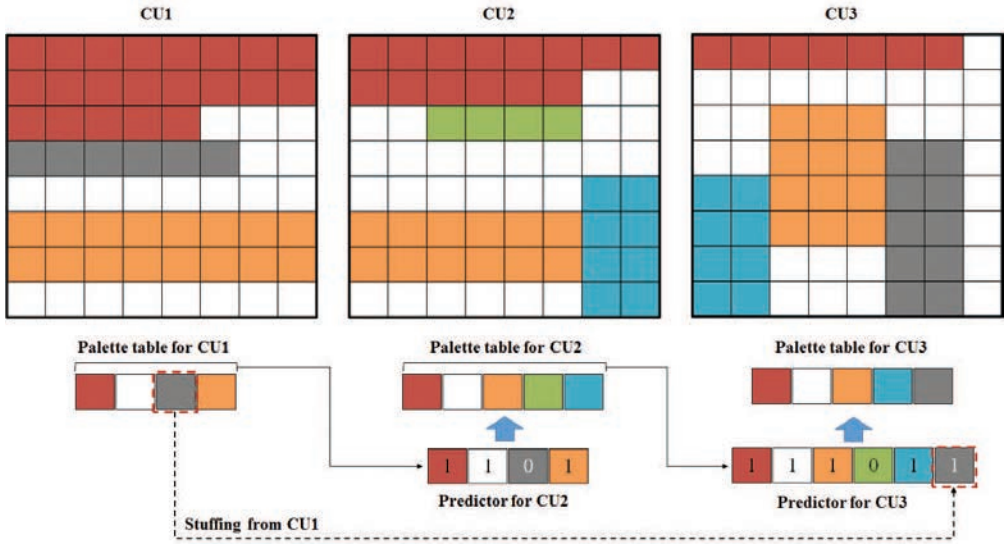
1. Palette table coding

〈그림 4〉는 palette table의 생성 과정을 부호화 입장에서 개념적으로 표현한 그림이다. 우선 화소 값들을 히스토그램으로 나타내었을 때 N 개의 피크가 있다고 가정하고, N 개의 피크 화소 값을 major color라고 정의하고, major color를 중심으로 설정한 양자화 구간(quantization zone) 내부에 위치한 컬러들은 major color로 양자화 한다. 최종적으로 N 개의 major color는 각각의 color index로 맵핑된다. 이때 major color로 표현되지 않는 양자화 구간 외부의 화소 값은 escape color이라 하며, escape color는 color index로 부호화 하지 않고 화소 값 자체를 양자화 하여 부호화 한다. 이해를 돕기 위해 〈그림 4〉에서는 한 가지 컬러 채널(예, R)의 히스토그램을 표현하였다. 하지만 실제 color index 값 하나는 3 채널(예, R, G, B) 화소 값들의 조합을 대표한다. 즉, 하나의 color index가 주어졌을 때 3 채널로 이루어진 완전한 화소 값을 생성할 수 있다.

지금까지 palette table의 color index 생성과정을 개념적으로 살펴보았다. 이렇게 생성된 palette table은 CU별로 독립적으로 부호화해야 하는데, 만약 각 CU의 palette table 간에 공통요소가 있다면 전송 정보를 상당



〈그림 4〉 히스토그램 기반 컬러 분류



〈그림 5〉 palette table의 predictor 생성 과정

히 줄일 수 있을 것이다. 따라서 이를 활용하기 위하여 각 CU별 palette table 간의 예측부호화를 수행한다. 이와 관련된 기술들을 자세히 살펴보면 다음과 같다.

palette table은 앞서 언급한 바와 같이 화소 값 히스토그램을 기반으로 생성된다. 이 과정에서 과거에 palette 모드로 부호화된 CU를 참조하여 현재 CU 의 palette table의 predictor[9](이전 CU의 palette을 사용함)를 생성하며, predictor 내의 각 color index (palette entry) 별로 현재 CU에 실제로 사용되었는지의 여부에 대한 정보(flag)를 전송한다. 이 flag는 previous_palette_entry_flag 라고 부른다. 〈그림 5〉는 palette table 생성 과정에서 과거의 CU를 참조하는 과정의 예를 보여주고 있다. SCM 2.0 발표 이전의 palette 모드 구현을 기준으로 설명하면, CU2를 현재 부호화 중인 CU 라 할 때, palette 모드가 사용된 CU 중, 부/복호화 순서를 기준으로 현재 CU와 가장 인접한 CU(〈그림 5〉의 CU1)의 palette table을 현재 CU의 predictor로 사용한다.

palette stuffing 기술은 시간적으로 가장 가까운 CU뿐만 아니라, 바로 그 이전 CU의 palette table로부터 현재palette table의 predictor를 생성하는 기술이다.

(1) Palette stuffing

〈그림 5〉에서 만약 CU1에서 사용된 총 color index 개수가 적다면 CU2에 대한 predictor가 가지는 color index의 개수 역시 적을 것이다. 이것은 predictor로 사용될 후보 color index의 개수가 적기 때문에, color index를 재사용한다는 입장에서 보면 그 활용 빈도가 떨어질 가능성이 높다. 이를 보완하기 위하여 SCM 2.0에서는 palette stuffing이라는 기술을 채택하였다. 본 기술은 시간적으로 가장 가까운 CU뿐만 아니라, 바로 그 이전 CU의 palette table로부터 현재 palette table의 predictor를 생성하는 기술이다. 〈그림 5〉를 예로 들면, CU3에 대한 palette table의 predictor가 가지는 entry 중 최우측 요소(검정색)는 CU2가 아닌 CU1으로부터 생성된 것을 알 수 있다. 참고로 SCM 2.0 발표 이전의 palette 모드는 1개의 CU를 참조하여 predictor를 생성하였으나, palette stuffing 기술을 채택한 SCM 2.0의 palette 모드는 2개의 CU를 참조하여 predictor를 생성한다는 차이점이 있다.

(2) Grouping reuse flag

상기 palette stuffing 기술은 palette table의 predictor가 갖는 palette entry 개수가 적은 경우를 해결하는 방법이다. 하지만 반대로 predictor 내의 palette entry의 개수가 과도하게 많은 경우, previous_palette_entry_flag의 개수 또한 증가한다(비트 량 증가). 이러한 문제점을 해결하기 위하여 위의 flag 4개를 하나의 그룹으로 구분하고, 그룹 단위로 그 사용 여부를 알리는 정보를 보내는 방식을 사용하고 있다. 이 그룹 단위의 flag는 palette_all_zeros_in_group flag(이하 grouping reuse flag)라 한다. grouping reuse flag는 현 그룹에 포함된 palette entry들이 모두 사용되지 않은 경우에 '1'로 설정되며, 이때 그룹 내의 previous_palette_entry_flag들은 전송되지 않는다. 추가적으로 영상 부호화에서 널리 활용되는 EOB(end of block, 블록 내에서 부호화되는 마지막 샘플임을 알리는 flag)와 유사한 개념의 flag도 사용되고 있다. 이는 palette_last_group flag이라 하며, 해당 그룹이 현재 CU의 palette table에서 사용된 마지막 palette entry를 포함하는지 여부를 알리는데 사용된다. 따라서 grouping reuse flag 값이 '0'인 경우(즉, 현재 그룹 내의 palette entry 중 현재 CU에 사용된 palette entry가 적어도 하나 이상 존재하는 경우), palette_last_group flag를 통해 추가적인 그룹이 있는지의 여부를 알리며, palette_last_group flag가 '1'인 경우 현재 palette table에 대한 predictor의 시그널링을 종료한다.

(3) Palette sharing

palette sharing은 과거의 palette 모드로 부호화된 CU의 palette table을 현재 CU에 그대로 사용하는 방법이다. 이때 사용되는 과거 CU는 부/복호화 순서를 기준으로 시간적으로 바로 이전의 palette 모드로 부호화된 CU로 한정한다.

Adaptive color space transform 기술은 컬러 공간 변환을 통해 부호화 효율을 높이는 것이다.

2. Color index map coding

앞서 설명한 palette table이 생성된 이후, 현재 CU 내의 화소들은 color index로 바뀌어 표현되며 이를 color index map이라 한다. color index map의 부호화에 앞서, CU 단위의 palette_transpose_flag를 사용하여 color index의 스캔 방향에 대한 정보를 전송한다. 따라서 color index map의 스캔 방향은 각 CU별로 다르게 선택할 수 있다. 상기 과정을 통해 최종적으로 구해진 color index map은 'INDEX 모드', 'COPY_ABOVE 모드', 그리고 'ESCAPE 모드'의 3가지 모드를 사용하여 부호화된다. 각 모드의 의미와 전송되는 정보는 다음과 같다.

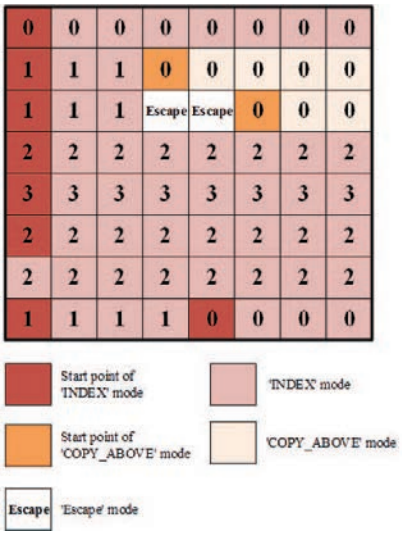
INDEX 모드: 런(run)의 시작 color index 값과 런 길이(run-length)를 부호화 한다. 따라서 모드 정보, color index, 런 길이의 3가지 정보가 전송된다.

COPY_ABOVE 모드: 현재 위치를 기준으로 상단 행의 color index들을 런 길이만큼 현재 위치로부터 시작하여 채워 넣는다. 따라서 모드 정보, 런 길이의 2가지 정보가 전송된다.

ESCAPE 모드: escape color로 결정된 화소는 상기 INDEX 모드와 COPY_ABOVE 모드를 사용하지 않고, 화소 값을 직접 양자화 하여 전송한다. 따라서 escape color 화소가 연속적으로 존재하더라도 런 길이를 사용하지 않고, 각 escape 화소별로 모드 정보와 양자화된 화소 값의 2가지 정보가 전송된다.

이렇게 3가지 모드를 이용한 color index map의 부호화 과정은 〈그림 6〉의 간단한 예에서 확인할 수 있다. palette 모드로 인해 향상되는 부호화 효율을 살펴보면, all intra 환경을 기준으로 최대 25.98%의 BD-rate 성능 향상을 보이며, 전체 실험 영상 평균적으로는 10.60%의 BD-rate 성능 향상이 보고되었다.

한편, 자연 영상의 부호화 성능만을 보면 0.05%의 미세한 BD-rate 성능 저하를 보인다[10]. 요약하면, palette 모드 역시 IBC와 마찬가지로 스크린 콘텐츠에 대하여 매우 효율적인 기술이지만, 자연 영상에 대해서는 그 효과가 미미함을 알 수 있다.



〈그림 6〉 color index map 부호화

Ⅵ. Adaptive color space transform

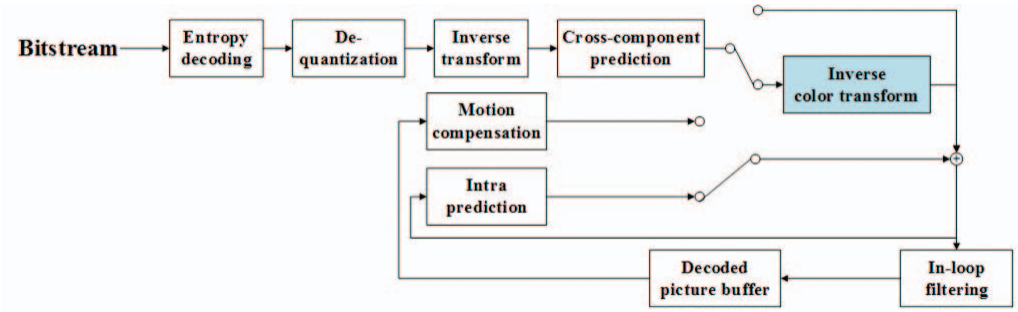
앞의 두 기술, 즉 IBC와 palette 부호화는 스크린 콘텐츠를 위해 설계된 기술인데 반해, adaptive color space transform 기술은 컬러 공간 변환을 통해 부호화 효율을 높이는 것이라고 할 수 있다. 따라서 스크린 콘텐츠와는 큰 관련성이 없으며, 스크린 콘텐츠와 자연 영상 모두의 경우에 있어 압축 효율을 제공하는 기술이다.

HEVC 버전 1은 전통적인 영상 압축 방법과 마찬가지로 YCbCr 영상의 부호화를 고려하여 설계되었다. 하지만 HEVC range extensions 및 SCC에서는 컬러 변환에서 발생하는 정보 손실을 막기 위하여 RGB의 직접 부호화를 지원하고 있다. 따라서 본 기술은 RGB 직접 부호화의 압축 효율을 돕는 기술로 이해할 수 있다. 컬러 변환에서 정보 손실이 발생하는 이유는 현재 사용되는 영상 시스템과 컬러 변환이 설계된 이유에서 찾을 수 있다.

전통적인 시스템들의 압축과정은 영상 획득과 디스플레이에서 사용하는 컬러 공간과는 다른 공간을 사용하여 왔다. 구체적으로 영상의 획득과 디스플레이 시에는 RGB 컬러 공간의 데이터가 사용되지만 영상의 압축 시에는 YCbCr 컬러 공간이 사용되어 왔다. YCbCr 컬러 공간은 1982년에 제정된 컬러 변환 표준으로, ITU-R BT.601(최초에는 CCIR.601)은 SD급, BT.709는 HD급을 위하여 제정된 표준이다. 또한 YCbCr은 이전 컬러시스템인 YUV의 특성을 이어받은 컬러 공간이지만, YUV는 텔레비전을 위한 아날로그 신호이고 YCbCr은 디지털 신호를 다룬다는 차이가 있다.

이처럼 영상 압축에서 YCbCr 컬러 공간이 선호되어 온 이유는 크게 두 가지로 설명할 수 있다. 첫째로, RGB로 표현된 신호는 각 채널 간의 상관도가 높기 때문에 상당한 중복성이 존재한다. 따라서 저장이나 전송 혹은 압축의 관점에서 상당히 비효율적이다. 두 번째 이유는 인지시각시스템(HVS: Human Visual System) 관점에서 찾을 수 있다. 사람의 눈은 밝기 성분(Y)보다 색차 성분(Cb, Cr)에 덜 민감하게 반응한다. 따라서 색차 성분은 서브 샘플링하거나 압축하기에 용이하다.

YCbCr 컬러 변환은 영상 압축 관점에서 유용한 측면을 가지지만, RGB에서 YCbCr로(변환) 또한 YCbCr에서 RGB로(역변환) 변환 과정에서 데이터 손실이 발생한다는 단점 또한 존재한다. 예를 들어 YCbCr 4:2:0의 경우에는 RGB로 역변환 할 경우(RGB → YCbCr → RGB) 26~40dB 수준으로 떨어지고, 서브 샘플링이 없는 4:4:4의 경우에도 대략 53dB 수준으로 화질 열화가 발생할 수 있다. 따라서 컬러 정보의 손실을 막는 가장 좋은 방법은 RGB 컬러 정보의 직접 부호화라고 할 수 있지만, 앞에서 언급했던 컬러 채널 간의 중복성은 여전히 존재하게 된다. adaptive color space transform은 이러한 중복성을 줄이는 역할을 한다. 본 기술은 RGB 컬러 공간에서 예측(prediction) 과정이 끝난 이후에 적용되는데, 즉 원본 값과 예측 값의 차분 값인 RGB 컬러 공간의 잔차 신호에 적용되며, 해당 잔차 신호의 컬러 공간을 선택적으로 변환한다. 〈그림 7〉은 adaptive color space transform 기술이 적용된 복호화기 구조를 나타낸다. 참고로 〈그림 7〉의 cross-component prediction 기술은 HEVC REXt 표준에 채택된 기술로, 각 컬러 채널간의 중복성을 예측 과정 수행을 통하여 감소시킨다. 따라서 채널 간 중복성을 줄이기 위한 기술이라는 측면에서는 adaptive color



〈그림 7〉 Adaptive color space transform을 포함한 복호화 순서도

space transform과 유사점을 찾을 수 있다. 하지만 adaptive color space transform의 경우에는 prediction 수행을 통해서가 아니라, decorrelation을 수행하는 변환이라는 점에서 차이점이 있다.

1. YCoCg & YCoCg-R color space

adaptive color space transform에서 사용하는 컬러 공간은 YCoCg이다. 이는 손실 부호화와 무손실 부호화에 따라 각각 YCoCg와 YCoCg-R 컬러 공간이라는 다른 명칭을 갖는다.

우선 손실 부호화에 사용되는 컬러 공간 변환인 YCoCg[11] 변환을 살펴보면, YCoCg 변환은 변환(forward transform) 과정에서는 식 (2)가, 역변환(inverse transform) 과정에서는 식 (3)이 사용된다.

Adaptive color space transform 기술은 RGB 실험 영상 평균적으로18.01%의 BD-rate 성능 향상이 보고되었다.

$$\begin{bmatrix} Y \\ Co \\ Cg \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 0 & -2 \\ -1 & 2 & -1 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} /4 \qquad (2)$$

FORWARD TRANSFORM

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 1 & -1 \\ 1 & 0 & 1 \\ 1 & -1 & -1 \end{bmatrix} \begin{bmatrix} Y \\ Co \\ Cg \end{bmatrix} \qquad (3)$$

INVERSE TRANSFORM

식 (2)와 (3)의 YCoCg 변환/역변환은 수많은 영상에 대하여 3×3 integer correlation matrix를 계산하고, 이에 대하여 KLT(Karhunen-Loeve Transform)을 수행하여 최대의 decorrelation을 제공하는 변환을 찾아내는 과정을 통해 설계되었다. 여기에 역변환의 계산을 수월하게 설계하기 위해 KLT의 값을 약간 조정하여 식 (2)와 식 (3)의 최종 변환을 얻게 되었다[11]. 따라서 식 (3)의 경우에는 덧셈 연산만으로 역변환이 가능하다.

YCoCg 변환은 정규화된 변환(normalized transform)이 아니라는 특징을 갖는다. 즉, Y와 Cg 신호의 norm은 $\sqrt{6}/4$이고 Co 신호의 norm은 $\sqrt{8}/4$이다. 따라서 손실 부호화에 adaptive color space transform이 사용된 경우에는 이를 보상하기 위해 QP(quantization parameter) 값을 조정하는 과정이 요구된다. SCM 2.0에서는 Y, Co, Cg 채널별로 (QP-5, QP-3, QP-5) 값이 적용된다. 컬러 공간 변환 이후, 양자화 과정에서 QP 값을 조정하는 과정을 제외하고는 HEVC REXt의 잔차 신호 부호화 과정과 동일한 과정을 수행한다. 참고로 디블록킹 필터링(deblocking filtering) 시에는 컬러 공간 변환 여부와 무관하게 해당 CU에 할당된 기본 QP 값을 기반으로 디블록킹 필터링을 수행한다.

YCoCg와 RGB 신호간의 변환 과정은 수학적으로 손실을 발생시키지 않으나 식 (2)의 나눗셈 연산을 수행하는 과정에서 소수 단의의 정확도가 요구된다. 따라서 데이터의 손실이 허용되는 손실 부호화 모드와는 달리, 무손실 부호화 모드의 경우 무손실을 보장하기 위해서는 소수점 이하의 숫자를 표현해야 하며, 이는 추가적인 비트 량 증가로 이어진다. 이러한 점 때문에 무손실

부호화 시의 adaptive color space transform에서는 정수 연산만으로도 완벽한 복원이 가능한 YCoCg-R[11] 변환을 사용한다. 변환 및 역변환은 각각 식 (4)와 식 (5)가 사용되는데, 이는 YCoCg 변환을 lifting scheme을 사용하여 수정한 변환이다.

$$\begin{aligned} Co &= R - B \\ t &= B + (Co \gg 1) \\ Cg &= G - t \\ Y &= t + (Cg \gg 1) \end{aligned} \quad (4)$$

FORWARD TRANSFORM

$$\begin{aligned} t &= Y - (Cg \gg 1) \\ G &= Cg + t \\ B &= t - (Co \gg 1) \\ R &= B + Co \end{aligned} \quad (5)$$

INVERSE TRANSFORM

adaptive color space transform 기술은 앞서 설명한 IBC와 palette 모드와는 달리 '스크린 콘텐츠'의 효율적인 부호화를 목적으로 설계된 기술이 아니다. 따라서 RGB 영상인 경우 자연 영상과 스크린 콘텐츠에 공히 고른 성능 향상을 보인다. all intra 환경을 기준으로 RGB 영상에 대해 최대 28.50%의 BD-rate 성능 향상을 보이며, RGB 실험 영상 평균적으로는 18.01%의 BD-rate 성능 향상이 보고되었다[12].

V. 결 론

본 고에서는 HEVC screen content coding(SCC) 표준 개발을 위한 SCM 2.0에 채택된 대표적인 기술을 살펴보았다. 스크린 콘텐츠(screen content)는 기존의 카메라로 촬영된 영상과는 확연하게 다른 특성을 지닌다는 점에서 영상 압축 분야에 새로운 화두로 떠오르고 있으며, 또한 종래의 영상 압축 표준과는 조금 다른 접근이 이루어지고 있다. 한편, HEVC SCC의 표준은 SCM 2.0 기준으로 아직 두 차례의 표준화 회의 밖에 진행되지 않은 상태로 여전히 시작단계라고 할 수 있다. 따라서 표준화 회의가 진행됨에 따라 본 고에서 설명한 기술들은 추가적인 개선 및 변경이 예상되며, 또한 그 외의 기술도 다수 추가될 가능성 또한 열려있는 상태이다. 따라서 산업계 및 학계는 HEVC SCC의 표준화 과정을 주목해 볼 필요가 있다.

감사의 글

본 고는 2011년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. 2011-001-7578)

참 고 문 헌

[1] High Efficiency Video Coding, Rec. ITU-T H.265 and ISO/IEC 23008-2, Jan. 2013.

[2] G. J. Sullivan, 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.

[3] R. Joshi, et al., "HEVC Screen Content Coding Draft Text 1" document JCT-VC-R1005, Sapporo, JP, 30 June - 9 July 2014.

[4] J. Boyce, et al., "Edition 2 Draft Text of High Efficiency Video Coding (HEVC), Including Format Range (RExt), Scalability (SHVC), and Multi-View (MV-HEVC) Extensions" document JCT-VC-R1013, Sapporo, JP, 30 June - 9 July 2014.

[5] SCM 2.0 software, available at: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-15.0+RExt-8.0+SCM-2.0

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

[7] G. Bjontegaard, "Calculation of Average PSNR Differences Between RD-curves," VCEG-M33, ITU-T Video Coding Experts Group (VCEG) Meeting, Austin, TX, April 2-4, 2001.

[8] C. Pang, et al., "SCCE1: Test 1.1 - Intra block copy with different search areas," document JCT-VC-R0184, Sapporo, JP, 30 June - 9 July 2014.

[9] C. Gisquet, et al., "SCCE3 Test A.3: palette stuffing," document JCT-VC-R0082, Sapporo, JP, 30 June – 9 July 2014.

[10] Patrice Onno, et al., "Suggested combined software and text for run-based palette mode," document JCT-VC-R0348, Sapporo, JP, 30 June – 9 July 2014.

[11] Henrique S. Malvar, et al., "Lifting-based reversible color transformations for image compression," SPIE Proceedings, Vol. 7073, DOI: 10.1117/12.797091, Oct. 2008.

[12] Li Zhang, et al., "SCCE5 Test 3.2.1: In-loop color-space transform," document JCT-VC-R0147, Sapporo, JP, 30 June – 9 July 2014.

양 승 하

• 2013년 성균관대학교 정보통신대학 졸업 (학사)
• 2013년~현재 성균관대학교 정보통신대학 석사과정

〈관심분야〉
멀티미디어, 영상압축, 신호처리

심 혁 재

• 2000년 성균관대학교 전자공학과 졸업 (학사)
• 2002년 성균관대학교 정보통신공학부 졸업 (석사)
• 2013년 성균관대학교 정보통신공학부 졸업 (공학박사)
• 2013년~현재 성균관대학교 정보통신공학부 박사 후 과정

〈관심분야〉
멀티미디어, 영상압축, 신호처리

전 병 우

• 1985년 서울대학교 전자공학과 졸업 (학사)
• 1987년 서울대학교 전자공학과 졸업 (석사)
• 1992년 Purdue Univ, School of Elec. 졸업 (공학박사)
• 1993년~1997년 삼성전자 신호처리연구소 선임/수석연구원
• 1997년~현재 성균관대학교 전자전기공학부 교수

〈관심분야〉
멀티미디어 영상압축, 영상인식, 신호처리