적응적인 움직임 벡터 해상도를 이용한 움직임 벡터 부호화 방법

장명훈^{a)}, 서찬원^{a)}, 한종기^{a)†}

Motion Vector Coding Using Adaptive Motion Resolution

Myung-Hun Jang^{a)}, Chan-Won Seo^{a)}, and Jong-Ki Han^{a)†}

요 약

MPEG-2, MPEG-4와 같은 기존의 비디오 코덱에서는 인터 예측을 수행할 때 고정된 해상도의 움직임 벡터를 사용한다. 그러나 KTA 참조 소프트웨어에서는 움직임 벡터의 해상도를 슬라이스 단위로 선택하여 사용할 수 있는 기능을 지원한다. 그러나 선택된 하나의 움직임 벡터 해상도를 슬라이스 전체에 일괄적으로 적용하기 때문에 영상의 국지적인 특성을 반영하는데 어려움이 있다. 본 논문에서는 탐색 구간에 따라 적응적으로 움직임 벡터의 해상도를 결정하는 방법을 제안한다. 움직임 벡터의 탐색 영역을 움직임 벡터가 예측 움직임 벡터로부터 떨어진 거리에 따라 다수개의 구간으로 분할하고, 각 구간에 대하여 하나의 움직임 벡터 해상도를 할당하여 움직임 예측에 적용한다. 따라서 제안하는 방법의 부호화 효율은 각 구간을 분할하는 Threshold와 움직임 벡터를 부호화하는 엔트로피 코딩 방법에 영향을 받는다. HEVC의 참조 소프트웨어인 HM3.0을 이용하여 실험한 결과, Random Access 부호화 구조에서는 평균적으로 약 0.9%의 성능 향상을 얻을 수 있었으며, Low Delay 부호화 구조에 B picture를 적용한 경우는 약 0.6%, P picture를 적용한 경우에는 약 2.7%의 평균 발생 비트량 감소를 확인할 수 있었다.

Abstract

In most conventional video codecs, such as MPEG-2 and MPEG-4, inter coding is performed with the fixed motion vector resolution. When KTA software was developed, resolution for MVs can be selected in each slice. Although KTA codec uses a variety of resolutions for ME, the selected resolution is applied over the entire pixels in the slice and the statistical property of the local area is not considered. In this paper, we propose an adaptive decision scheme for motion vector resolution which depends on region, where MV search area is divided to multiple regions according to the distance from PMV. In each region, the assigned resolution is used to estimate MV. Each region supports different resolution for ME from other regions. The efficiency of the proposed scheme is affected from threshold values to divide the search area and the entropy coding method to encode the estimated MV. Simulation results with HM3.0 which is the reference software of HEVC show that the proposed scheme provides bit rate gains of 0.9%, 0.6%, and 2.9% in Random Access, Low Delay with B picture, and Low Delay with P picture structures, respectively.

Keyword: Motion Vector Resolution, Motion Estimation, Motion Vector Coding

a) 세종대학교 정보통신공학과

Sejong University, Dept. of Information and Comm. Engineering

† 교신저자 : 한종기 (hjk@sejong.ac.kr)

※ 본 연구는 지식경제부 및 한국산업기술평가관리원의 산업융합원천기술 개발사업(정보통신)의 일환으로 수행하였음. [K1002142, 차세대 모바일 영상서비스를 위한 초경량 비디오 부호화 원천기술개발]

· 접수일(2011년12월14일), 수정일(2011년12월21일), 게재확정일(2011년12월21일)

1. 서론

비디오 코딩에서는 시간적으로 연속한 영상의 중복성을 최소화하기 위하여 인터 예측을 수행한다. 인터 예측에서는 현재 부호화하고자하는 블록과 유사한 블록을 과거에

이미 복호화 된 참조 영상에서 찾기 위하여 움직임 예측을 수행하고, 현재 블록과 참조 영상에서 찾은 블록의 차이 신호를 변환 및 양자화를 통해 부호화한다. 그리고 두 블록 사이의 공간 좌표 차이를 움직임 벡터로 표현한다. H.261에서는 최적의 움직임 벡터를 정수 화소 단위로 찾아서 움직임 예측을 수행 하며, MPEG-2와 MPEG-4에서는 보다 정확한 움직임 예측을 수행하기 위하여 보간법을 적용한 반 화소 단위 해상도의 움직임 벡터를 사용 한다^[1-3]. 최신 표준 비디오 코덱인 H.264/AVC에서는 윌-왜곡 최적화 관점에서 움직임 예측을 효과적으로 수행하기 위하여 1/4 화소 단위의 움직임 벡터 해상도를 사용 한다^[4-6]. H.264/AVC 표준화 이 후, H.264/AVC의 부호화 효율을 향상시키기 위하여 ITU-T VCEG에서는 Key Technical Area (KTA)를 개발하여 다양한 기술을 시험하였다^[7]. 움직임 벡터 해상도와 관련된 기술로써, 슬라이스 단위로 1/8 화소와 1/4 화소 단위의 움직임 벡터 해상도 중 하나를 선택하여 움직임 예측 및 부호화를 수행하는 기술이 연구되었다^[8]. 최근 표준화가 진행 중인 High Efficiency Video Coding (HEVC)에서도 H.264/AVC와 같이 움직임 벡터 해상도를 1/4 화소 단위로 고정하여 사용하고 있다^[9].

움직임 벡터 해상도는 비디오 코덱의 부호화 효율에 큰 영향을 미친다. 움직임 벡터를 표현함에 있어서, 정수 단위와 같이 낮은 단위의 해상도가 적용이 되는 경우에는 움직임 벡터의 부호화에 사용되는 비트량은 절감할 수 있다는 장점이 있는 반면, 움직임 예측의 정확도가 낮아져서 예측 오차 신호가 많이 발생하는 단점이 있다. 1/8 화소 단위 움직임 벡터 해상도와 같이 높은 해상도를 움직임 벡터의 부호화에 사용하게 되면, 낮은 움직임 벡터 해상도를 사용하는 경우에 비하여 같은 크기의 움직임 벡터를 부호화할 경우에도 더 많은 비트량을 사용하게 되는 단점이 발생한다. 그러나 움직임 예측의 정확도가 높아져서 예측 오차 신호의 양이 감소하게 되는 장점을 얻을 수 있게 된다. 기존 비디오 코덱들에서는 이 같은 움직임 벡터 해상도가 부호화 효율에 미치는 영향을 효과적으로 반영하지 못하고, 지정된 하나의 움직임 벡터 해상도만을 사용한다. 따라서 움직임 벡터의 해상도를 적응적으로 조절하여 비디오 코덱의 부호화 효율을 향상 시키는 기술이 필요하다.

본 논문에서는 최적의 움직임 벡터 해상도를 결정하기 위하여 움직임 예측 구역을 다수개의 구간으로 분할하고, 각 구간마다 하나의 움직임 벡터 해상도를 할당한다. 이 때 구간은 예측 움직임 벡터로부터 움직임 벡터가 떨어진 거리, 즉 차분 움직임 벡터의 절대값에 따라 결정하며, 각 구간을 분할하기 위하여 다수개의 Threshold를 사용한다. 부호화기에서 움직임 예측 시 사용한 Threshold들은 슬라이스 헤더에 부호화하여 복호기에서도 동일한 구간을 적용하여 움직임 벡터 해상도를 결정할 수 있도록 한다. 그리고 모든 Threshold들은 부호화 효율이 최대가 될 수 있도록 윌-왜곡 최적화 관점을 고려하여 결정한다.

본 논문의 구성은 다음과 같다. 제 II장에서는 기존 코덱들의 움직임 벡터 부호화 방법을 소개한다. 제 III장에서는 제안하는 적응적인 움직임 벡터 해상도를 이용한 차분 움직임 벡터의 부호화 방법에 대하여 설명하며, 최적의 Threshold를 결정하는 방법을 제 IV장에서 제안한다. 제 V장에서는 제안하는 알고리즘을 적용한 실험결과 및 분석을 서술하고, 마지막 제 VI장에서 본 논문의 결론을 맺는다.

II. 움직임 벡터 부호화

인터 예측은 영상의 시간적 중복성을 이용하여 현재 영상의 정보를 효과적으로 예측 부호화 할 수 있도록 하는 중요한 기술 중 하나이다. 인터 예측에서는 참조 영상에서 현재 블록과 가장 유사한 블록을 찾기 위하여 움직임 예측을 수행하며, 그 결과로 발생하는 두 블록간의 차이 신호를 변환 및 양자화를 통해서 부호화한다. 그리고 두 블록간의 상대적인 위치 정보를 움직임 벡터로 표현하고, 이 값을 예측을 통해 부호화하여 복호화기에 전송한다. H.264/AVC 및 KTA를 비롯하여 HEVC의 참조 소프트웨어인 HM3.0에서는 움직임 예측 과정에서는 최적의 움직임 벡터를 찾기 위하여 다음과 같은 비용함수를 이용한다 ^{[5][7][10]}.

$$J_M = D + \lambda R \quad (1)$$

수식 (1)에서 J_M 은 움직임 예측에 사용되는 비용함수를

의미한다. D 는 현재 블록과 예측 블록의 예측 오차이며, R 은 움직임 벡터와 예측 움직임 벡터의 차분 움직임 벡터와 그 밖의 움직임 정보를 부호화하는데 사용되는 비트량을 나타낸다. λ 는 라그랑지안 곱수를 의미한다. 일반적인 비디오 코덱에서는 D 를 계산하기 위하여 다음과 같은 Sum of Absolute Difference (SAD) 왜곡 측정 방법을 사용한다.

$$D = \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} |O(i,j) - M(i,j)| \quad (2)$$

여기서, O 와 M 은 각각 현재 블록과 예측 블록의 화소 값을 나타내며, W 와 H 는 현재 블록의 가로와 세로 길이를 표현한다. 수식 (1)에서 차분움직임 벡터와 움직임 정보의 부호화에 사용되는 비트량 R 은 다음과 같은 수식으로 계산한다.

$$R = b(MVD) + b(etc.) \quad (3)$$

수식 (3)에서 함수 $b(X)$ 는 입력값 X 를 부호화하는데 필요한 비트량을 출력하는 함수이다. MVD 는 차분 움직임 벡터를 의미하며, *etc.*는 그 밖에 부호화되는 참조 영상 인덱스 등의 정보를 나타낸다. 여기서 MVD 는 아래의 수식과 같이 계산된다.

$$MVD = MV - PMV \quad (4)$$

그리고 MVD 의 각 컴포넌트들을 표현하기 위하여 표 1과 같은 Exponential-Golomb code를 사용한다^{[6][7][9]}.

표 1. 고정된 움직임 벡터 해상도에 적용되는 Exponential-Golomb code
Table 1. Exponential-Golomb code for fixed MV resolution

차분 움직임 벡터의 한 컴포넌트	코드 번호	비트 스트링
0	0	0
1	1	010
-1	2	011
2	3	00100
-2	4	00101
3	5	00110
...	...	...

서론에서 이야기 한 것처럼 비디오 코덱들은 움직임 예측 과정에서 더 정확한 예측 블록을 만들기 위해 움직임 벡터의 해상도를 높여 정수화소 뿐 아니라 보간 된 화소에서도 움직임 예측을 수행한다. 이처럼 높은 해상도의 움직임 벡터를 사용함으로써 얻을 수 있는 정확한 예측 블록은 블록간 발생하는 차분 신호의 양을 감소시켜서 부호화 할 데이터 량을 감소시켜주는 역할을 한다. 그러나 차분 움직임 벡터의 부호화에 더 많은 비트량을 사용해야하는 부담이 발생한다. 표 2는 움직임 벡터의 해상도에 따라 차분 움직임 벡터에 할당되는 코드 번호를 보여주며, 표 3은 각 코드 번호를 Exponential-Golomb code를 이용하여 표현한 비트 스트링을 보여준다. 예를 들어, 차분 움직임 벡터의 x 축 값이 정수 화소 단위로 3만큼 발생하는 경우를 생각해 보면, 정수 화소 단위 움직임 벡터를 사용하는 경우에는 코드 번호가 5가 되고 비트 스트링이 '00110'이 되어 5 비트로 표현한다. 그러나 1/8 움직임 벡터 해상도를 적용하게 되면, 코드 번호가 47이 할당되어 3이라는 차분 움직임 벡터를 11 비트로 표현한다. 이처럼 움직임 벡터 해상도에 따라 차분 움직임 벡터의 부호화에 필요한 비트량이 변화하게 된다.

표 2. 움직임 벡터 해상도에 따른 코드 번호 할당
Table 2. Code number assignment according to MV resolutions

차분 움직임 벡터의 한 컴포넌트	코드 번호			
	1/8 화소	1/4 화소	1/2 화소	1/1 화소
0	0	0	0	0
1/8	1			
-1/8	2			
2/8	3	1		
-2/8	4	2		
...	...	...	...	...
8/8	15	7	2	1
...	...	...	...	...
24/8	47	23	6	3
...	...	...	...	...

표 3. 코드 번호에 따른 Exponential-Golomb code
Table 3. Exponential-Golomb code for code number

코드 번호	비트 스트링
0	0
1	010
2	011
3	00100
4	00101
5	00110
6	00111
...	...
23	000011000
...	...
47	00000101111
...	...

III. 적응적인 움직임 벡터 해상도를 이용한 차분 움직임 벡터 부호화

1. 제안배경

움직임 벡터의 해상도는 정확도가 높은 예측 블록을 생성하는데 도움을 주지만, 차분 움직임 벡터를 표현하는데 있어서 더 많은 비트량을 요구하게 된다. 반대로 낮은 움직임 벡터 해상도를 사용하는 경우에는 차분 움직임 벡터에 할당되는 비트량은 작지만, 정확한 예측 블록을 생성하는데 어려움이 있다. 따라서 움직임 벡터의 해상도가 예측 블록의 정확도에 큰 영향을 미치지 않는 블록의 경우에는 낮은 움직임 벡터 해상도를 적용하고, 그 반대의 경우에 대해서는 높은 해상도의 움직임 벡터를 사용하는 것이 움직임 벡터의 해상도를 조절하여 부호화 효율을 높이는데 가장 효과적이다.

예측 움직임 벡터가 실제 발생한 움직임 벡터와 유사한 경우에는 차분 움직임 벡터의 절대값 크기가 작기 때문에 차분 움직임 벡터를 부호화 하는데 사용되는 비트량이 적다. 이런 경우에는 수식 (1)의 두 블록간의 오차신호 D 를

줄이기 위해 높은 해상도를 고려하는 것이 유리하다. 예측 움직임 벡터가 움직임 벡터를 정확하게 예측하지 못하면 차분 움직임 벡터를 부호화 하는데 필요한 비트량이 많아 지기 때문에 부호화 효율이 낮아진다. 이 경우에는 낮은 해상도의 움직임 벡터를 고려하는 것이 부호화 효율 측면에서 유리하다^[11]. 이러한 경향을 확인하기 위하여 HEVC의 참조 소프트웨어인 HM3.0을 이용하여 두 가지 경우에 대하여 실험하였다^[10]. HM3.0은 1/4 화소 단위의 움직임 벡터의 해상도를 사용한다. 그리고 실험에서는 HEVC 표준화 과정에서 사용되는 영상 중 Class D (412×240) 해상도에 해당하는 영상을 사용하였으며, 다른 실험 조건은 HEVC 표준화에 사용되는 공통 실험 조건을 이용하였다^[12]. 첫 번째 실험에서는, 움직임 벡터의 해상도를 1/2 화소로 설정하고 기존 HM3.0의 실험 결과와 비교하였으며, 표 4에서 실험 결과를 보여준다.

표 4. 움직임 벡터의 해상도를 1/2 화소단위로 사용했을 경우 부호화 효율
Table 4. Performance with 1/2-pel MV resolution

해상도	실험 영상	BD-Rate (%)
Class D (416×240)	BasketballPass	-0.1
	BQSquare	9.4
	Blowing Bubbles	2.7
	RaceHorses	0.9
Average		3.2

표 4는 1/4 화소 단위의 움직임 벡터를 사용한 경우가 1/2 화소 단위의 움직임 벡터의 경우에 비하여 평균적으로 약 3.2%의 부호화 효율이 좋음을 보여준다. 특히, 영상의 공간 복잡도가 높은 BQSquare나 Blowing Bubbles 같은 실험 영상에서는 두 경우의 부호화 효율 차이가 큼을 확인할 수 있다. 반면, 큰 움직임이 많이 발생하는 BasketballPass에 대한 실험 결과는 1/2 화소 단위 움직임 벡터 해상도를 적용하는 것이 약 0.1% 정도 부호화 효율이 좋음을 알 수 있었다. 이는 영상의 특성에 따라 특정 움직임 벡터 해상도가 효율성을 보일 수 있음을 의미한다. 또 다른 실험으로, 표

5에서는 HM3.0과 1/8 화소 단위 움직임 벡터를 사용한 경우의 부호화 효율을 비교한다.

표 5. 움직임 벡터의 해상도를 1/8 화소단위로 사용했을 경우 부호화 효율
Table 5. Performance with 1/8-pel MV resolution

해상도	실험 영상	BD-Rate (%)
Class D	BasketballPass	1.9
	BQSquare	-5.7
	Blowing Bubbles	-0.4
	RaceHorses	2.4
Average		-0.5

표 5에서는, HM3.0에서 사용하는 1/4 화소 단위 움직임 벡터 해상도보다 높은 1/8 단위 움직임 벡터 해상도를 적용한 결과로써, 영상의 공간 복잡도가 높고 움직임이 느린 실험 영상에서는 부호화 효율 향상의 효과를 얻을 수 있었지만, 상대적으로 움직임이 크고 많이 발생하는 실험 영상에서는 부호화 효율이 저하되는 현상을 보인다.

표 4와 표 5의 실험 결과를 종합해보면, 낮은 해상도의 움직임 벡터는 움직임 큰 영상의 부호화에서 효과적으로 사용되고, 높은 해상도의 움직임 벡터는 영상의 움직임이 상대적으로 적고 공간 복잡도가 높은 영상에서 부호화 효율을 향상시킬 수 있었다. 이러한 경향은 영상에서 물체의 움직임 크기와 양이 최적의 움직임 벡터 해상도와 관련이 있음을 나타낸다. 일반적으로, 움직임이 크고 복잡한 영상의 경우에는 움직임 예측의 정확도가 낮고, 움직임이 적은 영상에서는 움직임 예측이 비교적 정확하게 이루어진다. 이러한 경향은, 실제 움직임 벡터들이 예측 움직임 벡터로부터 떨어진 거리, 즉 차이 움직임 벡터의 크기를 고려하면 최적의 움직임 벡터 해상도를 결정할 수 있음을 의미한다. 따라서 본 논문에서는 차분 움직임 벡터의 크기를 고려하는 움직임 벡터의 해상도 결정 알고리즘을 제안한다.

2. 차분 움직임 벡터의 코드 번호 할당방법

차분 움직임 벡터의 크기를 고려하는 적응적인 움직임

벡터 해상도 결정 방법은 그림 1과 같이 표현할 수 있다. 움직임 벡터가 예측 움직임 벡터를 중심으로 가까운 거리에 존재한다면 차분 움직임 벡터에 대한 발생비트가 적기 때문에 1/8 화소 단위까지 움직임 벡터 해상도를 고려하여 움직임을 추정한다. 반대로 차분 움직임 벡터의 절대값의 크기가 큰 경우에는 정수 화소 단위의 움직임 벡터만을 고려하여 차분 움직임 벡터를 부호화하는 과정에서 발생하는 비트량을 감소시켜 부호화 효율을 높인다.

그림 1. 차분 움직임 벡터의 크기에 따른 움직임 벡터 해상도 결정구간
Fig. 1. Ranges for MV resolutions based on absolute value of MVD

그림 1에서 $S_0 \sim S_3$ 은 각각 1/8 화소 단위부터 1/4, 1/2, 정수 화소 단위의 움직임 벡터 해상도를 사용하는 구간을 의미하며, $Th_0 \sim Th_3$ 은 각 구간을 분할하는 경계를 표현하는 Threshold들을 나타낸다.

그림 1과 같이 구간별로 사용하는 움직임 벡터 해상도가 달라짐에 따라, 제안하는 알고리즘에서는 차분 움직임 벡터를 부호화할 때 사용되는 새로운 코드 번호 할당 방법이 필요하다. 기존에 HM3.0에서는 코드 번호 할당이 그림 2와 같이 균일한 간격으로 이루어진다.

그림 2. 차분 움직임 벡터에 대한 기존 코드 번호 할당 방법
Fig. 2. Conventional code number assignment for MVDs

그림 3. 차분 움직임 벡터에 대한 제안하는 코드 번호 할당 방법
Fig. 3. Proposed code number assignment for MVDs

제안하는 방법에서는 기존 HM3.0과 달리, 다수개의 움직임 벡터 해상도가 구간별로 다르게 설정됨에 따라 그림 3과 같이 적응적인 코드 번호 할당 방법이 적용된다.

그림 3에서와 같이, 제안하는 코드 번호 할당 방법에서는, S_0 에 해당하는 구간 즉, 차분 움직임 벡터가 $Th_0 \leq |MVD| < Th_1$ 에 해당되는 경우에는 해당 차분 움직임 벡터는 1/8 화소 단위의 움직임 벡터 해상도가 적용되어 부호화되며, 이 같은 순서로 각 구간에 해당되는 차분 움직임 벡터는 1/4, 1/2, 정수 화소 단위 움직임 벡터 해상도를 적용하여 부호화 된다. 움직임 벡터 해상도가 낮아짐에 따라 코드가 할당되는 되는 간격이 점차 넓어진다. 따라서 기존 방법과 제안하는 방법을 이용하여 동일한 크기의 큰 차분 움직임 벡터를 부호화 할 경우에, 작은 코드 번호를 할당하게 되어 작은 비트량으로 차분 움직임 벡터를 표현할 수 있게 된다. 예를 들어 차분 움직임 벡터의 크기가 24인 경우, 그림 1의 기존 방법을 적용하게 되면, 코드 번호가 47이 할당되어 표 3에 따라 11비트로 표현된다. 반면, 제안하는 방법을 적용하게 되면, 17이 할당되어 상대적으로 더 짧은 비트 스트링으로 차분 움직임 벡터를 표현할 수 있다.

차분 움직임 벡터를 부호화 할 때 차분 움직임 벡터의 x 컴포넌트와 y 컴포넌트의 크기 차이가 존재하기 때문에 움직임 벡터의 해상도를 결정하는 과정에 모호함이 존재한

다. 이 문제를 해결하기 위해서 먼저 차분 움직임 벡터의 x축과 y축 컴포넌트들의 절대값의 크기를 비교하여 그 값이 더 큰 차분 움직임 벡터를 기준으로 현재 부호화하려는 움직임 벡터의 해상도를 결정한다. 결정된 움직임 벡터의 해상도에 따라 현재 차분 움직임 벡터에 해당하는 새로운 코드 번호를 할당하고 부호화를 수행한다. 예측 움직임 벡터의 위치와 차분 움직임 벡터의 위치가 그림 4와 같이 존

그림 4. 차분 움직임 벡터 크기에 따른 움직임 벡터 해상도 결정 방법
Fig. 4. MV resolution decision based on absolute value of MVD

재 할 경우를 예로 설명하자면, 현재 x 좌표에 해당하는 차분 움직임 벡터의 절대값 크기가 y 좌표의 절대값 크기보다 큰 값을 가지므로, x 좌표의 차분 움직임 벡터를 이용하여 움직임 벡터의 해상도를 결정하게 된다. 따라서 그림 4에서는 x 좌표의 차분 움직임 벡터가 포함되는 영역에 할당된 1/2 화소 단위 움직임 벡터 해상도가 적용된다. 복호기에서는 먼저 두 개의 좌표를 모두 복호화 하여 복호 된 값의 절대값 크기가 큰 좌표 값을 복원하고, 복원된 값을 이용하여 현재 사용되는 움직임 벡터의 해상도를 계산하여 차분 움직임 벡터의 크기가 작은 값을 복원한다.

3. 예측 움직임 벡터의 보정

본 논문에서 제안하는 알고리즘은 움직임 벡터의 해상도를 결정함에 있어서, 차분 움직임 벡터의 크기를 이용한다. 이에 따라, 보다 명확한 알고리즘의 설계를 위하여 예측 움직임 벡터의 해상도역시 움직임 벡터와 동기화 시킬 필요가 있다. 예측 움직임 벡터의 동기화는 다음 수식을 이용하여 이루어진다.

$$PMV_r = \begin{cases} PMV_{1/8} & \text{if } r=0 \\ \lfloor PMV_{1/8}/2 \rfloor \times 2 & \text{if } r=1 \\ \lfloor PMV_{1/8}/4 \rfloor \times 4 & \text{if } r=2 \\ \lfloor PMV_{1/8}/8 \rfloor \times 8 & \text{otherwise} \end{cases} \quad (5)$$

수식에서 PMV 는 예측 움직임 벡터를 의미하며, r 은 움직임 벡터의 해상도를 나타내는 색인 정보이다. 각 해상도와 색인 r 의 관계는 표 6과 같다.

표 6. 색인 r 에 따른 움직임 벡터의 해상도

Table 6. MV resolution and index r

움직임 벡터 해상도	색인 (r)
1/8-pel	0
1/4-pel	1
1/2-pel	2
1-pel	3

IV. 최적의 Threshold 결정방법

본 절에서는 각 구간을 분할하는 Threshold 값들을 최적화하기 위한 알고리즘을 제안한다. 제안하는 알고리즘은 총 3 단계로 이루어져있다. 1 단계에서는 정보 수집을 수행하고 2 단계에서는 최적의 Threshold들을 결정한다. 그리고 마지막 단계에서는 최적화된 Threshold를 적용하여 현재 슬라이스의 부호화를 수행하게 된다.

1 단계에서는 현재 슬라이스에 대한 움직임 정보를 수집한다. 이를 위하여 현재 슬라이스에 대하여 최대 움직임 벡터 해상도인 1/8 화소 단위를 적용하고 부호화를 수행한다. 이 과정에서 각 해상도에 해당하는 움직임 예측의 결과를 저장하도록 하는데, 그 중 움직임 예측 오차와 차분 움직임 벡터를 저장한다. 저장된 움직임 예측 오차는 각 해상도에 따라 $Distortion_X(j)$ 에 저장하며, X 는 각 움직임 벡터의 해상도의 색인을 나타내고 j 는 현재 슬라이스 내에서 인터모드로 부호화 된 Prediction Unit (PU)의 색인을 나타낸다. 차분 움직임 벡터는 $MVD_X(j)$ 에 저장한다.

1 단계에서 저장한 각 PU와 해상도 별 예측 오차들과 차분 움직임 벡터들을 이용하여 2 단계에서는 Threshold의 최적화를 수행한다. 가장 먼저 각 구간의 경계를 나타내는 Threshold인 $\{Th_0, Th_1, Th_2, Th_3\}$ 을 $\{0, SR_{max}, SR_{max}, SR_{max}\}$ 로 초기화한다. 여기서 SR_{max} 는 움직임 예측의 최대 범위를 나타낸다. 각 Threshold 값들을 갱신할 때 사용할 $\{\Delta Th_0, \Delta Th_1, \Delta Th_2, \Delta Th_3\}$ 는 $\{0, 2, 4, 8\}$ 로 설정하여 최대 움직임 벡터 해상도인 1/8 화소 단위 해상도를 기준으로, 각 움직임 벡터 해상도에 따라 적절하게 Threshold 값들을 갱신 할 수 있도록 한다. 그리고 Threshold를 갱신할 때, 사용할 색인인 k 는 0으로, 움직임 벡터 해상도를 표현하는 색인인 r 은 1로 초기화한다. 이와 같이 초기화 작업이 마무리 되면, 각 움직임 벡터 해상도에 대한 Threshold들을 아래와 같은 수식으로 계산한다.

$$Th_r = \Delta Th_r \times k \quad (6)$$

수식 (6)을 통해 변경된 하나의 Threshold 값 Th_r 을 반

영하여 1 단계에서 인터 예측으로 부호화된 PU들의 비용함수를 다시 계산하여 현재 슬라이스의 새로운 비용함수를 다음과 같이 계산한다.

$$J^S(k) = \sum_{j=1}^T J^{Th}(j) \quad (7)$$

$$J^{Th}(j) = \text{Distortion}_r(j) + \lambda R^{Th}(j) \quad (8)$$

여기서, $J^S(k)$ 는 계산된 Threshold들을 반영한 현재 슬라이스의 비용함수를 의미한다. 그리고 $J^{Th}(j)$ 는 인터 PU 단위로 계산한 비용함수를 나타내며, 수식 (8)을 이용하여 계산한다. T 는 1 단계에서 얻어진 정보로써, 현재 슬라이스에서 인터로 부호화 된 PU들의 수를 나타낸다. 수식 (8)에서 $R^{Th}(j)$ 는 현재 계산된 Threshold들을 반영하여 변화된 코드 번호를 사용하여 움직임 벡터의 발생 비트량을 계산한 결과이다. 수식 (6)~(8)을 이용하여 Th_r 의 값이 최대 움직임 예측 영역이 넘지 않는 범위에서 k 를 1씩 증가시키면서 $J^S(k)$ 값들을 계산한다. 만약, Th_r 의 값이 최대 움직임 예측 영역을 넘어서게 되면, 현재까지 계산한 $J^S(k)$ 들 중 가장 작은 값을 갖도록 하는 하나의 k 를 아래 수식을 이용하여 선택한다.

$$k^* = \arg \min J^S(k) \quad (9)$$

수식 (9)에서 선택된 k^* 는 현재 고려하는 움직임 벡터 해상도의 Threshold 계산에 다음과 같이 적용된다.

$$Th_r = \Delta Th_r \times k^* \quad (10)$$

수식 (10)의 계산 후에, 현재 움직임 벡터 해상도의 색인인 r 의 값이 3보다 크거나 같으면 모든 움직임 벡터 해상도의 Threshold 값이 계산되었음을 의미하여 최적화 과정을 마무리하고, 그렇지 않으면, r 을 1 증가시키고, 다음 움직임 벡터 해상도의 Threshold를 계산하기 위한 준비과정을 수행한다. 이 과정에서는 높은 해상도의 Threshold 값들보다 큰 범위에서 현재 구하고자 하는 움직임 벡터 해상도의

Threshold를 계산하기 위하여, 아래와 같이 k 의 초기값 설정을 위한 계산을 수행한다.

$$m^* = \arg \min \{m | \Delta Th_r \times m \geq Th_{r-1}\} \quad (10)$$

위의 수식에서 계산된 m^* 의 값은 현재 구하고자하는 Threshold를 계산하는데 사용되는 k 의 초기값으로 설정된다. 이러한 순서로 수식 (6)~(11)을 이용하여 모든 움직임 벡터 해상도에 해당하는 Threshold 값을 최적화할 수 있다. 그림 5는 본 절에서 설명한 Threshold 최적화 과정을 정리한 블록 다이어그램이다.

그림 5. 최적의 Threshold 결정을 위한 블록 다이어그램

Fig. 5. Block diagram for optimal threshold decision

마지막 3 단계에서는 그림 4를 이용하여 결정된 최적의 Threshold 값들을 이용하여 현재 슬라이스를 부호화한다. 이 때, 각 Threshold들은 복호화기에서도 부호화기와 동일하게 적용해야하기 때문에 해당 정보를 슬라이스 단위로 부호화한다. 각 Threshold 값을 그대로 전송되지 않고, 수식 (12)를 이용하여 계산한 결과값을 전송한다.

$$v_r = \frac{Th_r}{\Delta Th_r} \quad (11)$$

수식 (12)에서 계산한 모든 v_r 값들은 슬라이스 헤더에 새로운 선택스를 추가하여 부호화한다.

V. 실험 결과 및 분석

본 절에서는 제안하는 적응적인 움직임 벡터 해상도를 적용한 실험 결과를 보인다. 실험에서는 HEVC의 참조 소프트웨어인 HM3.0을 사용하였으며, JCT-VC에서 권고하는 일반 실험 조건을 적용하여 제안하는 알고리즘을 검증하였다^{[10][12]}. 현재 HM3.0에서 1/8 화소 단위의 움직임 벡터 해상도를 사용하지 않는 관계로, 해당 실험에서는 표 7과 표 8에서 보이는 DCT-IF 8tap를 사용하였다^[13-14]. 실험 결과에서 부호화 효율은 BD-Rate를 사용하여 측정하였으며, 결과가 음수값을 가지면, 비교 대상과 비교해서 부호화 효율이 향상되었음을 의미하고, 양수값을 가지면 부호화 효율이 저하되었음을 나타낸다. 표 9~16에서 RA는 ‘Random Access’, LB는 ‘Lowdelay with B picture’, LP는 ‘Low delay with P picture’ 부호화 구조를 의미하며, HE는 ‘High Efficiency’를 LC는 ‘Low Complexity’ 실험 조건을

표 7. DCT-IF 8 taps 계수 (밝기신호)

Table 7. Coefficients of DCT-IF 8 taps (Luma component)

Position	Filter Coefficients
1/8	{-1, 3, -6, 62, 9, -4, 2, -1}
3/8	{-2, 5, -12, 50, 30, -10, 4, -1}
5/8	{-1, 4, -10, 30, 50, -12, 5, -2}
7/8	{-1, 2, -4, 9, 62, -6, 3, -1}

의미한다.

표 8. DCT-IF 8 taps 계수 (색차신호)

Table 8. Coefficients of DCT-IF 8 taps (Chroma component)

Position	Filter Coefficients
1/16	{-2, 63, 4, -1}
3/16	{-5, 59, 13, -3}
5/16	{-6, 52, 23, -5}
7/16	{-7, 43, 34, -6}
9/16	{-6, 34, 43, -7}
11/16	{-5, 23, 52, -6}
13/16	{-3, 13, 59, -5}
15/16	{-1, 4, 63, -2}

표 9~11은 HM3.0과 부호화 성능 차이를 보여준다. 제안하는 적응적인 움직임 벡터 해상도 알고리즘은 Random

표 9. Random Access 부호화 구조에서의 부호화 성능

Table 9. Performance for Random Access coding structure

해상도	실험 영상	RA-HE	RA-LC
		BD-Rate (%)	BD-Rate (%)
Class A (2560x1600)	Traffic	-0.3	-0.4
	PeopleOnStreet	-0.3	-0.5
	Nebuta	0	-0.1
	SteamLocomotive	-0.9	-1.0
Class A Average		-0.4	-0.5
Class B (1920x1080)	Kimono	-0.8	-1.1
	ParkScene	-0.2	-0.3
	Cactus	-0.4	-0.5
	BasketballDrive	-0.3	-0.5
	BQTerrace	-1.2	-1.5
Class B Average		-0.6	-0.8
Class C (832x480)	BasketballDrill	-0.7	-1.0
	BQMall	-0.5	-0.6
	PartyScene	-1.2	-1.9
	RaceHorses	-0.3	-0.4
Class C Average		-0.7	-1.0
Class D (416x240)	BasketballPass	-0.3	-0.5
	BQSquare	-3.1	-5.0
	Blowing Bubbles	-0.8	-1.3
	RaceHorses	-0.3	-0.4
Class D Average		-1.1	-1.8
Total Average		-0.7	-1.0

표 10. Low delay with B picture 부호화 구조에서의 부호화 성능

Table 10. Performance for Low delay with B picture coding structure

해상도	실험 영상	LB-HE	LB-LC
		BD-Rate (%)	BD-Rate (%)
Class B (1920x1080)	Kimono	-0.4	-1.2
	ParkScene	-0.1	-0.2
	Cactus	-0.2	-0.2
	BasketballDrive	-0.3	-0.6
	BQTerrace	-0.1	-0.6
Class B Average		-0.2	-0.5
Class C (832x480)	BasketballDrill	-0.5	-0.8
	BQMall	-0.3	-0.4
	PartyScene	-1.1	-1.9
	RaceHorses	-0.2	-0.6
Class C Average		-0.5	-0.9
Class D (416x240)	BasketballPass	-0.2	-0.5
	BQSquare	-3.3	-5.7
	Blowing Bubbles	-0.6	-1.0
	RaceHorses	-0.1	-0.3
Class D Average		-1.1	-1.9
Class E (1280x720)	Vidyo 1	0.5	0.2
	Vidyo 2	0	0.6
	Vidyo 3	0.2	0.4
Class E Average		0.2	0.4
Total Average		-0.4	-0.8

Access 부호화 구조에서는 평균적으로 약 0.9%의 성능 향상을 얻을 수 있었으며, Low delay with B picture 부호화 구조에 적용한 경우는 약 0.6%, Low delay with P picture의 경우에는 약 2.7%의 평균 발생 비트량 감소를 확인할 수 있었다.

Low delay with B picture를 사용하는 부호화 구조에서는 HM3.0의 일반 실험 조건에서 Generalized P and B pictures (GPB)를 사용한다. GPB가 적용되면 과거 참조 영상만 존재하는 경우에도 양방향 예측을 수행함으로써 부호화 효율을 향상시키는 효과를 얻게 된다. 게다가 GPB에 적용된 양방향 예측으로 인하여 때때로 영상의 보간 효과가 발생하게 되는데 Low delay with P Picture 부호화 구조에서는 GPB가 사용되지 못하여 이러한 효과를 가지지 못한다. 이 때문에 제안하는 적응적 움직임 벡터 해상도 기술이 해당 부호화 구조에서 더욱 효과적으로 동작하

표 11. Low delay with P picture 부호화 구조에서의 부호화 성능

Table 11. Performance for Low delay with P picture coding structure

해상도	실험 영상	LP-HE	LP-LC
		BD-Rate (%)	BD-Rate (%)
Class B (1920x1080)	Kimono	-0.5	-0.8
	ParkScene	-0.3	-0.4
	Cactus	-0.5	-0.4
	BasketballDrive	-0.4	-0.4
	BQTerrace	-6.5	-6.0
Class B Average		-1.6	-1.6
Class C (832x480)	BasketballDrill	-1.6	-1.3
	BQMall	-2.2	-2.9
	PartyScene	-7.0	-6.5
	RaceHorses	-0.2	-0.1
Class C Average		-2.7	-2.7
Class D (416x240)	BasketballPass	-0.3	-0.3
	BQSquare	-16.7	-16.1
	Blowing Bubbles	-3.7	-3.6
	RaceHorses	-0.1	-0.3
Class D Average		-5.2	-5.1
Class E (1280x720)	Vidyo 1	-0.8	-0.2
	Vidyo 2	-3.9	-0.7
	Vidyo 3	-1.5	-0.3
Class E Average		-2.1	-0.4
Total Average		-2.9	-2.5

게 된다.

다양한 환경에서 제안하는 알고리즘을 평가하기 위하여, HM3.0에 1/8 화소 단위 움직임 벡터 해상도를 적용한 부호화 결과와 제안하는 알고리즘의 부호화 결과를 비교하였다. 이 때, 1/8 화소 단위의 영상 보간을 대하여 제안하는 기술과 같이 표 7과 표 8에 해당하는 보간 필터를 사용하였다. 표 12~14은 각 부호화 구조와 실험 조건에 따른 부호화 성능 차이를 보여준다. 여기서 ‘HM3.0+1/8-pel’은 HM3.0에 1/8 화소 단위 움직임 벡터 해상도를 적용한 결과와 비교했음을 나타낸다.

제안하는 방법으로 적응적으로 움직임 벡터 해상도를 적용한 결과가 ‘HM3.0+1/8-pel’의 결과보다 Random Access 구조에서는 약 1.5%, Low delay with B picture 구조에서는 약 1.7%, 그리고 Low delay with P picture 구조에서는 약 1.0%의 평균 부호화 효율 향상을 얻을 수 있었다. 본 실험

에서는 앞의 표 9~11의 경향과는 반대로 영상의 크기가 큰 영상에서 높은 성능 향상을 확인할 수 있다. 이는 1/8 화소 단위로 움직임 벡터를 표현하는데 사용되는 비트량이 증가했기 때문이다. 또한 반대 특성을 갖는 영상에 대한 결과도 앞의 HM3.0과 비교한 결과와 반대 경향을 보임을 확인할 수 있다.

표 9~14를 정리해보면 Random Access 구조와 Low delay with B picture 구조에서는 HM3.0이 HM3.0+1/8-pel 보단 성능이 높고 Low delay with P picture 구조에서는 반대로 HM3.0+ 1/8-pel의 성능이 HM3.0보다 더 높음을 알 수 있다. 이러한 이유는 앞에서 설명한 것처럼 GPB 때문이다. 따라서 Random Access구조나 Low delay with B picture구조에서는 움직임 벡터를 1/8 화소 단위까지 고려하면 차분

움직임 벡터를 효과적으로 부호화 할 수가 없어서 움직임 벡터를 1/4 화소 단위까지 고려하여 부호화 하는 것보다 성능이 떨어지게 되는 것이다. 반대로 Low delay with P picture 구조에서는 이러한 문제점이 없기 때문에 움직임 벡터를 1/8 화소 단위까지 고려한 것이 1/4 화소 단위까지 고려한 것보다 성능이 높다. 하지만 이러한 Low delay with P picture구조에서도 제안하는 알고리즘은 부호화 효율을 1.0% 향상시키고 있다.

실험 결과 본 논문에서 제안하는 Threshold 최적화와 적응적인 움직임 벡터 해상도 조절 알고리즘이 효과적으로 움직임 벡터 해상도를 구간별로 할당함으로써 기존에 움직임 벡터의 해상도를 고정하여 부호화 하는 것보단 부호화 성능을 향상시킴을 확인 할 수 있다.

표 12. Random Access 구조에서의 부호화 성능 (vs. HM3.0+1/8-pel)
Table 12. Performance for Random Access coding structure (vs. HM3.0+1/8-pel)

해상도	실험 영상	RA-HE	RA-LC
		BD-Rate (%)	BD-Rate (%)
Class A (2560x1600)	Traffic	-0.9	-1.0
	PeopleOnStreet	-2.7	-3.3
	Nebuta	-0.5	-1.0
	SteamLocomotive	-2.2	-3.2
Class A Average		-1.5	-2.1
Class B (1920x1080)	Kimono	-2.2	-2.7
	ParkScene	-0.9	-1.1
	Cactus	-0.8	-1.1
	BasketballDrive	-1.5	-2.1
Class B Average		-1.2	-1.6
Class C (832x480)	BasketballDrill	-1.2	-1.7
	BQMall	-0.9	-1.0
	PartyScene	-0.4	-0.3
	RaceHorses	-2.0	-2.7
Class C Average		-1.1	-1.4
Class D (416x240)	BasketballPass	-1.9	-2.4
	BQSquare	-0.1	-0.1
	Blowing Bubbles	-0.8	-0.6
	RaceHorses	-2.5	-3.1
Class D Average		-1.3	-1.6
Total Average		-1.3	-1.7

표 13. Low delay with B picture 구조에서의 부호화 성능 (vs. HM3.0+1/8-pel)

Table 13. Performance for Low delay with B picture coding structure (vs. HM3.0+1/8-pel)

해상도	실험 영상	LB-HE	LB-LC
		BD-Rate (%)	BD-Rate (%)
Class B (1920x1080)	Kimono	-1.6	-2.9
	ParkScene	-1.0	-1.8
	Cactus	-1.0	-2.1
	BasketballDrive	-1.4	-2.1
	BQTerrace	-0.7	-1.6
Class B Average		-1.1	-2.1
Class C (832x480)	BasketballDrill	-1.9	-1.9
	BQMall	-0.9	-1.2
	PartyScene	-0.2	-0.2
	RaceHorses	-1.8	-2.8
Class C Average		-1.2	-1.5
Class D (416x240)	BasketballPass	-1.8	-2.4
	BQSquare	0	0
	Blowing Bubbles	-0.4	-0.3
	RaceHorses	-2.3	-2.8
Class D Average		-1.1	-1.4
Class E (1280x720)	Vidyo 1	-2.2	-3.3
	Vidyo 2	-1.5	-4.5
	Vidyo 3	-2.0	-3.4
Class E Average		-1.9	-3.7
Total Average		-1.3	-2.1

표 14. Low delay with P picture 구조에서의 부호화 성능 (vs. HM3.0+1/8-pel)

Table 14. Performance for Low delay with P picture coding structure (vs. HM3.0+1/8-pel)

해상도	실험 영상	LP-HE	LP-LC
		BD-Rate (%)	BD-Rate (%)
Class B (1920x1080)	Kimono	-1.5	-2.4
	ParkScene	-0.3	-0.9
	Cactus	-0.4	-1.8
	BasketballDrive	-0.1	-0.6
	BQTerrace	-0.1	-0.2
Class B Average		-0.5	-1.2
Class C (832x480)	BasketballDrill	-0.9	-1.4
	BQMall	0	-1.4
	PartyScene	0	0
	RaceHorses	-0.8	-1.4
Class C Average		-0.4	-1.0
Class D (416x240)	BasketballPass	-1.4	-1.8
	BQSquare	0.1	0.1
	Blowing Bubbles	0	-0.1
	RaceHorses	-1.9	-2.3
Class D Average		-0.8	-1.0
Class E (1280x720)	Vidyo 1	-1.3	-2.4
	Vidyo 2	-0.9	-2.0
	Vidyo 3	-1.5	-2.0
Class E Average		-1.2	-2.1
Total Average		-0.7	-1.3

표 15와 표 16은 제안하는 알고리즘의 부호화기와 복호화기 복잡도를 HM3.0과 비교한 결과이다. 부호화기는 복잡한 Threshold 최적화 단계가 적용되어 HM3.0에 비하여 높은 복잡도를 가진다. 표 15에서는 제안하는 알고리즘을 사용하는 부호화기는 HM3.0과 비교하여 평균적으로 약 4배 정도의 복잡도를 갖음을 보인다. 이러한 이유는 기존의 HM3.0은 움직임 벡터를 1/4 화소 단위까지만 고려하여 부호화 하는 반면에 제안하는 알고리즘은 1/8 화소 단위까지 고려하기 때문에 움직임 예측하는데 복잡도가 증가 되며 또한 Threshold값을 최적화 하는 단계에서도 복잡도가 증가된다. 그리고 무엇보다 이렇게 복잡해진 과정을 여러 번 부호화하기 때문에 복잡도가 크게 증가하게 된다. 반면에 복호화기는 표 16에 나타나진 것처럼 HM3.0의 복호화기 복잡도와 비교해서 약 2% 정도만 복잡함을 나타내고 있다. 복호기에서는 움직임 벡터를 예측할 필요도 없고, Threshold값을 만들 필요도 없으며, 여러 번 복호화할 필요도 없기 때문이다. 다만 제안하는 기술을 이용하였기 때문에 복호화기에서 수신된 인덱스 값을 처리하여 차분 움직임 벡터를 복호화하는 추가의 과정이 필요하다. 그래서 제안하는 알고리즘의 복호기 복잡도가 기존 HM3.0보다 약 2%정도 더 복잡하다.

표 15. 제안하는 알고리즘을 적용한 인코더의 복잡도

Table 15. Encoder complexity of the proposed algorithm

해상도	RA-HE (%)	RA-LC (%)	LB-HE (%)	LB-LC (%)	LP-HE (%)	LP-LC (%)
Class A	377	362	x	x	x	x
Class B	373	416	438	382	348	352
Class C	364	425	450	351	327	340
Class D	374	431	460	363	355	417
Class E	x	x	403	317	369	389
Average	373	408	440	357	349	371

표 16. 제안하는 알고리즘을 적용한 디코더의 복잡도

Table 16. Decoder complexity of the proposed algorithm

해상도	RA-HE (%)	RA-LC (%)	LB-HE (%)	LB-LC (%)	LP-HE (%)	LP-LC (%)
Class A	103	101	x	x	x	x
Class B	101	104	98	100	100	102
Class C	102	103	100	102	100	101
Class D	102	105	102	103	101	102
Class E	x	x	101	101	102	103
Average	102	103	99	101	101	102

VI. 결 론

본 논문에서는 차분 움직임 벡터의 크기에 따라 움직임 벡터의 해상도를 결정하는 기술을 제안하였다. 제안하는 알고리즘은 차분 움직임 벡터의 절대값 크기가 작으면 높은 해상도의 움직임 벡터를 사용하여 움직임 예측 정확도를 높임으로써 부호화 효율을 높이고, 반대로 차분 움직임 벡터의 크기가 크면 낮은 해상도의 움직임 벡터를 사용하여 차분 움직임 벡터에 할당되는 비트량을 감소 시켜서 부호화 효율을 향상시킨다. 실험을 통하여 제안하는 알고리즘이 HM3.0과 비교하여 Random Access 부호화 구조에서는 평균적으로 약 0.9%의 성능 향상을 얻을 수 있었으며, Low Delay 부호화 구조에 B picture를 적용한 경우는 약 0.6%, P picture를 적용한 경우에는 약 2.7%의 평균 발생 비트량 감소를 확인할 수 있었다. 제안하는 알고리즘은 인코더의 복잡도를 증가시키는 반면 복호화기의 복잡도 증가가 거의 없기 때문에, 실시간 방송과 같은 시스템 보다는 DVD 또는 VOD 서비스와 같은 시스템에 적용하는 것이 적절하다. 그리고 차기 연구 주제로 부호화 시간 증가의 대부분을 차지하는 Threshold 최적화 과정의 단순화에 대한 연구를 진행 중이다.

참 고 문 헌

- [1] Committee Consultative International Telegraphic et Telegraph (CCITT), H261 video for audiovisual services at p*64 kbits, Geneva, 1990
- [2] Int. Telecommun. Union-Telecommun. (ITU-T) and Int. Standards Org./Int. Electrotech. Comm. (ISO/IEC) JTC 1, Rec. H262 and ISO/IEC 13 818-2 (MPEG-2 Video), Generic Coding of Moving Pictures and Associated Audio Information-Part 2: Video 1994,11
- [3] Int. Standards Org./Int. Electrotech. Comm. (ISO/IEC) JTC 1, ISO/IEC 14496-2 (MPEG-4 visual version 1), Coding of Audio-Visual Objects-Part 2: Visual 1999.
- [4] ITU-T Recommendation H.264 and ISO/IEC 14496-10 AVC "Advanced video coding for generic audiovisual services," version 3: 2005
- [5] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard" IEEE Trans. Circuits Syst. Video Technol., Vol. 13, no. 7, pp. 560~576, 2003. 7.
- [6] Draft ITU-T Recommendation and Final Draft International Standard of Joint Specification (ITU-T Rec. H.264/IEC 14496-10 AVC). Mar, 2003
- [7] ITU-T VCEG, Reference ITU-T VCEG-KTA Software is available at <http://iphome.hhi.de/suehring/tml/download/KTA>
- [8] J.Ostremann and M.Narroschke, "Motion compensated prediction with 1/8-pel displacement vector resolution", ITU-T SG16 Q6 VCEG-AD09, 2006, 10
- [9] T. Wiegand, B. Bross, W.-J. Han, J.-R. Ohm, and G. J. Sullivan, "WD3: Working Draft 3 of High Efficiency Video Coding", JCTVC-E603, Geneva, Switzerland, Mar, 2011
- [10] K. McCann, B. Bross, S. Sekiguchi, W.-J. Han, "HM3: High Efficiency Video Coding (HEVC) Test Model 3 Encoder Description", JCTVC-E602, Geneva, Switzerland, Mar., 2011.
- [11] M. H. Jang, C.W. Seo, and J. K. Han "Motion Estimation using Region-based Adaptive Motion Vector Prediction," The 26th International Technical Conference on Circuits/Systems, Computers and Communications, Jun. 20-22 2011.
- [12] F. Bossen, "Common test conditions and software reference configuration", JCTVC-E700, Geneva, Switzerland, Mar, 2011
- [13] K. McCann, W.-J. Han and I-K Kim, "Samsung's Response to the Call for Proposals on Video Compression Technology," JCTVC-A124, Dresden, Germany, April 2010.
- [14] E. Alshina, J. Chen, A. Alshin, N. Shlyakhov, and W.-J. Han, "CE3: Experimental results of DCTIF by Samsung", JCT-VC meeting contribution JCTVC-D344, Daegu, KR, Jan., 2011.

저 자 소 개

장 명 훈

- 2010년 : 세종대학교 정보통신공학과 학사
- 2010년 ~ 현재: 세종대학교 정보통신공학과 석사과정
- 주관심분야 : HEVC, H.264/AVC, 영상처리

서 찬 원

- 2007년 : 세종대학교 정보통신공학과 학사
- 2009년 : 세종대학교 정보통신공학과 석사
- 2010년 ~ 현재 : 세종대학교 정보통신공학과 박사과정
- 주관심분야 : HEVC, H.264/AVC, SVC

한 종 기

- 1992년 : 한국과학기술원(KAIST) 전기 및 전자공학과 학사
- 1994년 : 한국과학기술원(KAIST) 전기 및 전자공학과 석사
- 1999년 : 한국과학기술원(KAIST) 전기 및 전자공학과 박사
- 1999년 2월 ~ 2001년 8월 : 삼성전자 DM 연구소 책임 연구원
- 2001년 9월 ~ 2006년 2월 : 세종대학교 정보통신공학과 조교수
- 2006년 3월 ~ 2011년 8월 : 세종대학교 정보통신공학과 부교수
- 2011년 9월 ~ 현재 : 세종대학교 정보통신공학과 교수
- 주관심분야 : HEVC, H.264/AVC, SVC, DVC, Transcoding, 영상처리, 신호처리