



저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

공학석사 학위논문

조건부 Bilinear SATD 추정을
이용한 HEVC Fractional Motion
Estimation 의 복잡도 개선

A Complexity Reduction Algorithm of the
Fractional Motion Estimation in HEVC using
Conditional Bilinear SATD Estimation

2016 년 2 월

서울대학교 대학원

전기 정보 공학부

임 대 빈

조건부 Bilinear SATD 추정을 이용한 HEVC Fractional Motion Estimation 의 복잡도 개선

지도 교수 채 수 익

이 논문을 공학석사 학위논문으로 제출함
2016 년 2 월

서울대학교 대학원
전기 정보 공학부
임 대 빈

임대빈의 공학석사 학위论문을 인준함
2016 년 2 월

위 원 장 _____ 최 기 영 (인)

부위원장 _____ 채 수 익 (인)

위 원 _____ 이 혁 재 (인)

초 록

HEVC encoder 의 inter prediction 은 시간적으로 인접한 picture 들 간의 높은 상관성을 이용하여 현재 블록에 대한 움직임을 예측하는 과정으로 reference picture list 에서 현재 블록과 움직임이 제일 유사한 prediction unit (PU)을 선택하고 현재 블록과의 차분 값을 부호화 한다. Motion estimation (ME)를 수행하여 발생한 차분 값의 크기가 압축 성능을 좌우하게 되므로 ME 의 예측 정밀도가 결국 전체 코딩 효율 결정한다고 할 수 있는데 ME 는 크게 정수 화소 단위의 움직임을 추정하는 integer motion estimation (IME)와 정수 화소 단위의 이하의 움직임을 추정하는 fractional motion estimation (FME)로 구분할 수 있다. IME 에서 각 PU partition 에 대해 정수 화소 단위에서의 최적 motion vector (IMV)를 찾은 후 FME 에서 현재 picture 의 해상력을 4 배까지 증가 시키는 interpolation 을 통해 integer-pel 사이의 fractional-pel 을 생성한 뒤 half-pel search 와 quarter-pel search 로 이어지는 정밀한 탐색을 수행하여 최적의 fractional motion vector (FMV)를 찾게 된다. HEVC 에서는 FME 를 수행하기 위해 reference picture 로부터 참조하는 reference block 은 이미 quantization/de-quantization 을 수행한 후 de-blocking filter 나 sample adaptive offset 을 수행하여 재구성한 신호로, 여기에 7/8 탭 low pass filtering 을 적용한 interpolation 을 수행하여 fractional-pel 을 생성 한다.

제안하는 FME 알고리즘 에서는 Half-pel search 수행 시 IMV 위치의 best integer-pel 과 그 주위를 둘러싸고 있는 8 개 pixel 에

대한 PU 의 sum of absolute transformed differences (SATD)-based distortion 값들의 최대값과 최소값의 차이에 대한 비율을 확인하여, 일정 threshold 이하 이면 SATD distortion 이나 재구성된 영상이 모두 locally uniform 하다고 판단하고 HEVC test model(HM)과 같은 pixel-interpolation 대신 SATD 값을 이용하는 bilinear interpolation 을 통해 각 half pixel 위치에 대한 SATD distortion 값을 구한다. 그리고 각 위치 별 motion vector(MV)의 rate 값을 앞서 구해진 SATD 값에 더하여 총 9 개의 위치 별 Rate Distortion(RD)-cost 를 구하여 RD-cost 값이 최소인 위치를 최종 best half-pel 로 선정한다. Quarter-pel search 에 대해서도 비슷한 방식으로 이전 search 단계의 best half-pel 과 그 주변 8 개 half-pel 의 정보를 모니터링 하여 uniform 한지 여부를 판단하여 best quarter-pel 을 선정한다.

Inter prediction 은 주어진 reference index 에 포함된 모든 reference frame 에 대해 각 PU partition마다 partition index 별로 해당 동작을 수행하는데 이것을 CU size 마다 반복 수행하게 되므로, 따라서 FME 도 모든 loop 안에서 반복 연산을 수행하게 되어 HM 인코딩 시간을 기준으로 전체에서 약 45%를 차지하지 않게 되어 단일 모듈로는 가장 많은 연산 량을 보이고 있다. 이로 인해 HM 인코더의 구현 시 throughput 성능을 고려하면 가장 연산이 복잡한 FME 의 고속화와 동시에 저 복잡도를 실현 하는 것이 당면한 과제로, 선행 연구들은 연산 량을 저감하기 위해 단순히 계산을 적게 하는 방법들을 적용하면서 trade-off 로 심한 화질 열화를 감수하고 있거나 빠른 연산을 추구하면서 resource 를 많이 투입하여 화질 열화는 방지하였지만 hardware cost 측면에서

열세를 보이는 등, 화질 저하와 연산 복잡도 개선 사이에서 한쪽 방향으로 치우치는 경향을 보이고 있다. 이에 본 논문에서는 화질 저하는 최소화 하면서 FME 의 연산 복잡도를 효율적으로 줄이기 위해 이전 단계의 연산 정보를 상황에 맞게 잘 이용하여 HM 에서 사용한 matrix 의 interpolation 연산이 아닌 scaler 의 interpolation 연산만 필요한 방법으로 FME 의 연산 복잡도를 대폭 간소화 하는 방법을 제시 한다.

주요어 : HEVC, Inter Prediction, Fractional Motion Estimation,
Complexity reduction, Interpolation

학 번 : 2014-21602

목 차

제 1 장 서론	11
1.1 연구의 배경 및 내용	11
1.2 선행 연구	13
1.3 논문의 구성	15
제 2 장 Inter Prediction in HM encoder	17
2.1 Merge Estimation	20
2.2 Advance Motion Vector Prediction(AMVP)	23
2.3 Motion Esitimation(ME)	25
2.3.1 Integer Motion Esitimation(IME)	28
2.3.2 Fractional Motion Esitimation(FME)	31
2.3.2.1 Interpolation	32
2.3.2.2 Half-pel and Quarter-pel Search	39
제 3 장 제안하는 HM encoder hardware 구조	42
3.1 Hardware Pipeline 구조	42
3.2 Pseudo Merge and AMVP Candidate List	43
제 4 장 Interpolation Complexity Reduction for FME	48
4.1 Computational Complexity for Interpolation	49
4.2 Hardware Complexity for Interpolation and FME	52
4.3 Interpolation 연산 간소화를 위한 실험 검토	57
4.4 Bilinear 와 Bicubic Interpolation 의 적용	66
4.4.1 Bilinear Interpolation	67
4.4.2 Bilcubic Interpolation	70
4.5 Interpolation Filter 개수 저감을 통한 연산 간소화	72

제 5 장 실험 결과.....	75
5.1 SATD 를 이용한 조건부 Bilinear Estimation	75
5.2 선행 연구와 성능 비교.....	85
제 6 장 결 론	88
참고 문헌	90
Abstract	68

표 목차

표 1. Prediction stage 에서 pseudo merge 및 AMVP candidate list 에 의한 BD-rate[%] 성능 변화	46
표 2. 4x4 블록 사이즈의 interpolation 을 위한 연산 복잡도	50
표 3. PU partition 크기 별 interpolation 연산 수행 cycle	54
표 4. 64x64 CTU 한 개 연산을 위한 FME 의 수행 cycle	56
표 5. 가정하는 4K@30fps HEVC 인코더의 specification	56
표 6. Interpolation filter 개수 저감 방법 별 BDRate 성능 변화 비교	73
표 7. Interpolation 방법 별 연산 복잡도[%]와 BDRate 저하율[%] 비교.....	74
표 8. UHD 시퀀스에서 threshold 별 bilinear SATD estimation 적용 시 성능 변화 (BDRate[%], interpolation 연산복잡도[%])	82
표 9. WQXGA 시퀀스에서 threshold 별 bilinear SATD estimation 적용 시 성능 변화 (BDRate[%], interpolation 연산복잡도[%])	82
표 10. Full-HD 시퀀스에서 threshold 별 bilinear SATD estimation 적용 시 성능 변화 (BDRate[%], interpolation 연산복잡도[%])	83
표 11. Interpolation 방법 별 IC[%]와 HC[%] 및 BDBR[%] 비교.....	86

그림 목차

그림 1. HM encoder 의 각 모듈 별 복잡도 비율	12
그림 2 Inter prediction flow chart in HM encoder	19
그림 3. Merge candidate list 의 구성	20
그림 4. Neighbor PU 의 N block	22
그림 5. Temporal merge candidate search 과정	22
그림 6. AMPV 후보 리스트의 선택 과정	25
그림 7. Inter Prediction and CU mode encoding flow in HM encoder	27
그림 8. Test zone search for integer motion estimation in HM encoder.....	29
그림 9. TZS 의 격자 탐색에서 각 탐색 차수 별 integer-pel 탐색 위치와 1pixel 거리의 integer-pel.....	30
그림 11. Interpolation 수행 시 integer 및 fractional-pel 의 위치.....	33
그림 12. integer-pel I 와 1/2 fractional-pel b 및 quarter fractional-pel a, c 의 위치.....	34
그림 13. integer-pel 을 이용한 1D interpolation.....	37
그림 14. fractional-pel 을 이용한 2D interpolation	20
그림 15. 색차 성분에 대한 integer-pel 와 fractional-pel 의 위치.....	38
그림 16. 색차 성분에 대한 interpolation	38
그림 17. HM 인코더의 fractional-pel 탐색 순서.....	40

그림 18. 가정하는 HM 인코더의 Hardware pipeline 구조..	42
그림 19. Unavailable Spatial candidates for AMVP candidate list	45
그림 20. Quarter-pel 정밀도 기준 fractional-pel 의 위치..	50
그림 21. 제안하는 HEVC 인코더의 hardware implementation 에서 interpolation unit 의 구조	53
그림 22. Interpolation 연산 간소화 방법 별 integer-pel 과 fractional-pel 의 위치	59
그림 23. 1080p 5 개 시퀀스에서 최적 integer-pel 과 그 주변 8 개 integer-pel 의 RD-cost 값 과 SAD 값의 분포 비율.....	64
그림 24. 1080p 5 개 시퀀스에서 최적 integer-pel 와 그 주변 24 개 integer-pel 들의 RD-cost 값 과 SAD 값의 분포 비율.....	69
그림 25. 최적 integer-pel 주변 분포의 특성을 반영한 2 가지 방법으로 bilinear interpolation 적용 시 BD-rate 성능 변화.....	69
그림 26. 최적 integer-pel 주변 분포의 특성을 반영한 2 가지 방법 별로 bicubic interpolation 적용 시 BD-rate 성능 변화.....	71
그림 27. Proposed FME algorithm using conditional bilinear SATD estimation with two thresholds	77

그림 28. IME 수행 후 결정 된 최적 integer-pel A0.0 와 bilinear interpolation 을 수행하기 위한 주변 8 개 integer-pels	78
그림 29. Ratioint 및 Rationhalf 에 대한 probability mass functions	81
그림 30. Integer-pel 및 half-pel 해상도에서 영상 크기에 따른 주변 pixel 간의 linearity 특성 비교	84

제 1 장 서 론

1.1 연구의 배경 및 내용

High efficiency video coding (HEVC)는 이전 국제 비디오 압축 표준인 H.264.AVC 에 비해 약 35% ~ 50% 정도의 부호화 효율을 보이는 차세대 비디오 압축 표준으로 HD 급 영상과 초고화질 UHD 및 QUHD 급 영상의 방대한 데이터를 한정된 전송 송로 데이터 폭 이내에서 효과적으로 압축하여 전송하기 위해 개발이 되었다. 현재 스마트폰 에서부터 IPTV, DTV 및 디지털 캠코더에 이르는 다양한 멀티미디어 기기들의 동영상 압축에 사용되고 있는 H.264/AVC 표준은 초고화질 동영상을 서비스 하기에는 데이터 량이 물리적인 전송 송로의 데이터 폭 한계에 직면한 상태로 ITU-T VCEG (Video Coding Expert Group)과 ISO/IEC MPEG (Moving Picture Experts Group)의 표준화 활동으로 2013 년 1 월 최종 표준안(FDIS: Final Draft International Standard)이 완성 되었다. 이후 전세계적으로 HEVC 를 이용한 위성, 인터넷 스트리밍, 케이블 및 지상파 방송 서비스 상용 기술들이 다양하게 시도 되고 있으며 초고화질 용 방송 규격에 대한 제정이 마무리 될 것으로 예상 되는 2016 년 상반기 이후에는 본격적인 HEVC 하드웨어 및 소프트웨어 솔루션이 시장에서 서비스 될 것으로 기대하고 있다.

HEVC 는 이전 표준인 H.264/AVC 대비 압축 효율을 증대시키기 위해 다양한 압축 기술을 활용하고 있는데 H.264/AVC 가 16x16 pixel 크기의 단일 계층 매크로 블록(MB: Macro block)을 코딩의 기본 단위로 사용하던 것에 비해

HEVC에서는 64x64 pixel 크기의 다 계층 코딩 트리 구조 (CTU: Coding Tree Unit)를 도입하였고, 하나의 CTU 는 다시 32x32, 16x16, 8x8 크기의 여러 코딩 유닛(CU: Coding Unit)으로 분할할 수 있도록 지원하고 있다. 화면 간 PU 에서 예측 단위 (PU: Prediction Unit)는 H.264/AVC 에서 16x16, 8x8 및 4x4 크기로 분할할 수 있도록 지원하는데 비해 HEVC 의 경우 64x64 에서부터 4x4 까지의 분할할 수 있고 asymmetric 분할까지 지원하면서 움직임의 좀 더 세밀하게 예측할 수 있도록 변경하였다. 변환 블록(TU: Transform Block)도 또한 8x8 과 4x4 크기만 지원하던데 비해 32x32 에서부터 4x4 까지 좀 더 다양한 크기를 지원하도록 변경 되었다. 이와 같이 여러 계층의 다양한 PU 및 분할 블록에 대해 지원을 하게 되면서 압축 효율은 증대하게 되었으나 당연히도 HEVC 의 인코더의 복잡도는 H.264/AVC 대비 약 5 배 정도 증가[1] 되었다.

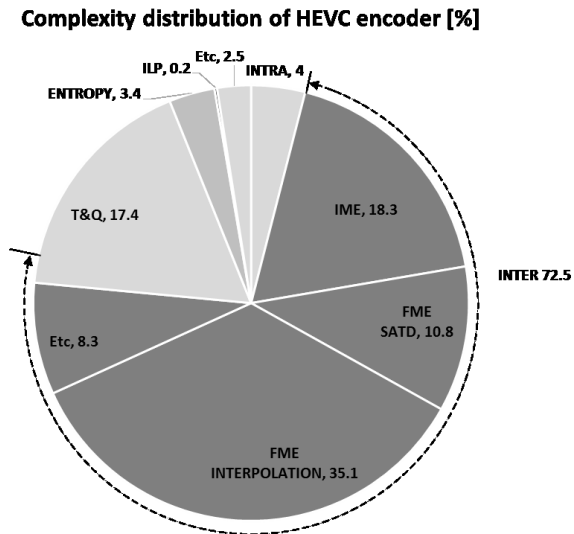


그림 1. HM encoder 의 각 모듈 별 복잡도 비율

그림 1 은 Kimono, PartkScene, Cactus, BasketballDrive, BQTerrace 5 개의 1080p sequence 에 대해 인코딩 시간을 기준으로 각 부분별 연산 복잡도를 나타내고 있다. 이 중 화면 간 예측이 약 72% 정도로 전체 인코더에서 대부분의 연산 복잡도를 차지하고 있으며 이 중에서도 fractional-pel 탐색을 위한 interpolation 연산이 약 35%로 단일 모듈로는 제일 높은 연산 복잡도를 보이고 있다. 이는 6/2 tap filter [2]를 적용한 H.264/AVC 와는 달리 좀 더 정밀한 예측을 위해 7/8 tap filter [3]를 적용하면서 기본적인 interpolation 연산량이 증가한 것에 더하여 지원되는 CU 및 PU 분할 모드도 또한 늘어나면서 발생하게 된 것으로 본 논문에서는 HEVC 인코더의 효과적인 구현을 위해 연산 복잡도가 제일 높은 화면 간 예측에서 FME 수행 시 연산 복잡도를 효과적으로 줄이는 방안에 대해 제안한다.

1.2 선행 연구

지금껏 소개된 MPEG-2, MPEG-4, VC-1, AVS, H.264/AVC 및 HEVC 와 같은 여러 가지 동영상 압축 코덱에는 각각 고유의 interpolation 방법 [4]이 적용되어 있다. 앞서 언급한 바와 같이 HEVC 에서는 이전 표준인 H.264/AVC 가 6/2 탭 필터를 사용하던 것에 비해 7/8 탭 필터를 채용함으로써 인해 interpolation 된 영상으로 부터 motion estimation 시 정밀도는 향상 되어 좀 더 향상된 성능을 기대할 수 있게 되었으나 부가적으로 연산량도 또한 증가하게 되었다.

굳이 이전 표준들과의 비교를 하지 않더라도 interpolation 은 일반적으로 각각의 동영상 압축 코덱에서 연산량을 제일 많이 필요로 하는 과정으로 많은 연구자들이 interpolation 의 연산 복잡도를 낮추기 위해 다양한 시도들이 있었다. 우선, 단순히 화면 간 예측 수행 시 PU 의 인위적인 제한을 통해 해당 예측 분할 모드에서는 아예 motion estimation 과정을 수행하지 않음으로써 interpolation 의 연산을 단순화 하는 방법 [5]이 소개 되었는데 PU 를 8×8 로 제한 하여 허용된 모드 이외에는 motion estimation 을 수행하지 않음으로써 interpolation 연산을 줄임과 동시에 전체 연산 복잡도를 줄이는 시도가 있었다. 그러나 이는 허용된 크기의 PU 만 연산을 수행함으로써 인해 throughput 향상을 기대하는 방법으로 허용된 모드보다 더 큰 크기의 유닛을 처리하기 위해서는 제시된 throughput 보다 성능이 저하 될 수 밖에 없는 단점이 있다. 이와는 달리 interpolation 필터의 탭 수를 감소 조절하여 pixel 단위 interpolation 연산을 비교적 간소화 하는 방법들도 제안 되었는데 $7/8$ tap 을 적용하는 대신에 이전 표준인 H.264/AVC 의 $6/2$ 탭을 HEVC 에 적용 [6],[7] 하여 연산량을 다소 감소 시키는 방법도 연구되었고, 복잡도는 감소 시키되 화질 열화는 최소화 하기 위해 H.264 와 HEVC 의 interpolation filter 를 적절히 조합하여 half-pel 에 대해서는 8 tap filter 를 사용하고 quarter-pel 에 대해서만 2 tap filter 를 제시한 연구 [8]도 있다. 또한 [9],[10]에서는 pixel 의 휘도 값을 이용하여 fractional-pel 을 생성하는 일반적인 interpolation 방식이 아닌 RD-cost 값을 이용하는 방식을 제안하였는데 각 fractional-pel 위치의 RD-cost 값을 계산하여 pixel-interpolation 이 아닌 cost-interpolation

방법을 제시하고 있다. integer-pel 탐색 후 구해진 최적의 integer-pel 위치에서 논문에서 제안한 2 차원 방정식을 이용하여 fractional-pel 위치의 RD-cost 값을 계산한 후 각각의 RD-cost 값을 비교하여 최적의 fractional-pel 의 위치를 선택하는 방식으로 기존의 pixel 단위 interpolation 이 복잡 다단한 matrix 연산이 필요한 대신 제안 방식으로는 단순한 scaler 연산만을 수행함으로 인해 복잡도를 획기적으로 낮출 수 있는 장점이 있다. 하지만, integer 화소 레벨에서 quarter 화소 레벨로 급격히 approximation 수행하게 되면서 코딩 효율의 저하가 심하다는 단점을 극복하기 어렵다.

선행 연구들이 다양한 방법을 이용하여 interpolation 의 연산량을 간소화 하기 위해 노력하였는데 본 논문에서는 이러한 연장 선상 에서 선행 연구들에서 제시한 방안들과 그 외 [11],[12]에서 소개한 bicubic interpolation 에서 사용한 4 tap filter 를 적용하는 방안에 대해서도 고려하여 화질 열화는 최소화 하면서 interpolation 의 연산량을 최대한 간소화 하기 위한 최적의 방안을 제시하였다.

1.3 논문의 구성

본 논문은 우선 2 장에서 HM 인코더에서 Inter prediction 의 전체 구조가 어떻게 구성 되어있는지 우선 개략적으로 설명한 후 Inter prediction 내 각 부분 별 동작이 어떻게 수행 되는지에 대해 다음과 같은 순서로 상세히 설명하였다. 먼저 Merge estimation 과 Advanced motion vector prediction (AMVP)의 수행 동작을 비교 하였고, AMVP candidate list 에서 선택된 best candidate 에 대해

motion estimation 을 수행하면서 integer motion estimation(IME)과 fractional motion estimation(FME)이 어떠한 동작으로 연산을 수행하는지 설명한 후 3 장에서 제안하는 HM 인코더의 하드웨어 구조에서 inter prediction 부분에 대한 가정을 기술 하였다. 4 장에서는 interpolation 과 FME 의 연산복잡도와 hardware 복잡도를 분석한 뒤 interpolation 연산을 간소화 하는 서로 다른 방안들과 각 방안 별 실험 결과를 정리하였다. 5 장에서 앞장의 실험을 바탕으로 설계한 제안 알고리즘의 설명과 이에 대한 실험 결과 및 선행 연구들과의 실험 결과를 비교한 후 6 장에서 결론을 정리 하였다.

제 2 장 Inter Prediction in HM encoder

HEVC 의 화면 간 예측 (Inter Prediction)은 비디오 스트림에서 시간적으로 인접한 picture 들 간 동일 위치의 블록들은 서로 높은 상관성을 갖고 있다는 사실을 바탕으로 이 정보를 효과적으로 이용하여 코딩 하려고 하는 현재 블록에 대한 움직임 예측하는 과정을 말한다. 이미 코딩 된 picture 들의 정보를 저장하고 있는 reference picture list 에서 현재 picture 와 reference picture 간의 POC 의 차이가 가장 적은 picture 를 이용하는데 POC 차이가 가장 적은 reference picture 에는 현재 블록과 확률적으로 가장 유사한 움직임 정보를 보유한 블록이 있다는 가정하에 reference picture 를 선택하고 이를 이용하여 현재 블록에 대해 motion estimation 을 수행하여 하나의 예측 블록을 생성하는 일련의 과정을 수행한다. 그 결과로 발생하는 두 블록간의 차분 신호를 양자화를 통해 부호화하고 두 블록간의 상대적인 위치 정보를 움직임 벡터로 표현하여 이 값을 또한 부호화하여 디코더로 전송한다.

움직임을 예측하는 과정에서 다양한 방법을 통해 좀 더 정밀한 예측 블록을 만들게 되면 부호화 할 차분 신호의 데이터 량을 최소화 할 수 있는데, 적은 데이터 량은 전송 송로에 부담을 줄일 수 있는 것과 동시에 에너지를 적게 사용한다는 의미와 같기 때문에 최근의 동영상 압축 코덱 에서는 좀 더 정확한 예측 블록을 생성하기 위해 움직임 벡터의 해상도를 높여 integer-pel 뿐만 아니라 half pixel 및 quarter-pel 단위의 fractional-pel 정밀도로 ME 를 수행하고 있다.

Inter prediction 에서 하나의 slice 는 P-slice(predictive slice)와 B-slice(bi-predictive slice)로 구분할 수 있는데 P-slice 는 하나의 prediction unit(PU)는 한 개의 reference picture list 에서 reference picture 를 선택하는 단방향 예측 (Uni-directional prediction)만을 수행하고 B-slice 인 경우에 PU 는 두 개의 reference picture list 의 정보를 이용하는 양방향 예측(Bi-directional prediction)과 단방향 예측을 선택적으로 이용할 수 있다.

HEVC 는 입력 동영상을 부호화 할 때 하나의 picture 에서 블록을 64x64 크기의 CTU 단위로 분할하고 이를 다시 64x64, 32x32, 16x16, 그리고 8x8 의 CU 단위로 분할하는 quad-tree 구조를 채용하고 있다. Inter prediction 에서 CU 단위로 분할된 블록은 다시 prediction unit(PU)로 분할 되는데 CU 가 블록에 대한 코딩의 기본 단위라 한다면 PU 는 예측의 기본 단위로 단일 계층 다수의 분할 모드로 구성 되어 있다. 즉, inter prediction 에서 PU 는 $2N \times 2N$ 의 CU 모드로부터 7 개의 모드, 즉 $2N \times 2N$, $2N \times N$, $N \times 2N$, $2N \times nU$, $2N \times nD$, $nL \times 2N$, 및 $nR \times 2N$ 로 분할이 가능하다.

그림 2 는 HM 인코더에서 inter prediction 의 전체 수행 동작의 흐름을 보여준다. 각 CU 분할 모드 별로 Inter prediction 이 수행 되면 각 PU 분할 모드 별로 skip, merge, 그리고 AMVP 모드로 예측 블록의 움직임 벡터를 예측하는 동작을 수행하는데, 우선 skip 모드는 현재 블록의 크기가 $2N \times 2N$ 일때 현재 블록의 움직임 정보와 동일한 움직임 정보를 갖는 후보 (candidate)가 차분 신호가 발생하지 않는 경우 AMVP 를 수행 하지 않고 inter prediction 동작을 종료하여 $2N \times 2N$ 에 대해 빠른 동작 수행을

시도한다. CU $2N \times 2N$ 이 전체 분할 모드에서 평균적으로 약 80%정도를 차지하고 있으므로 $2N \times 2N$ 에 대해서만 AMVP 를 위한 연산을 수행하지 않는 것만으로도 빠른 동작을 수행하는데 도움이 된다고 할 수 있다. Merge 모드의 경우 현재 블록의 움직임 정보와 동일한 움직임 정보를 갖고 있는 시·공간적으로 이웃한 블록의 정보를 이용하여 현재 블록의 예측 블록을 비교적 간단하게 생성하는 동작을 수행한다. AMVP 모드는 앞서 설명한 skip 과 merge 모드에 해당되지 않는 경우 현재 블록의 움직임 벡터와 가장 유사한 움직임 벡터를 갖는 AMVP 후보자를 선택하여 ME 와 motion compensation(MC)과 같은 복잡한 예측 연산을 수행하여 현재 블록의 예측 블록을 생성하게 된다.

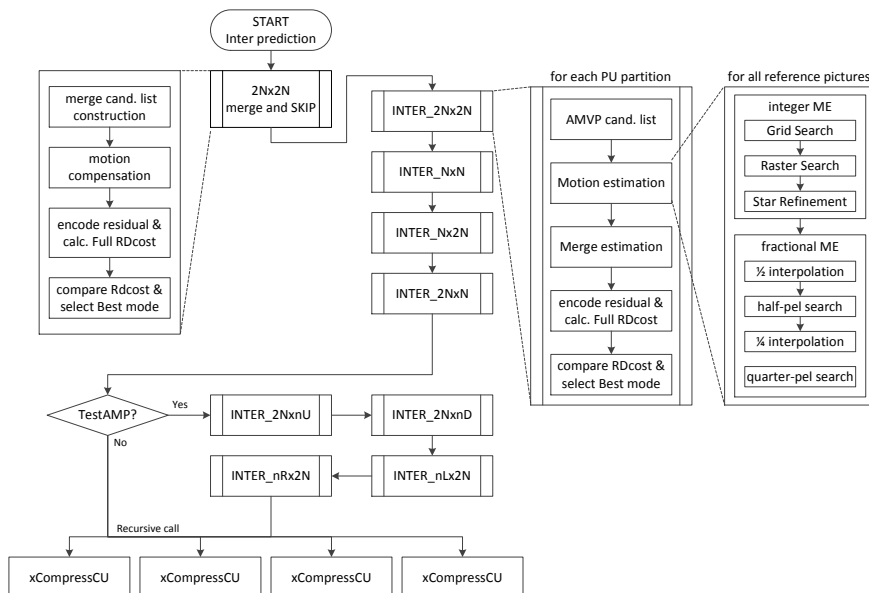


그림 2 Inter prediction flow chart in HM encoder

그림 2 에서와 같이 Inter prediction 에서 skip 모드는 $2N \times 2N$ 에 대해서만 수행하지만, merge 와 AMVP 의 모드는 각 PU

모드 별로 모두 수행하는 것과 동시에 모든 reference picture 별로 수행하게 되어 그림 1 에서와 같이 전체 HM 인코더의 복잡도에서 약 72%가량의 복잡도를 차지하는 원인이 되었고 이에 따라 inter prediction 을 좀 더 효율적으로 수행하도록 최적화에 대한 필요성이 대두 되었다.

2.1 Merge Estimation

HM 에서 Merge estimation 과정은 이미 prediction 을 수행한 인접 block 을 탐색하여 merge candidate 를 구성한 뒤 MV 를 구하고, SATD based Low complexity RD cost 를 이용하여 cost 비교를 통해 merge block 을 선택한 후 해당 정보를 merge index 로써 signaling 한다.

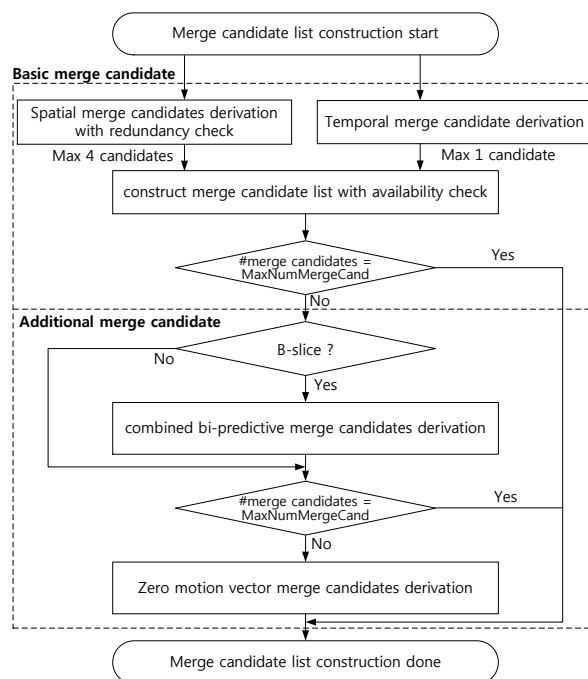


그림 3. Merge candidate list 의 구성

Maximum candidate list 의 수는 slice header 에 정의 된 five_minus_max_num_merge_cand 의 syntax element(SE)를 통해 받으며 1~5 까지의 range 가 있는데, 기본적으로 HM 의 candidate list 에는 5 개를 사용할 수 있게 적용되어 있다. 이 중 spatial candidate 는 최대 4 개, temporal candidate 는 1 개를 사용하되 candidate list 가 다 채워지면 이후에는 candidate 를 추가하는 동작을 수행 하지 않는다. 따라서 merge candidate list 를 구성과정을 크게 두 가지로 구분할 수 있는데, 첫 번째로 spatial & temporal merge 의 Basic merge candidate 및 두 번째로 combined bi-predictive & zero motion vector merge 의 additional merge candidate 로 분류할 수 있으며 이는 그림 3 과 같다.

Basic merge candidate 구성 시 각각의 spatial & temporal merge candidate 는 availableFlag 를 확인하여 available 한 경우에 대해 list 구성이 되는데 merge 는 현재 block 이 인접 block 들과 motion 정보가 유사하다는 가정에서 출발하므로, merge candidate list 를 구성할 때 PU 의 분할 모양에 따라 공간적으로 인접한 주변 블록들을 우선적으로 merge candidate list 에 추가한다.

그림 4 와 같이 현재 PU 의 주변에 위치한 다섯 개의 block 을 A1, B1, B0, A0, B2 의 순서로 서로간 redundancy 와 unavailability 를 확인하면서 탐색하여 최대 4 개까지 merge candidate list 에 추가한다.

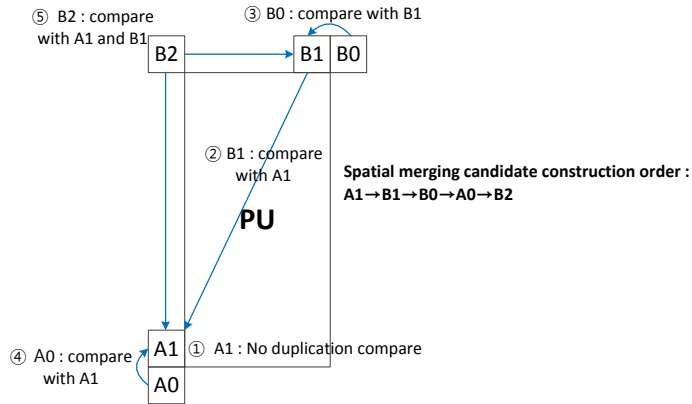


그림 4. Neighbor PU 의 N block

만약, 공간적으로 인접한 블록에서 후보자를 다 채우지 못할 경우 시간적으로 인접한 블록에서 후보자를 더 찾게 된다. temporal merge candidate list 구성 과정이 진행은 그림 5 와 같이 collocated PU 는 선택된 reference picture 에서 현재 PU 에 대응 되는 위치에 존재하는 PU 를 의미하며 예측 성능을 이유로 H 의 block 를 우선 탐색 후 motion 정보가 없다면 C1 을 다음 candidate 로 고려하고 있다.

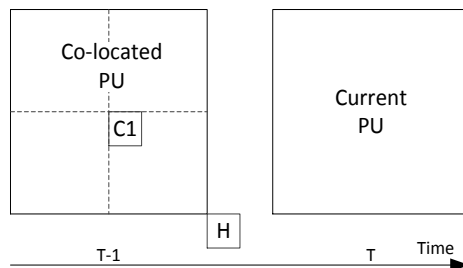


그림 5. Temporal merge candidate search 과정

시간적, 공간적으로 인접한 블록에서 후보자를 찾아도 최대 후보자 개수 5 개를 채우지 못할 경우가 발생하면 일종의 가상

후보자를 생성하게 되는데 이미 후보 리스트에 추가되어 있는 후보자들의 조합으로 나머지 빈 공간의 후보 리스트를 채우게 되며 이를 combined bi-predictive candidate 라 부른다. 만약, 이 경우에도 후보 리스트를 채우지 못하는 경우가 발생하게 되면 zero 움직임 벡터를 비어 있는 후보 리스트에 채우게 된다. 병합 후보 리스트 생성 과정에서 각 단계별 후보 탐색을 수행하면서 최대 후보 개수인 5 개 도달하게 되면 나머지 탐색 과정은 취소되고 candidate list 구성 과정은 종료하게 되며 이 후, 이들 중 가장 낮은 RD-cost 값을 갖고 있는 후보가 best merge candidate 로써 선택이 된다.

2.2 Advanced Motion Vector Prediction (AMVP)

AMVP 수행의 기본적인 동작은 앞서 기술한 merge estimation 과 유사하다. 즉, 부호화 하고자 하는 현재 블록의 공간적으로 이웃한 블록과 시간적으로 인접한 블록으로부터 움직임 벡터의 후보자를 탐색하여 후보 리스트에 추가하고 각 후보들끼리 RD-cost 을 비교하여 최적의 후보를 선택하는 일련의 과정을 수행한다. 다만, merge estimation 이 움직임 벡터, 참조 picture 의 번호, 단 방향 혹은 양 방향 PU 를 최적의 merge 후보로부터 그대로 가져와 사용하여 부가적인 연산이 필요가 없는 대신 AMVP 에서는 선정된 최종 후보의 위치를 초기 위치 값으로 하여 motion estimation 을 수행한 후 이를 통해 최적의 움직임 벡터를 찾는 과정을 추가로 수행하게 되어 merge estimation 과 비교하여 좀 더 많은 연산을 필요로 한다. AMVP 시 공간적 후보의 위치는 merge estimation 시와 동일한 위치에서 찾게 되며 그림 6 와 같이 이중

2 개를 후보 리스트에 추가하게 되는데, 표준에서는 이때 반드시 블록의 왼쪽 이웃에 위치한 후보들에서 한 개를, 나머지 한 개는 블록의 위쪽에 위치한 후보들 중에서 한 개를 추가하게 된다. 만약 현재의 PU 와 이웃 블록이 같은 reference picture list 를 사용하지만 다른 reference picture 를 참조하고 있거나, reference picture list 도 다르고 다른 reference picture 를 참조하고 있게 되면 각 후보를 스케일링 하여 후보리스트에 추가하게 된다. 따라서 블록의 움직임 연관성 측면에서 보았을 때 스케일링을 사용하지 않는 후보가 우선 순위에 있게 되는데 이는 예측 과정에서 일반적으로 공간적인 후보를 이용 한다는 것은 기본적으로 현재 PU 와 이웃 PU 의 움직임 정보가 유사할 것이라는 가정에 기초를 두고 있기 때문에 가장 움직임 정보가 유사하다고 생각되는 공간적 후보에 후보 리스트 선정 시 우선 순위를 높이는 것으로 생각할 수 있다. AMVP 후보 리스트에 최대 2 개까지 채워지기 전 까지 $A0 \rightarrow A1 \rightarrow \text{scaled } A0 \rightarrow \text{scaled } A1 \rightarrow B0 \rightarrow B1 \rightarrow B2 \rightarrow \text{A/B side redundancy check} \rightarrow \text{scaled } B0 \rightarrow \text{scaled } B1 \rightarrow \text{scaled } B2 \rightarrow \text{temporal H} \rightarrow \text{temporal C} \rightarrow \text{zero vector}$ 의 순서로 추가 후 merge estimation 과 동일하게 후보들 간 RD-cost 비용 비교를 통해 최적의 AMVP candidate 를 선정한다.

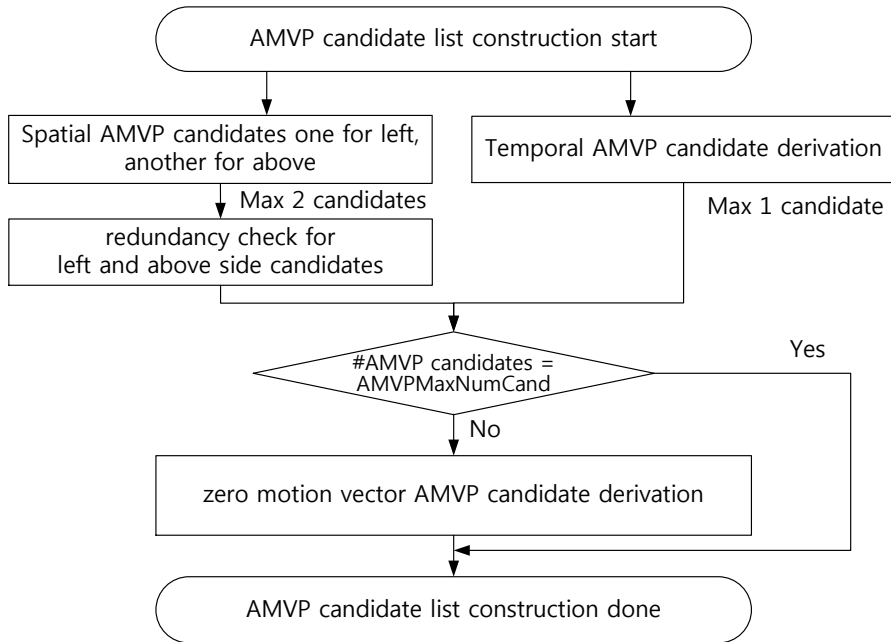
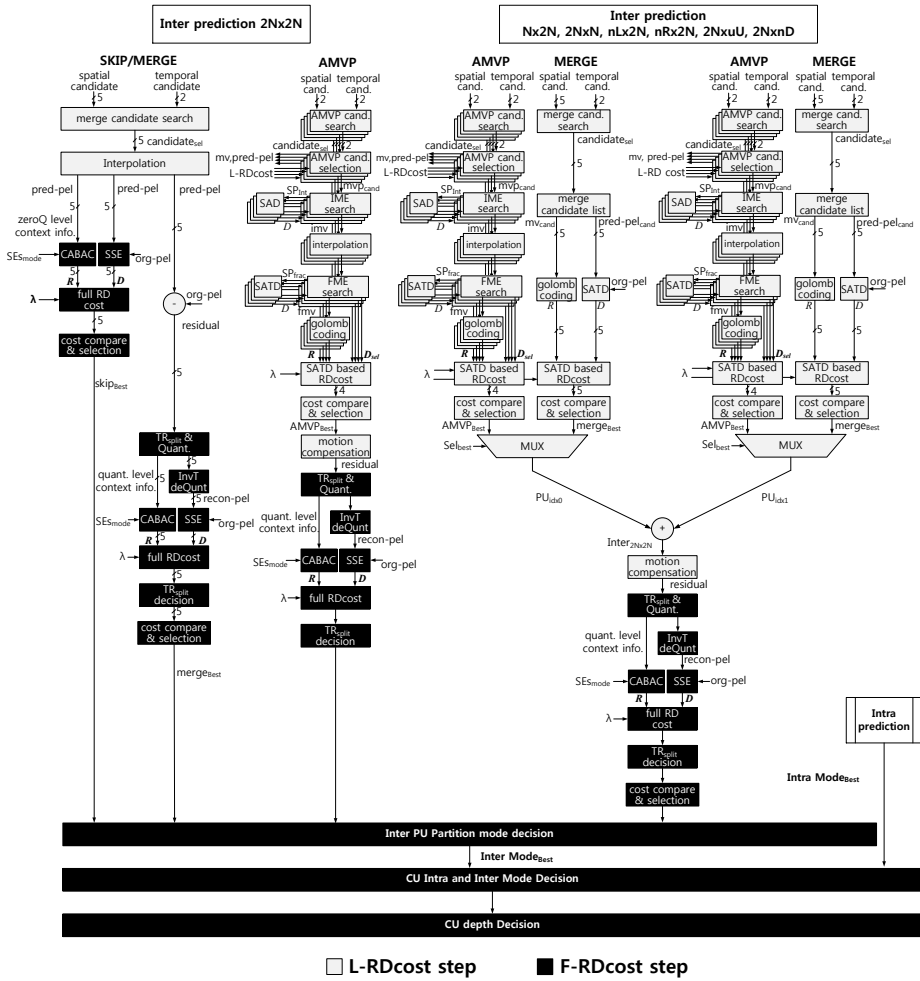


그림 6. AMPV 후보 리스트의 선택 과정

2.3 Motion Estimation (ME)

그림 7 의 CU mode 결정까지의 Inter Prediction 수행 과정을 보면 AMVP 와 merge estimation 을 통해 선택된 각 best 후보가 RD-cost 값을 기준으로 경쟁을 하여 해당 PU 의 최적 mode 로 결정이 된다. AMVP 후보 리스트에서 선택이 된 후보들 중 RD-cost 값이 가장 작은 움직임 벡터 후보를 AMVP 의 최적 후보로 선택한 후 motion estimation 의 초기 위치로 지정 한다. 이 후 Uni-prediction 인 경우 integer-pel 에 대한 움직임을 test zone Search(TZS) [13] 방법을 이용하고, Bi-prediction 인 경우 full search 의 방법을 통해 최적의 integer-pel 위치를 선택한 후 해당 integer-pel 위치를 기준으로 하여 fractional-pel 의 움직임을

추정하게 된다. HEVC에서는 quarter-pel 정밀도의 fractional-pel 탐색을 지원하고 있으므로, 탭 필터링 interpolation 을 통해 8 개의 half-pel 의 위치를 찾은 다음 각 half-pel 후보들에 대한 RD-cost 비교를 통해 최적의 half-pel 위치를 선택 한다. 그리고 나서 최적 half-pel 위치에서 또다시 interpolation 을 통해 quarter-pel 에 대한 8 개의 후보 위치를 찾은 후 동일한 비용 비교 방법을 통해 최적의 quarter-pel 을 선택하게 된다. AMVP 과정을 거쳐서 선택 된 최적의 예측 pixel 과는 별개로 merge estimation 은 독립적으로 수행되며 merge estimation 에 의해서도 최적의 예측 pixel 이 구해지는데 각각의 과정에서 구해진 두 개의 최적 예측 pixel 들은 다시 한번 RD-cost 비교를 통해 현재 CU 내에서 최적의 PU 모드를 선택하게 된다. 이 후 각각의 최적 PU 들의 조합으로 이루어진 CU 들끼리의 RD-cost 비교를 통해 최적의 CU 모드를 선택하고 CU 사이즈 간 비교를 통해 최적의 CU 크기도 결정하게 된다.



- R: rate
- D: distortion
- SE: syntax element
- SP: search point
- int: integer-pel
- frac: fractional-pel (half, quarter)
- pred-pel: prediction pel
- org-pel: original-pel
- recon-pel: reconstruction-pel

그림 7. Inter Prediction and CU mode encoding flow in HM encoder

2.3.1 Integer Motion Estimation (IME)

HM은 uni-prediction인 경우 integer-pel의 탐색을 위해 TZS 알고리즘을 채택하고 있다. TZS 알고리즘은 정해진 탐색 영역의 모든 integer-pel 위치에 대하여 탐색을 하는 대신 그림 8 과 같이 마름모 탐색 (Diamond search) 혹은 격자 탐색(Grid search)로 불리는 탐색 방법으로 탐색을 우선 시작한다. 모든 참조 picture 별로 모든 PU 에 대하여 정해진 탐색 영역 내에서 integer-pel 위치를 전부 탐색을 하는 full search 를 사용하게 될 경우 탐색 결과에 대한 성능은 제일 뛰어나겠지만 화면 크기와 탐색의 범위가 커지면 커질수록 연산량이 매우 커지게 되는 단점을 피할 수 없다. HEVC 가 UHD 및 QUHD 영상의 압축을 목적으로 개발된 표준임을 상기하였을 때 고화질 영상 사이즈에 따른 막대한 연산량은 부담일 수 밖에 없으며 이러한 연산량을 줄이기 위해 채택된 고속 motion estimation 알고리즘이 TZS 라고 할 수 있다. 격자 탐색은 탐색 차수 별로 4 개 혹은 8 개의 탐색 위치에 대해서만 최적 integer-pel 위치를 찾는 방법으로 최초 3 차수를 탐색하여 초기 위치 대비 최적 위치가 갱신되지 않으면 탐색을 종료하되 그렇지 않고 3 차수를 탐색하는 동안 최적 위치가 갱신이 되었다면 다시 3 차수를 탐색하게 된다. 이 과정을 주어진 64x64 pixel 의 탐색 영역 안에서 수행하게 되는데 만약, 최적으로 판단된 integer-pel 의 위치가 초기 위치로부터 5 pixel 이상 떨어져 있는 위치에 존재하고 있다면 이것은 움직임의 크기가 커서 초기 탐색을 시작한 위치보다 멀리 있는 영역에서 최적의 움직임 벡터가 존재할 가능성이 크다고 생각할 수 있다. 따라서 이때는 격자 탐색 방법을 사용하여 integer-pel 후보들을 성글게 탐색한 결과로는 정확한

움직임 벡터를 찾지 못했을 확률이 크다고 판단하여 탐색 영역에 대해 전역 탐색(Raster Search) 방법으로 균등하게 샘플링 하여 최소 비용의 위치를 판단한 다음 그 위치를 다시 탐색의 시작점으로 하여 다시 격자 탐색을 하는 Star refinement 를 수행하게 된다. Star refinement 시에는 이전의 초기의 격자 탐색 방법과는 다르게 최적 위치로 찾아진 위치의 탐색 거리가 0 이 될 때까지 반복하는 과정을 수행한다.

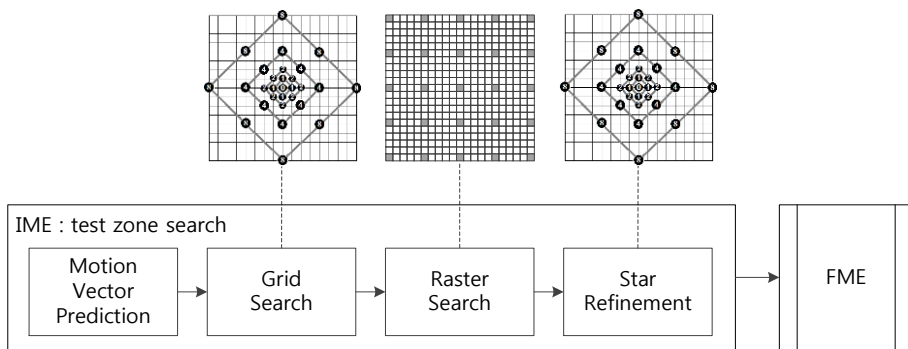


그림 8. Test zone search for integer motion estimation in HM encoder

이 세가지 탐색 방법을 통해 최적의 integer-pel 위치를 선택하게 되는데 이러한 격자(Grid) 탐색, 샘플링 전역(Raster) 탐색 및 격자 (Star refinement) 탐색으로 이어지는 일련의 integer-pel 탐색 과정을 TZS 알고리즘이라 한다.

TZS 의 첫 번째 search 방법인 격자 탐색의 마지막 단계에서, 찾아진 최적 integer-pel 의 거리가 초기 위치로부터 0 인지 혹은 1 인지 체크하는 부분이 있다. 이는 그 값이 0 이면 정확한 움직임 벡터를 찾았음을 의미하므로 TZS 과정을 종료함을 의미하고 그

값이 1이면 그림 9의 1 차수 혹은 2차수 중 거리가 1인 integer-pel 위치 중 하나를 최적 위치로 찾았다는 의미인데, 여기에서는 지정된 격자 위치에 대해서만 탐색이 진행되었기 때문에, 나머지 탐색되지 않은 integer-pel 위치 중 인접한 pixel 들에 대해서도 연산을 수행하여 더 최적인 pixel 들이 있는지 찾겠다는 의도 이다.

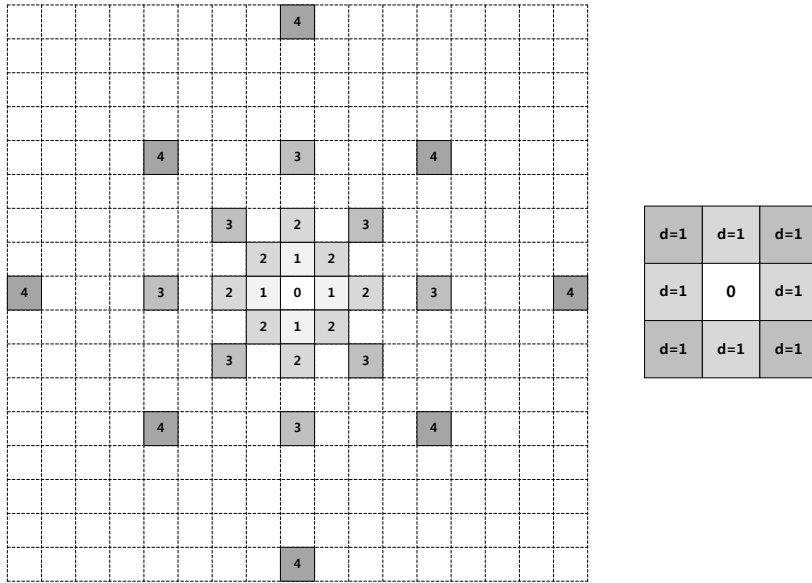


그림 9. TZS 의 격자 탐색에서 각 탐색 차수 별 integer-pel 탐색 위치와 1pixel 거리의 integer-pel

예를 들어, 위의 그림 9 에서 0 위치가 최적으로 결정되었다면 IME 를 종료하게 된다. 만약, 최적 integer-pel 의 위치가 초기 위치 대비 거리가 1 인 위치에서 결정되었다면 위 그림 9 의 오른쪽과 같이 주변의 pixel 들이 해당 된다. 즉, 최적 integer-pel 의 위치가 초기 위치 대비 1 의 거리를 갖는 다면 격자 탐색의 방법의 측면에서 이미 찾은 integer-pel 위치보다 더 최적인 위치의 integer-pel 인접해 있을 확률이 큰데 이 위치에 대한

탐색을 누락할 가능성이 있으므로 이를 보완하여 찾는 것으로 각각의 경우 별로 2-point 탐색의 위치는 아래 그림 10 과 같다. 즉, 다시 말하면 탐색 차수 1 의 4 개 pixel 와 차수 2 의 거리가 1 pixel 인 4 개 pixel 의 위치가 최적의 integer-pel 로 선택이 될 경우 잠재적으로 더 최적일 확률이 큰 인접 pixel 들을 좀 더 탐색해 보는 알고리즘이 2-point 탐색이다.

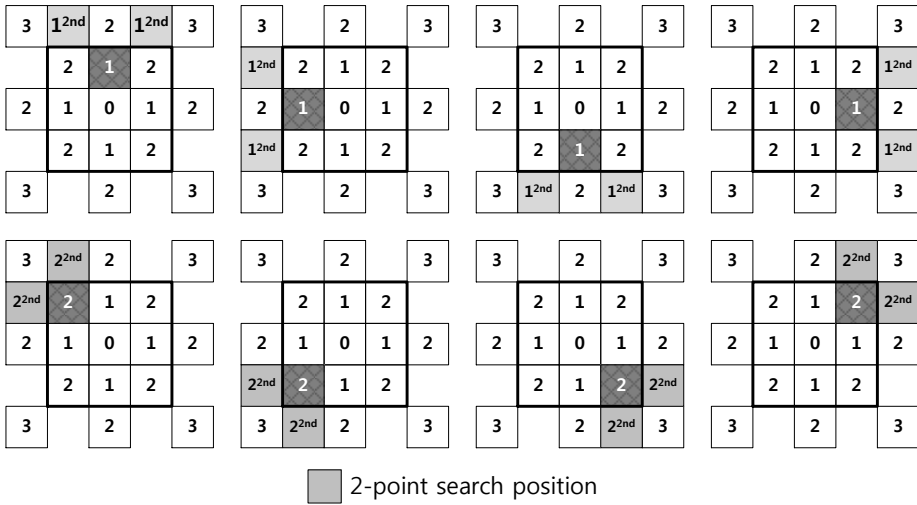


그림 10. 2-point 탐색 시 탐색 pixel 들의 위치

2.3.2 Fractional Motion Estimation (FME)

FME 는 IME 에서 선택한 최적의 integer motion vector 를 기준으로 좀 더 세밀하게 sub-pixel 단위의 MV 를 구하는 과정으로, 세부적으로는 half & quarter-pel 의 위치 생성을 위한 interpolation, 최적 fractional-pel 탐색과 이를 위한 RD-cost 함수 계산을 위한 hadamard transform 연산으로 구성된다. motion estimation 의 과정은 얼마나 정밀하게 움직임 을 잘 추정 하였는지가 인코더의 성능에 절대적인 영향을 미치게 되므로

fractional-pel 의 정확도를 높이는 것이 부호화의 효율 향상에 결정적인 역할을 한다고 할 수 있다.

2.3.2.1 Interpolation

Image processing 에서 영상의 interpolation 이란 일반적으로 복원 픽처에 실제 존재하는 integer-pel 을 기준으로 하여 interpolation 필터 연산을 통해 실제 존재하지 않는 가상의 fractional-pel 을 만들어 내는 과정을 일컫는다. HEVC 에서는 integer-pel 로부터 half-pel 및 quarter-pel 값을 생성하여 fractional-pel motion estimation 을 수행하는데 fractional-pel 값을 생성하기 위해 HEVC 에서는 DCT 기반의 interpolation 필터(DCT based interpolation filter, DCT-IF)를 이용한다. 이렇게 interpolation 필터를 이용하여 fractional-pel 을 생성하는 방법에는 선형(linear) interpolation 법, Bi-linear(bi-linear) interpolation 법, 입방(cubic) interpolation 법, Bi-cubic(bi-cubic) interpolation 법 및 위너(Wiener) interpolation 법 등 다양한 interpolation 방법이 존재하는데 이전 표준인 H.264/AVC 가 위너 interpolation 법을 이용한 6 탭 필터링 interpolation 을 이용했었다 한다면, HEVC 에서는 초기 표준화 논의 시 1/8 pixel 정밀도의 SIFO(Single pass switched interpolation filters with offsets) 12 탭 필터링을 고려 하였다가 복잡도와 부호화 성능을 고려하여 최종적으로는 quarter-pel 정밀도를 기준으로 휘도 성분에 대해서는 7/8 탭 필터링을, 색차 성분에는 4 탭 필터링 interpolation 을 적용하게 되었다.

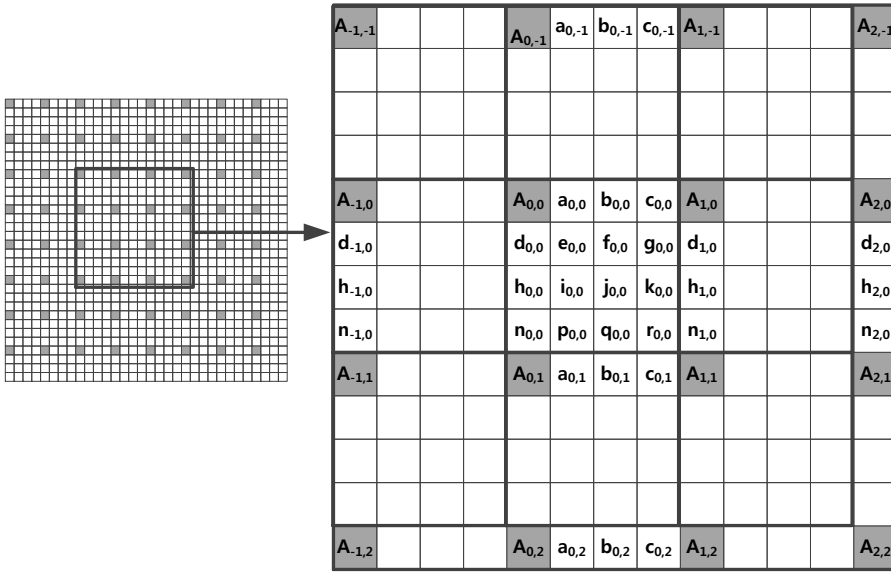


그림 11. Interpolation 수행 시 integer 및 fractional-pel 의 위치

그림 11.의 대문자 A 는 각각의 integer-pel 위치를 의미하고, 소문자는 fractional-pel 의 위치를 나타내는데 b, h, j 의 경우 half 단위 pixel 이고 a, c, d 등은 quarter 단위 pixel 의 위치를 나타낸다. integer-pel 와 같은 행렬을 공유하는 a, b, c 와 d, h, n 의 경우에는 1 차원 (1-dimension, 1D) interpolation 필터를 이용하여 interpolation 할 수 있으나 그 외 fractional-pel 들은 1 차원 interpolation 을 통해 얻어진 fractional-pel 의 값을 이용해 2 차원 (2-dimension, 2D) interpolation 필터를 이용하여 interpolation 을 수행하는데 여기서, 2 차원의 의미는 세로방향 1D interpolation 필터를 수행하여 나온 interpolation 값을 통해 다시 가로방향 1D interpolation 필터를 이용하여 interpolation 값을 얻기 때문에 2D 라고 지칭하고 있다.

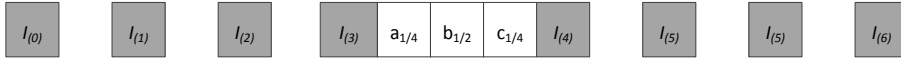


그림 12. integer-pel I 와 1/2 fractional-pel b 및 quarter fractional-pel a, c 의 위치

위의 그림 12 에서 half-pel 인 $b_{1/2}$ 의 위치를 interpolation 하기 위해서는 앞서 설명한 바와 같이 8 탭 필터링을 사용하므로 아래 식 (1)과 같이 표현할 수 있다.

$$b_{1/2} = k_0 \cdot I_{(0)} + k_1 \cdot I_{(1)} + k_2 \cdot I_{(2)} + k_3 \cdot I_{(3)} + k_4 \cdot I_{(4)} + k_5 \cdot I_{(5)} + k_6 \cdot I_{(6)} + k_7 \cdot I_{(7)} \quad (1)$$

여기서, $I_{(0)} \sim I_{(7)}$ 는 각각의 integer-pel 의 휘도 값이고 $k_0 \sim k_7$ 은 각 integer-pel 에 대응 하는 가중 값으로 integer 의 값을 갖는 interpolation 필터 계수이다. 각각의 interpolation 필터 계수들은 DCT(discrete cosine transform) 수식 (2) 와 IDCT(Inverse cosine transform) 수식 (3)을 통하여 유도할 수 있다.

$$F(k) = \alpha(k) \sum_{n=0}^{N-1} I(n) \cos\left(\frac{(2n+1)k\pi}{2N}\right) \quad (2)$$

$$I(x) = \sum_{m=0}^{N-1} \alpha(m) F(m) \cos\left(\frac{(2x+1)m\pi}{2N}\right) \quad (3)$$

$$\alpha(0) = \frac{1}{\sqrt{N}}, \alpha(k) = \sqrt{\frac{2}{N}} \quad (4)$$

식(3)은 식(1)의 integer 위치 정보를 이용하여 아래 식(5)와 같이 정리할 수 있다.

$$\begin{aligned}
 I(x) = & \alpha(0)F(0)\cos\left(\frac{(2x+1)0\pi}{2N}\right) + \alpha(1)F(1)\cos\left(\frac{(2x+1)1\pi}{2N}\right) \\
 & + \dots + \\
 & \alpha(N-1)F(N-1)\cos\left(\frac{(2x+1)(N-1)\pi}{2N}\right)
 \end{aligned} \tag{5}$$

식(5)에 DCT 함수 식(2)를 각각의 integer-pel 위치별로 대입하면 아래 식(6)과 같이 정리할 수 있다.

$$\begin{aligned}
 I(x) = & \left[\alpha^2(0) \left\{ \begin{aligned} & I(0)\cos\left(\frac{(2\cdot 0+1)0\pi}{2N}\right) + \dots + \\ & I(N-1)\cos\left(\frac{(2\cdot (N-1)+1)0\pi}{2N}\right) \end{aligned} \right\} \cos\left(\frac{(2x+1)0\pi}{2N}\right) \right] + \\
 & \left[\alpha^2(1) \left\{ \begin{aligned} & I(1)\cos\left(\frac{(2\cdot 1+1)1\pi}{2N}\right) + \dots + \\ & I(N-1)\cos\left(\frac{(2\cdot (N-1)+1)1\pi}{2N}\right) \end{aligned} \right\} \cos\left(\frac{(2x+1)1\pi}{2N}\right) \right] \\
 & + \dots + \\
 & \left[\alpha^2(N-1) \left\{ \begin{aligned} & I(N-1)\cos\left(\frac{(2\cdot (N-1)+1)(N-1)\pi}{2N}\right) \\ & + \dots + \\ & I(N-1)\cos\left(\frac{(2\cdot (N-1)+1)(N-1)\pi}{2N}\right) \end{aligned} \right\} \cos\left(\frac{(2x+1)(N-1)\pi}{2N}\right) \right]
 \end{aligned} \tag{6}$$

위의 식 (6)을 IDCT 함수 I(x)를 중심으로 정리하면 아래 식(7)과 같이 integer-pel 과 유도된 계수의 곱으로 표현할 수 있다.

$$I(u) = I(0) \cdot c_0(u) + I(1) \cdot c_1(u) + \dots + I(N-1) c_{N-1}(u)$$

$$c_x(y) = \sum_{u=0}^{N-1} \alpha^2(u) \cos\left(\frac{(2x+1)u\pi}{2N}\right) \cos\left(\frac{(2y+1)u\pi}{2N}\right) \quad (7)$$

식 (7)에서 x 는 위상을 의미하고, y 는 interpolation 필터 계수의 인덱스를 의미한다. HEVC 에서는 실수 연산을 줄이기 위해 integer 계수로 업-스케일 된 interpolation 필터 계수를 생성하여 사용하고 있기 때문에 interpolation 필터가 적용된 각 fractional-pel 위치의 값은 원본 픽셀 값 대비 64 배 업-스케일 된 값을 갖게 된다. 따라서 인코더와 디코더 에서는 업-스케일된 fractional-pel 값에 6 비트 쉬프트 연산을 통해 출력 비트의 비트 깊이를 원본 값과 동일하게 유지한다.

그림 11. 에서 $a_{0,0}$, $b_{0,0}$, $c_{0,0}$, $d_{0,0}$, $h_{0,0}$, 그리고 $n_{0,0}$ 은 1D interpolation 필터를 이용하여 만들 수 있는 fractional-pel 들이고 $e_{0,0}$, $f_{0,0}$, $g_{0,0}$, $i_{0,0}$, $j_{0,0}$, $k_{0,0}$, $p_{0,0}$, $q_{0,0}$, 그리고 $r_{0,0}$ 은 2D interpolation 필터를 사용하여 만들어진 $a_{0,i}$, $b_{0,i}$, $c_{0,i}$ 을 이용하여 만들 수 있는데, 여기에서 i 는 -3 부터 4 를 의미한다. 위 수식을 이용하여 HEVC 표준에 적용된 각각의 FIR 필터 계수를 개별 pixel 위치에 적용하여 fractional-pel 의 휘도 값을 만들 수 있는데 quarter fractional-pel 와 3/4 fractional-pel 의 경우 위에서 설명한 half-pel 과는 달리 위상 변위(Phase shift)를 이용하여 유도한 7 탭 필터를 이용하게 된다. 각각의 pixel 위치

별로 HEVC 에 적용된 fractional-pel 휘도 성분에 대한 수식은
아래 그림 13, 그림 14 과 같다.

$$\begin{aligned}
 a_{0,0} &= (-A_{-3,0} + 4*A_{-2,0} - 10*A_{-1,0} + 58*A_{0,0} + 17*A_{1,0} - 5*A_{2,0} + A_{3,0})/64 \\
 b_{0,0} &= (-A_{-3,0} + 4*A_{-2,0} - 11*A_{-1,0} + 40*A_{0,0} + 40*A_{1,0} - 11*A_{2,0} + 4*A_{3,0} - A_{4,0})/64 \\
 c_{0,0} &= (A_{-2,0} - 5*A_{-1,0} + 17*A_{0,0} + 58*A_{1,0} - 10*A_{2,0} + 4*A_{3,0} - A_{4,0})/64 \\
 d_{0,0} &= (-A_{0,-3} + 4*A_{0,-2} - 10*A_{0,-1} + 58*A_{0,0} + 17*A_{0,1} - 5*A_{0,2} + A_{0,3})/64 \\
 h_{0,0} &= (-A_{0,-3} + 4*A_{0,-2} - 11*A_{0,-1} + 40*A_{0,0} + 40*A_{0,1} - 11*A_{0,2} + 4*A_{0,3} - A_{0,4})/64 \\
 n_{0,0} &= (A_{0,-2} - 5*A_{0,-1} + 17*A_{0,0} + 58*A_{0,1} - 10*A_{0,2} + 4*A_{0,3} - A_{0,4})/64
 \end{aligned}$$

그림 13. integer-pel 을 이용한 1D interpolation

$$\begin{aligned}
 e_{0,0} &= (-a_{0,-3} + 4*a_{0,-2} - 10*a_{0,-1} + 58*a_{0,0} + 17*a_{0,1} - 5*a_{0,2} + a_{0,3})/64 \\
 i_{0,0} &= (-a_{0,-3} + 4*a_{0,-2} - 11*a_{0,-1} + 40*a_{0,0} + 40*a_{0,1} - 11*a_{0,2} + 4*a_{0,3} - a_{0,4})/64 \\
 p_{0,0} &= (a_{0,-2} - 5*a_{0,-1} + 17*a_{0,0} + 58*a_{0,1} - 10*a_{0,2} + 4*a_{0,3} - a_{0,4})/64 \\
 f_{0,0} &= (-b_{0,-3} + 4*b_{0,-2} - 10*b_{0,-1} + 58*b_{0,0} + 17*b_{0,1} - 5*b_{0,2} + b_{0,3})/64 \\
 j_{0,0} &= (-b_{0,-3} + 4*b_{0,-2} - 11*b_{0,-1} + 40*b_{0,0} + 40*b_{0,1} - 11*b_{0,2} + 4*b_{0,3} - b_{0,4})/64 \\
 q_{0,0} &= (b_{0,-2} - 5*b_{0,-1} + 17*b_{0,0} + 58*b_{0,1} - 10*b_{0,2} + 4*b_{0,3} - b_{0,4})/64 \\
 g_{0,0} &= (-c_{0,-3} + 4*c_{0,-2} - 10*c_{0,-1} + 58*c_{0,0} + 17*c_{0,1} - 5*c_{0,2} + c_{0,3})/64 \\
 k_{0,0} &= (-c_{0,-3} + 4*c_{0,-2} - 11*c_{0,-1} + 40*c_{0,0} + 40*c_{0,1} - 11*c_{0,2} + 4*c_{0,3} - c_{0,4})/64 \\
 r_{0,0} &= (c_{0,-2} - 5*c_{0,-1} + 17*c_{0,0} + 58*c_{0,1} - 10*c_{0,2} + 4*c_{0,3} - c_{0,4})/64
 \end{aligned}$$

그림 14. fractional-pel 을 이용한 2D interpolation

각 fractional-pel 위치별 색차 성분에 대한 interpolation 은 4 탭
필터를 사용하고 있으며 필터 계수는 휘도 성분과 동일한 방식으로
식 (7)을 이용하여 유도할 수 있다.

				he _{0,-1}					
	B _{0,0}	ab _{0,0}	ac _{0,0}	ad _{0,0}	ae _{0,0}	af _{0,0}	ag _{0,0}	ah _{0,0}	B _{1,0}
	ba _{0,0}	bb _{0,0}	...	...	...	...	...	...	
	ca _{0,0}	...	...	...	...	...	...	...	
	da _{0,0}	...	...	...	...	...	...	...	
eh _{-1,0}	ea _{0,0}	...	...	...	ee _{0,0}	...	...	...	ea _{0,1}
	fa _{0,0}	...	...	...	...	...	...	...	
	ga _{0,0}	...	...	...	...	...	...	...	
	ha _{0,0}	...	...	...	...	...	...	hh _{0,0}	
	B _{0,1}				ae _{0,1}				B _{1,1}

그림 15. 색차 성분에 대한 integer-pel 와 fractional-pel 의 위치

그림 15 는 회색의 integer-pel 위치에 대한 각각의 fractional-pel 위치를 나타낸 것으로 4 탭 필터를 이용하여 각 위치별 색차 성분에 대한 수식은 아래 그림 16 과 같이 정리할 수 있다.

$$\begin{aligned}
 ab_{0,0} &= (-2*B_{-1,0} + 58*B_{0,0} - 10*B_{1,0} - 2*B_{2,0})/64 \\
 ac_{0,0} &= (-4*B_{-1,0} + 54*B_{0,0} - 16*B_{1,0} - 2*B_{2,0})/64 \\
 ad_{0,0} &= (-6*B_{-1,0} + 46*B_{0,0} - 28*B_{1,0} - 4*B_{2,0})/64 \\
 ae_{0,0} &= (-4*B_{-1,0} + 36*B_{0,0} - 36*B_{1,0} - 4*B_{2,0})/64 \\
 af_{0,0} &= (-4*B_{-1,0} + 28*B_{0,0} - 46*B_{1,0} - 6*B_{2,0})/64 \\
 ag_{0,0} &= (-2*B_{-1,0} + 16*B_{0,0} - 54*B_{1,0} - 4*B_{2,0})/64 \\
 ah_{0,0} &= (-2*B_{-1,0} + 10*B_{0,0} - 58*B_{1,0} - 2*B_{2,0})/64
 \end{aligned}$$

그림 16. 색차 성분에 대한 interpolation

2. 3. 2. 2 Half-pel and Quarter-pel Search

Interpolation 을 통해 각 fractional-pel 들의 위치를 찾고 난 다음에는 생성된 fractional-pel 들 중 가장 최적인 fractional-pel 위치를 찾는 과정을 수행하게 된다. 그림 17 은 integer-pel 탐색 과정에서 선택된 최적의 integer-pel 위치를 기준으로 최적의 fractional-pel 위치를 찾는 과정을 나타내고 있다. 주어진 탐색 영역 안에 존재하는 integer-pel 들 중에서 가장 낮은 RD-cost 값을 갖고 있는 위치를 최적의 integer-pel 로 선택하는 과정이 IME 라고 한다면, FME 는 주어진 최적 integer-pel 의 주변에 존재하는 fractional-pel 들 중에서 가장 낮은 RD-cost 값을 갖고 있는 위치를 찾아 이를 최적의 fractional-pel 로 선택하는 과정이라 할 수 있다.

HM 인코더에서는 fractional-pel 탐색 시 half-pel 탐색 후 quarter-pel 탐색을 하는 2 단계 탐색을 통해 최종 best quarter-pel 를 선택하게 되는데 우선, half-pel 탐색에서는 최적의 integer-pel 위치를 초기 위치로 하여 그 주변의 8 개의 half-pel 에 대해서 SATD-based RD-cost 을 구하고 각 위치별 RD-cost 값 비교를 통해 가장 낮은 RD-cost 값을 보유한 위치를 최적의 half-pel 로 선택한다. 이후 quarter-pel 탐색에서는 이전 단계에서 선택 된 최적 half-pel 위치를 기준으로 동일하게 주변의 8 개의 quarter-pel 에 대해서 SATD-based RD-cost 를 계산하여 각 위치 별 RD-cost 값의 비교를 통해 최적의 quarter-pel 을 선택하는데 이렇게 얻은 움직임 벡터가 현재 PU 을 위한 움직임 벡터로 결정이 되며 이러한 일련의 fractional-pel 탐색 과정을 half-pel search 및 quarter-pel search 라 한다.

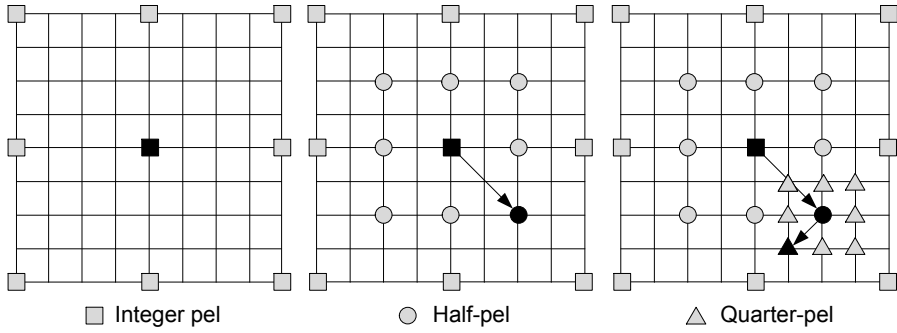


그림 17. HM 인코더의 fractional-pel 탐색 순서

Motion estimation에서는 인코더의 복잡도를 고려하여 RD-cost 값을 간소화 된 형태로 계산하는데 integer-pel 탐색 과정에서는 식(8)과 같이 SAD(sum of absolute difference) SATD 함수를 이용한 RD-cost 값을 적용하고, fractional-pel 탐색 과정에서는 식(9)와 같이 SATD(sum of absolute hadamard transformed difference) SATD 함수를 이용한 RD-cost 값을 적용하고 있다. 여기서 O 는 현재 블록의 원본 픽셀 값을, P 는 참조 블록의 후보 픽셀 값을 의미한다. 복잡도 감소 측면에서 움직임 매개 변수 부호화에 쓰이는 비트량 R 도 또한 실제 부호화를 하여 엔트로피 코더를 통해 그 값을 얻는게 아니라 Exp-Golomb code 를 이용하여 테이블로 저장하고 있는 비트량 표를 이용해 간소화 하여 계산하고 있다.

$$IntegerME, J_{LRD} = SAD(\sum |O_i - P_i|) + \lambda_{pred} \times R_{mvd} \quad (8)$$

$$FractionalME, J_{LRD} = SATD(\sum H \cdot |O_i - P_i| \cdot H^T) + \lambda_{pred} \times R_{mvd} \quad (9)$$

fractional-pel 에 대한 RD-cost 의 값의 경우 interpolation 된 fractional-pel 가 실제 존재하지 않는 값을 임의로 생성한 것이므로 성능을 고려할 경우 integer-pel 에 대한 계산 시 보다 좀 더 integer 교환 탐색이 필요하게 되므로 이에 따라 HEVC 표준에서는 integer-pel 와 fractional-pel 탐색 시 SATD 함수가 각각 달리 적용되어 있다.

제 3 장 제안하는 HM encoder hardware 구조

3.1 Hardware Pipeline 구조

가정하는 전체 pipeline 구조는 아래 그림 18 과 같으며 현재 수행하고 있는 연구의 범위는 그림에서 회색으로 칠해진 prediction stage 에 대한 부분이다.

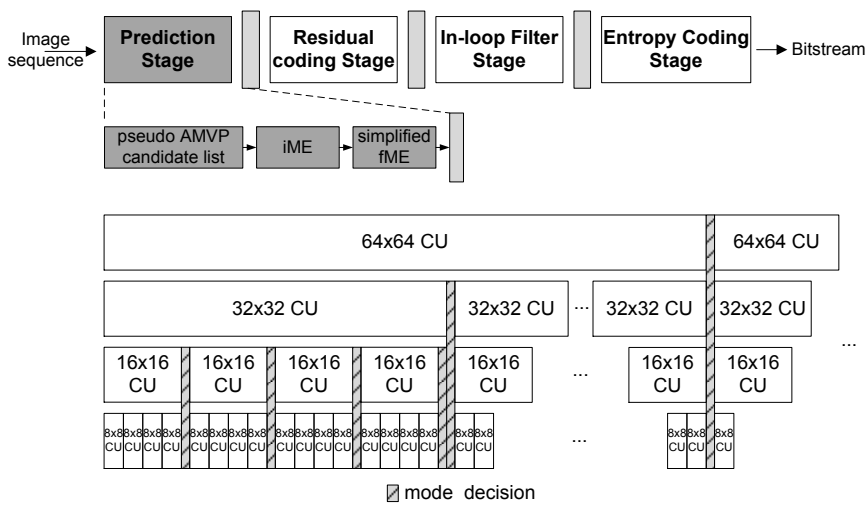


그림 18. 가정하는 HM 인코더의 Hardware pipeline 구조

HEVC 의 HM encoder 는 위 그림처럼 prediction stage, residual coding stage, in-loop filter state 와 entropy coding stage 의 4 단계 CTU 단위 pipeline 으로 구성된다고 가정한다. prediction stage 에서는 inter prediction 과 intra prediction 을 수행하게 되는데 본 논문에서는 inter prediction 에 대한 설계 방안에 대해 기술한다. Prediction stage 와 residual coding stage 는 CU depth 별로 서로 다른 pipeline 크기를 갖게 되고 PU 분할 모드인 $2N \times 2N$, $2N \times 2N$, $2N \times N$, $N \times 2N$, $2N \times nU$, $2N \times nD$, $nL \times 2N$, 및

nRx2N 의 순서로 순차적으로 수행 하되 주어진 연산 unit 들에서 최대한 병렬 수행하도록 하고 또한 하위 CU depth 는 상위 CU depth 와 mode 결정시 sync 를 한다고 가정 하였다. 각 CU depth 의 pipeline stage 는 PU 모드 결정 과는 무관하게 순차로 진행할 수 있다. 따라서 하나의 CU 내 마지막 순서의 PU 분할 모드인 nRx2N 에 대한 prediction stage 가 종료되면 해당 CU 의 decision process 와는 무관하게 다음 CU 의 첫번째 PU 분할 모드인 2Nx2N 에 대한 prediction stage 를 진행할 수 있다. 이때, prediction stage 의 시작지점인 MVP 를 결정하기 위한 AMVP candidate list 를 구성할 때 pipeline 구조와 HEVC 표준에 정의된 Z-scan order 의 인코딩 순서를 따르는 것에 의해 available 하지 않은 candidate 가 발생하게 된다. 따라서, IME 를 수행하기 위한 초기 탐색 위치를 결정할 때 slice 나 tile 의 boundary 조건에 기인한 spatial candidate 의 available 조건에 추가로 pipeline 구조에 기인한 available 조건을 고려하여 pseudo-AMVP candidate list 를 구성한 후 이 결과로 IME 의 초기 탐색 위치를 결정하도록 한다.

3. 2 Pseudo Merge and AMVP Candidate List

Pseudo merge and AMVP candidate list 를 위해 HW 의 pipeline 구조를 고려하여 각 조건 별로 spatial candidate 의 available 한 지 여 fractional 를 따지게 되는 경우, pipeline 구조의 가정에 따라 세 가지로 경우로 분류 할 수 있다. 이때 pipeline 의 기본적인 가정은 다음의 경우를 따르게 된다.

1. CU depth 는 parallel 하게 encoding 이 가능하며 하위 depth 의 4 개의 CU 가 encoding 동작이 완료 되면 상위 depth 와 sync 를 하여 mode decision 을 한다.
2. 동일 CU 안에서 PU Partition 은 encoding 순서에 따른 dependency 가 있다.
3. PU partition mode 간에는 dependency 가 없다.

위의 pipeline 의 가정을 기반으로 각각의 candidate list 구성을 위한 availability 에 대한 가정은 아래와 같이 세가지 정도로 분류할 수 있다.

Case 1) HM 의 encoding 순서와 동일하게 수행 하여 이전 PU 의 prediction 이 완료 되지 않아 현재 PU 의 spatial candidate 가 조건 별로 available 하지 않는 block 이 발생하는 경우

Case 2) merge candidate list 구성 시 PU $2N \times 2N$ 의 encoding 은 다른 PU partition mode 와는 달리 HM 에서 full RD cost 을 이용하여 merge estimation 을 수행한다. 따라서 제안하는 HW pipeline 구조 상 Full RD cost 의 연산 동작은 prediction stage 이후에 진행 되므로 $2N \times 2N$ 의 경우에 모든 spatial candidate 는 HM 과 동일한 availability 를 갖게 된다. 여기서 HM 과 동일한 availability 란, 이전 candidate block 의 mv 와 동일한 mv 로 인해 redundancy check

과정에서 available 하지 않은 block 이 발생하는 경우를 의미한다.

Case 3) 마지막으로 PU $2N \times 2N$ 외 나머지 PU partition 의 경우에도 prediction 을 reconstruction 바로 직전에 수행함으로써 모든 PU partition 에 대한 spatial candidate 가 HM 과 동일한 availability 를 갖게 되고 따라서 pipeline 구조에 따른 degradation 은 없게 되는 경우가 있다. 이 경우는 prediction 의 순서 변경으로 인해 workload 가 증가하게 되는 점을 고려해야 한다.

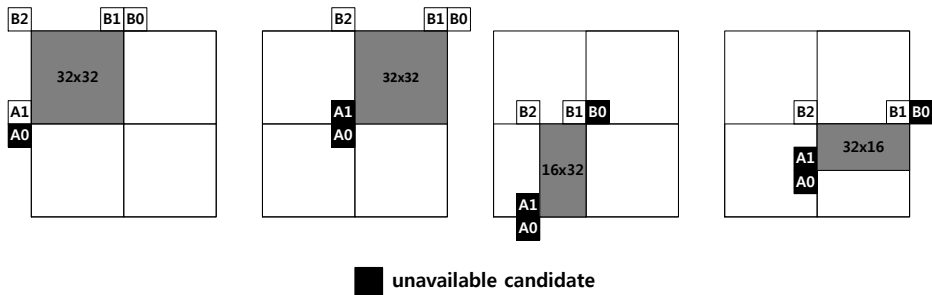


그림 19. Unavailable Spatial candidates for AMVP candidate list

그림 19 는 앞서 기술한 availability 의 조건에 따라 각 partition 별 5 개의 spatial candidate 중 unavailable 한 candidate 가 어떻게 되는지에 대한 구체적인 예시이다. IME 의 초기 탐색 위치는 unavailable 한 candidate 를 제외하고 구성 된 AMVP candidate list 에서 결정된 최적의 candidate 로부터 생성이 된다. 이때 각 stage 의 순서에 따라 현재 block 의 인코딩이 residual coding stage 를 수행될 시점에서는 앞서 pipeline stage 상 unavailable 했었던 candidate 들이 available 하게 된다. 따라서 mode decision 시 rate 를 구할 때는 이를 보상하여 인코딩 할 수

있도록 한다. 위의 availability 에 대한 분류에 대해 BD-rate 성능을 고려하여 실험한 결과는 아래와 같이 정리할 수 있다.

1 안. merge candidate list 구성 시 encoder 에서 pipeline 구조에 의해 approximation 을 할 경우 성능 변화 - 2Nx2N 은 제외하고 나머지 PU partition 에 대해서만 approximation 을 할 경우

2 안. AMVP candidate list 구성 시 encoder 에서 pipeline 구조에 의해 approximation 을 할 경우 성능 변화

3 안. merge candidate list 와 AMVP candidate list 를 구성 시 동시에 approximation 을 할 경우 성능 변화

표 1. Prediction stage 에서 pseudo merge 및 AMVP candidate list 에 의한 BD-rate[%] 성능 변화

1080p	Pseudo merge		Pseudo AMVP		Pseudo AMVP & merge	
	RA ¹	LB ²	RA	LB	RA	LB
Kimono	1.02	1.51	0.06	-0.01	1.13	1.43
ParkScene	0.80	0.96	0.05	0.01	0.75	0.91
Cactus	0.50	0.83	0.12	0.04	0.42	0.79
BasketballDrive	1.11	1.15	-0.04	0.08	0.94	1.22
BQTerrace	0.50	0.43	0.10	0.03	0.58	0.46
Average	0.79	0.97	0.06	0.03	0.76	0.96

¹Random access, ²Low delay-B

위의 실험에서 merge estimation 의 경우 AMVP 와 서로 dependency 가 없고 ME 를 수행하기 위해 pseudo candidate list 를 구성하지 않아도 무방하므로 merge 의 경우는 pipeline 상 residual coding stage 에서 수행하도록 가정하며 본 논문에서 2 안. Pseudo

AMVP candidate list 에 대해서만 적용하였으며, 이로 인한 BD-rate 저하는 random access 기준 약 0.06% 저하를 확인 할 수 있었다.

제 4 장 Interpolation Complexity Reduction for Fractional Motion Estimation

앞서 기술한 바와 같이 FME 의 연산 복잡도가 전체 HM 인코더의 연산 복잡도에서 제일 많은 부분을 차지하고 있다. 그 원인은 첫 번째로는 integer-pel 에서 fractional-pel 을 생성하기 위한 interpolation 을 수행하기 위한 반복적인 matrix 연산에 기인하고 있고, 두 번째로는 생성된 fractional-pel 들 중 최적의 위치를 찾기 위한 SATD 연산에 의해 발생하고 있다. 이러한 복잡한 FME 연산을 간소화 하기 위해 선행 연구들에서 제시된 방법들은 다음과 같이 분류하여 요약할 수 있다.

- 1) IME 수행 후 선택 된 최적 integer-pel 의 위치를 fractional-pel 의 최적 위치로 결정하고 FME 를 아예 수행 하지 않도록 하여 연산을 간소화 하는 방안
- 2) IME 를 수행하면서 발생한 integer-pel 의 계산 정보를 이용하여 간략한 계산을 통해 approximation 으로 최적 fractional-pel 의 위치를 결정하여 연산을 간소화 하는 방안
- 3) Interpolation 수행 시 interpolation 의 필터 탭 수를 적절히 줄여서 FME 의 연산을 간소화 하는 방안
- 4) Interpolation 수행 시 필터 탭 수는 그대로 유지할 하 되 필요한 탐색 위치의 수를 줄이는 방법을 이용하여 이에 따라 필요한 탐색 위치에 대해서만 선택적으로 interpolation 을 하는 방안

1 번의 경우, PU 분할 모드의 제한을 통해 FME 수행을 아예 하지 않도록 하면서 제한된 PU 분할 모드에서는 interpolation 과 fractional-pel 탐색 동작을 아예 수행 하지 않도록 하여 연산을 간소화 하는 방법으로 연산량 측면에서는 대폭 개선 효과를 기대할 수 있으나 심한 화질 열화를 피할 수 없다는 단점이 있다. 물론 나머지 방안들과 같이 interpolation 연산을 간소화 하기 위해 interpolation 필터 탭 수를 적절히 조절하여 interpolation 수행에서 발생하는 연산 복잡도를 낮추거나, 기존의 pixel-interpolation 과 같이 matrix 연산이 아닌 scalar 연산만 수행 할 수 있는 방법들을 고안하여 연산 량을 저감하는 방법 및 fractional-pel 의 탐색 위치를 적절히 줄여서 필요한 위치에 대해서만 선택적으로 interpolation 하는 방법들도 또한 연산 간소화에 따른 화질 열화는 피할 수 없다. 요점은 연산량 감소에 따른 화질 열화를 얼마나 최소화 할 수 있는지가 될 것으로 선행 연구에서 제시한 연산량 감소화 방안들에 대해 검토를 하여 FME 연산 복잡도는 최대한으로 개선하되 BD-rate 저하는 최소화 할 수 있는 방안이 무엇인지 찾아 보도록 하겠다.

4. 1 Computational Complexity for Interpolation

HM 인코더의 전체 복잡도에서 fractional-pel 생성을 위한 interpolation 필터링이 단일 연산으로는 약 35% 정도로 가장 높은 연산 비중을 보이고 있는데 이 중 pixel 의 휘도 성분에 대한 7/8 탭 interpolation 필터 연산이 70%이상 이기 때문에 interpolation 의 연산 복잡도를 줄이기 위해서는 pixel 의 휘도 성분에 대한 interpolation 연산 복잡도를 줄일 수 있는 방안을 우선적으로

고려하여야 한다. 구체적인 interpolation 연산 복잡도 저감 방안에 대한 실험에 앞서 우선 interpolation 의 연산량이 얼마나 되는지 우선 알아보고 이 후 각 실험 방법 별로 연산량이 얼마나 저감이 되는지 확인하고자 한다.

$A_{0,0}$	$a_{0,0}$	$b_{0,0}$	$c_{0,0}$	$A_{1,0}$
$d_{0,0}$	$e_{0,0}$	$f_{0,0}$	$g_{0,0}$	
$h_{0,0}$	$i_{0,0}$	$j_{0,0}$	$k_{0,0}$	
$n_{0,0}$	$p_{0,0}$	$q_{0,0}$	$r_{0,0}$	
$A_{0,1}$				$A_{1,1}$

그림 20. Quarter-pel 정밀도 기준 fractional-pel 의 위치

그림 20 은 quarter-pel 해상력에서 4x4 코딩 블록 크기를 기준으로 각 fractional-pel 의 위치를 나타낸다. 앞서 그림 13 과 그림 14 에서 HM 에서 수행되는 각 fractional-pel 들의 필터 연산 수식을 제시하였는데 해당 수식이 덧셈, 곱셈 및 쉬프트 연산으로 이루어져 있음을 상기 하였을 때 interpolation 을 위한 연산 복잡도는 아래 표와 같이 정리할 수 있다.

표 2. 4x4 블록 사이즈의 interpolation 을 위한 연산 복잡도

Fractional pixels	Operations		
	multiplication	addition	shift
A	7	6	6
a	8	7	6
b	7	6	6
c	7	6	6
d	7	6	6
e	7	6	6
f	7	6	6
g	8	7	6

h	8	7	6
i	8	7	6
j	8	7	6
k	7	6	6
n	7	6	6
p	7	6	6
q	7	6	6
r	7	6	6
Total	295		

표 2 를 기준으로 CU 8x8, 16x16, 32x32, 그리고 64x64 를 포함하는 한 개의 CTU 크기의 interpolation 을 수행을 위한 연산 복잡도는 중복 연산을 제거하고 총 1583040 이 되는데, 여기서 HM 의 interpolation 시 quarter-pel 중 $n_{x,x}$ 와 $d_{x,x}$ 의 경우 half-pel 의 선택 위치에 따라 interpolation 연산을 하지 않아도 되는 경우가 있기 때문에 이를 고려하여 최종적으로 HM 인코더에서 CTU 하나 당 interpolation 을 위한 연산 복잡도는 1376320 이라고 할 수 있다. 이때 전체 interpolation 의 연산량을 100%라고 하였을 때 half-pel 과 quarter-pel 을 interpolation 하기 위한 비율은 약 24.5% : 75.5%에 해당함을 확인할 수 있어 2D interpolation 계산으로 인해 전체 interpolation 에서 quarter-pel 의 위치 생성을 위한 interpolation 연산이 interpolation 연산 복잡도의 대부분을 차지하고 있는 것을 확인 할 수 있으므로 위의 실험 결과를 토대로 interpolation 연산량을 줄이기 위해서는 quarter-pel 을 위한 interpolation 연산량 개선에 집중을 하는 것이 효과적일 것이라는 사실을 확인할 수 있다.

HM 인코더에서 interpolation 을 위해서는 integer-pel 을 기준으로 해상력을 증가시켜서 fractional-pel 을 구하게 되는데, 이때 integer-pel 의 휘도 값을 기준으로 필터 계수의 조합을 통해

각 fractional-pel 위치의 휘도 값을 앞서 언급한 계산식을 통해 구하게 된다. 이후 각 fractional-pel 위치에 대해 SATD 로 distortion 을 구하고 여기에 각 위치 별 MV 에 대한 rate 을 더하여 RD-cost 을 구하여 이 중에 가장 낮은 RD-cost 를 갖는 위치를 best fractional-pel 로 선정하는 다소 복잡한 연산이 반복적으로 이어지므로 본 논문에서는 interpolation 연산 복잡도 간소화와 이에 따른 trade-off 로 BD-rate 성능저하를 최소화 하기 위한 방안을 고려하기 위해 앞 절에서 언급한 4 가지 방안 즉, 1) FME 를 아예 수행 하지 않는 방안, 2) integer-pel 탐색 시 발생한 계산 정보를 이용하는 방안, 3) interpolation 의 필터 탭 수를 적절히 조절하는 방안, 및 4) 탐색 위치 개수를 적절히 줄이는 방안에 대해 장단점을 알아본 후 이 방법 들 사이에서 연산 복잡도는 줄이면서 BD-rate 성능 저하는 최소한으로 할 수 있는 개선 방안을 찾아 보도록 한다.

4. 2 Hardware Complexity for Interpolation and FME

본 논문에서 제안한 Hardware 인코더의 구조에서 Interpolation 을 위한 hardware 구조는 H.264/AVC 에서 제안한 [14]의 구조를 HEVC 용으로 확장하여 차용하였다. 가로 방향으로 4 pixel 에 대하여 동시에 interpolation 을 수행하고 이를 연산하고자 하는 블록의 세로 크기만큼 반복하여 실행한다. 즉, Horizontal FIR 필터 4 개를 사용하여 연속된 4 개의 integer-pel 주변의 half-pel 과 quarter-pel 을 매 cycle 마다 생성하는데 half-pel 과 quarter-pel 을 만들기 위해 8-tap 필터를 사용하기 때문에 앞서 그림 12, 그림 13, 그림 14 에서 기술한 interpolation 방식에 따라 생성하고자 하는 fractional-pel 들의 앞에 위치한다.

3 개의 integer-pel 과 과 뒤에 위치한 4 개의 integer-pel 이 추가로 필요하게 된다. 매 cycle 마다 생성된 fractional-pel 들은 integer-pel 들과 함께 저장된다. 세로 방향의 fractional-pel 들은 vertical FIR 필터를 사용하여 수평방향의 fractional-pel 을 생성할 때와 동일한 방식으로 만들게 되며 이는 아래 그림 21 과 같이 도식화 할 수 있다.

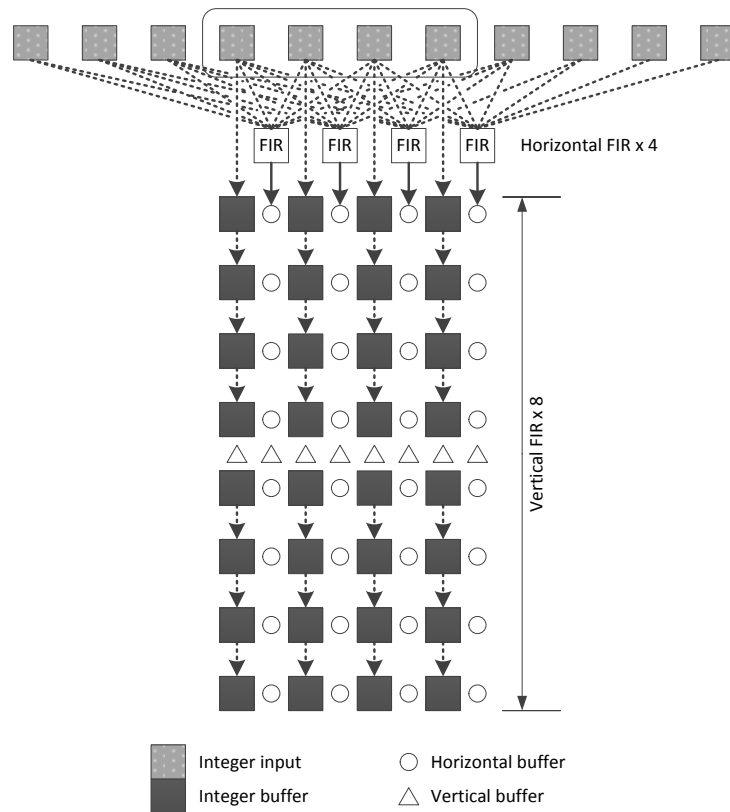


그림 21. 제안하는 HEVC 인코더의 hardware implementation 에서 interpolation unit 의 구조

여기서, interpolation 에 걸리는 시간은 PU partition 의 크기에 따라 달라지게 되므로 CU 의 depth 크기 별 각 PU partition 마다

interpolation 을 수행하기 위한 cycle 을 아래 표 3 과 같이 정리하였다. 표에서 (a)로 정의된 블록 수는 4x4 블록을 기준으로 하여 각 블록당 가로 4 pixel 단위로 나눈 수를 의미한다. PU 의 세로 길이는 내부 블록을 구성하는데 영향을 미치지 않으며 수행시간과 관계가 있다. (b)로 정의된 블록 당 cycle 수는 8 탭 필터링을 위해 블록의 integer-pel 단위의 세로 방향에서 위쪽 3 pixel 과 아래쪽 4 pixel 을 처리하는데 걸리는 수행 cycle 을 의미한다.

표 3. PU partition 크기 별 interpolation 연산 수행 cycle

CU Size	PU partition	W	H	# block (a)	cycle per block (b)	PU 별 수행 cycle (a) * (b)	PU 별 average 수행 cycle
64x64	2Nx2N	64	64	16	71	1136	1184
	2NxN	64	32	16	39	1248	
		64	32	16	39		
	Nx2N	32	64	8	71	1136	
		32	64	8	71		
	2NxN _U	64	16	16	23	1248	
		64	48	16	55		
	2NxN _D	64	48	16	55	1248	
		64	16	16	23		
	nLx2N	16	64	4	71	1136	
		48	64	12	71		
	nRx2N	48	64	12	71	1136	
16		64	4	71			
Sub total		8288					
32x32	2Nx2N	32	32	8	39	312	1344
	2NxN	32	16	8	23	368	
		32	16	8	23		
	Nx2N	16	32	4	39	312	
		16	32	4	39		

	2Nx _n U	32	8	8	15	368	
		32	24	8	31		
	2Nx _n D	32	24	8	31	368	
		32	8	8	15		
	nLx2N	8	32	2	39	312	
		24	32	6	39		
	nRx2N	24	32	6	39	312	
		8	32	2	39		
Sub total		2352					
16x16	2Nx2N	16	16	4	23	92	1664
	2NxN	16	8	4	15	120	
		16	8	4	15		
	Nx2N	8	16	2	23	92	
		8	16	2	23		
	2Nx _n U	16	4	4	11	120	
		16	12	4	19		
	2Nx _n D	16	12	4	19	120	
		16	4	4	11		
	nLx2N	4	16	1	23	92	
		12	16	3	23		
	nRx2N	12	16	3	23	92	
		4	16	1	23		
Sub total		728					
8x8	2Nx2N	8	8	2	15	30	2219
	2NxN	8	4	2	11	44	
		8	4	2	11		
	Nx2N	4	8	1	15	30	
		4	8	1	15		
Sub total		104					

위의 표의 결과를 바탕으로 64x64 크기의 CTU 하나의 FME 를 연산하기 위한 interpolation 과 FME 에 필요한 수행 cycle 을 아래 표와 같이 계산할 수 있다.

표 4. 64x64 CTU 한 개 연산을 위한 FME 의 수행 cycle

CU Size	#PU partition	#ref. picture	search point	#FME수 행 (4x4)	PU별 average 수행cycle	CU별 수행cycle
64x64	7	4	17	476	1184	563584
32x32	7	4	17	476	1344	639744
16x16	7	4	17	476	1664	792064
8x8	3	4	17	204	2219	452676
Total	2448068					

표 4 는 CTU 한 개 크기에 대해 FME 연산이 수행되는 횟수와 interpolation 을 수행하기 위한 cycle 에 대해 정리한 것으로 FME 수행 회수는 각 CU 크기 별 interpolation 연산이 발생하는 회수를 의미한다. Interpolation 연산은 FME 이 과정이 진행될 때 마다 한번씩 호출 되므로 FME 연산 회수와 interpolation 연산 회수는 동일하다. CU 별 수행 cycle 은 4x4 블록이 1 cycle 에 interpolation 연산이 완료된다는 가정하에 각 CU 크기 별 존재하는 PU partition 개수에 대해 모든 reference picture 의 개수와 half-pel 및 quarter-pel search 를 위한 search point 들의 수에 PU partition 의 interpolation 평균 cycle 을 곱으로 나타낼 수 있다.

표 5. 가정하는 4K@30fps HEVC 인코더의 specification

Specifications of an HEVC encoder	
Maximum resolution	UHD (3840x2160)
Maximum frame rate	30
# frames	4
# PU partitions	7
Search range	± 64
Frequency	400MHz

표 5 는 UHD 영상을 30fps, 400MHz 로 인코딩 하기 위한 가정하는 HEVC 인코더의 스펙을 나타낸다. 해당 인코더의 hardware 가정에서 64x64 크기의 CTU 하나를 처리하기 위한 수행 cycle 은 아래 식 (10)과 같이 약 6584 cycle 에 해당한다.

$$\frac{Cycles}{CTU} = \frac{frequency}{CTU\ rate} = \frac{400MHz * (64 * 64)}{(3840 * 2160) * 30fps} \approx 6584cycles \quad (10)$$

따라서, 앞서 계산한 FME 는 6584 cycle 에 모두 수행되어야 하므로 interpolation engine 은 약 372 개가 필요하게 된다.

4.3 Interpolation 연산 간소화를 위한 실험 검토

1) FME 연산 시 PU 분할 모드 제한을 통한 연산 간소화 방안

Integer-pel 탐색 후 구해진 최적의 integer-pel 위치를 FME 를 수행하지 않고 바로 최적의 fractional pixel 위치로 선택할 경우 HM 의 표준 테스트 조건의 1080p Class-B 시퀀스의 32 프레임 기준으로 random access 설정에서 약 9.7%, low delay 설정에서 약 9%의 화질 열화를 발생 시킨다. 심한 화질 저하를 방지하기 위하여 모든 PU 분할 모드에 대해 FME 를 수행하지 않도록 하는 것이 아니라 HEVC 공통 평가 기준의 Class-B 1080p 시퀀스에서 CU 모드 중 best 로 선택이 되는 비율이 약 80% 이상[15]인 2Nx2N 의 경우는 FME 를 수행하고 나머지 PU 분할 모드 중 최적으로 선택되는 비중이 낮은 4 개의 asymmetric 모드를 선택적으로 FME 를 수행하지 않도록 하여 심한 BD-rate 저하를 개선할 수 있다. 하지만 이럴 경우에도 BD-rate 의 저하가

4% 이상이 되어 본 논문에서 interpolation 연산량 감소와 동시에 화질 열화는 최소화 하려는 목표에 부합하지 않아 이후 실험에 대해 검토 시 우선 순위를 제일 낮게 두었다.

2) Quadratic 모델을 이용한 RD-cost interpolation

Quadratic interpolation model 은 IME 수행 후 구해 진 best integer-pel 과 그 주변 좌/우/상/하의 4 개 위치의 integer-pel 의 RD-cost 값을 smoothing 하는 계산을 통하여 동일 연장선상에 위치한 4 개의 quarter-pel 의 RD-cost 를 구하고 난 후 구해진 RD-cost 값을 비교하여 최적의 quarter-pel 위치로 선택하는 방식이다. Macro block(MB)의 사이즈 16x16 으로 제한되어 있던 H.264/AVC 에서 최초 제안한 방식으로 이를 지원하는 블록의 크기가 더 커진 HEVC 에 그대로 적용할 경우 error 율도 따라서 증가하게 되어 BD-rate 의 저하가 심하게 발생할 수 밖에 없고, 선행 논문에서 사용한 metric 인 RD-cost 가 앞서 식 (8),(9)에서 언급한 바와 같이 distortion 과 rate 의 합으로 이루어져 있음을 상기 하였을 때, 각 fractional-pel 의 위치 별 다른 MV 의 길이에 대한 값이 반영되어 있는 상태에서 approximation 을 하게 되면 당연히 error 율이 커질 수 밖에 없다는 점이 있다. 하지만, 해당 방법은 기존에 interpolation 이 pixel 의 휘도값을 이용하기 위해 복잡 다단한 matrix 연산을 필요로 하였는데 이를 단순한 scalar 연산으로만 대체 가능한 아이디어를 내포하고 있으므로 이 후 실험에서는 위의 연구를 좀 더 개선할 수 있는 방법을 제안 하였다.

3) Interpolation 필터 탭 개수 조절을 통한 연산 간소화 방안

그림 22 는 interpolation 연산을 간소화 하기 위한 각 방법 별 연산에 필요한 integer-pel 의 위치와 integer-pel 의 정보를 이용하여 계산을 통해 생성하게 되는 fractional-pel 들의 위치를 나타낸 것이다.

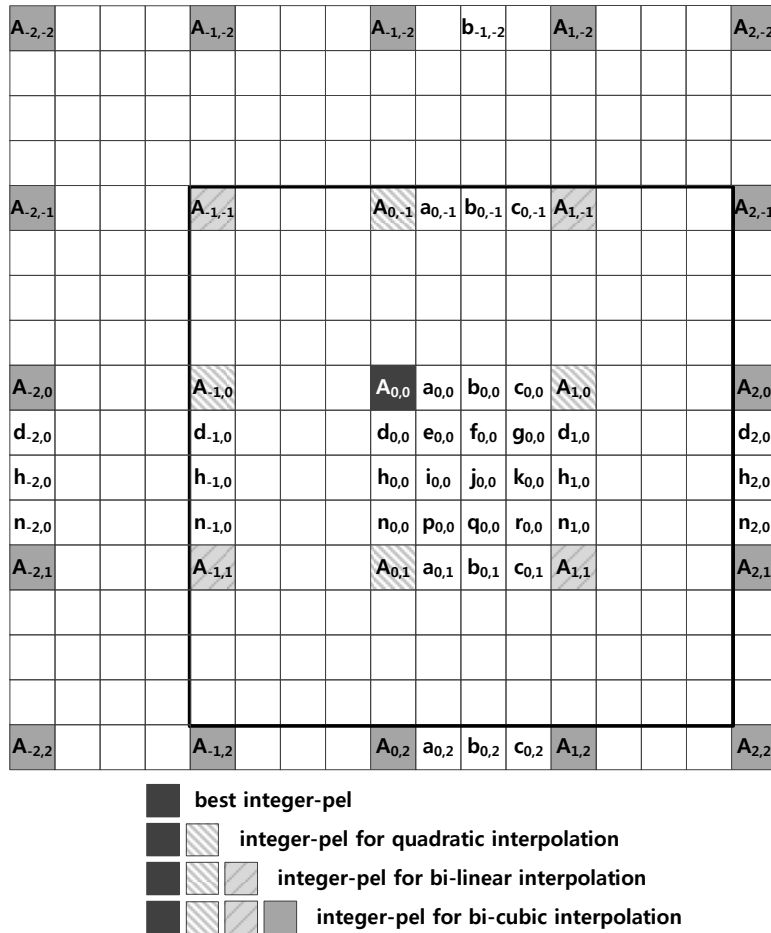


그림 22. Interpolation 연산 간소화 방법 별 integer-pel 과 fractional-pel 의 위치

IME 수행 후 선택된 최적의 integer-pel 주위로 각 interpolation 방법 별로 fractional-pel 생성을 위해 부가적으로 필요한 integer-pel 에 대해 알아보면 우선, quadratic 모델의 경우 최적 integer-pel 과 그 주변 상/하/좌/우 4 개의 integer-pel 의 정보가 필요하고, bilinear 모델의 경우 최적 integer-pel 과 그 주변 8 개의 integer-pel 의 정보가 필요하다. 그리고 bicubic 모델의 경우 주변 24 개의 integer-pel 의 정보를 이용하게 된다.

3-2) 양선형 (Bilinear) 모델을 이용한 interpolation

Bilinear interpolation 모델은 interpolation 을 위한 영역이 지엽적으로는 선형적인 특성을 갖고 있다는 가정에 기반하는 것으로서 인접한 두 integer pixel 간 변위가 크지 않다면 해당 integer-pel 과 동일 선상에 존재하는 fractional pixel 들은 좌/우 혹은 상/하 integer-pel 들의 평균값으로 간단히 계산할 수 있다.

본 논문에서는 interpolation 연산 수행 시 일반적인 interpolation 과 같이 인접 pixel 의 휘도 값을 이용하는 대신 RD-cost 혹은 SATD 값 만으로 half-pel 과 quarter-pel 위치의 interpolation 을 수행하는 방법을 제시한다. 이럴 경우, 탭 필터링 연산을 수행하기 위한 복잡한 연산을 제거하여 연산량을 간소화 할 수 있으며, pixel interpolation 으로 하여 최적 fractional-pel 을 탐색하기 위해 추가적으로 각 위치별 RD-cost 값 혹은 SATD 값을 계산하는 단계를 건너 뛰고 interpolation 연산 후 비교기를 통해 즉시 최적의 fractional-pel 을 고를 수 있다는 장점이 있다.

Bilinear interpolation 모델을 사용하기 위해서는 최적의 integer-pel 위치 주변으로 8 개 point 에 대한 RD-cost 혹은

SATD 값을 구하여 그 값을 기준으로 아래 수식 (11)을 이용하여 각각의 fractional-pel 에 대응하는 RD-cost 혹은 SATD 값을 계산 할 수 있다.

$$\begin{aligned}
 a_{0.0} &= (A_{0.0} + b_{0.0} + 1)/2 \\
 b_{0.0} &= (A_{0.0} + A_{1.0} + 1)/2 \\
 &\vdots \\
 e_{0.0} &= (b_{0.0} + h_{0.0} + 1)/2 \\
 h_{0.0} &= (A_{0.0} + A_{0.1} + 1)/2 \\
 &\vdots \\
 r_{0.0} &= (b_{0.1} + h_{1.0} + 1)/2
 \end{aligned} \tag{11}$$

3-3) 양입방 (Bicubic) 모델을 이용한 interpolation

Bicubic 모델을 이용한 interpolation 연산은 구하고자 하는 fractional-pel 위치의 수직 혹은 수평 방향의 4 개 pixel 의 값을 이용하는 것으로써 앞서 설명한 bilinear 모델이 2 개 integer-pel 의 정보만을 이용한다는 점과 비교하여 계산 량은 좀 더 많지만 주변의 특성을 좀 더 많이 활용하게 되므로 interpolation 의 결과는 당연히 좀 더 정밀할 것으로 기대할 수 있다.

본 논문에서는 Bicubic 모델을 이용하여 interpolation 연산 수행 시 앞서 bilinear 모델과 같이 인접 pixel 의 휘도 값을 이용하는 대신 RD-cost 혹은 SATD 값 만으로 half-pel 과 quarter-pel 위치의 interpolation 을 수행하는 방법을 적용 하였다. 따라서 IME 후 결정된 최적 integer-pel 과 그 주변의 24 개 integer-pel 에 대한 정보가 필요하며 아래 식(12)을 이용하여 각 integer-pel 에 대한 weighting 을 주어 필요로 하는 fractional-pel 위치에 대한 값을 계산할 수 있다.

$$\begin{aligned}
a_{0,0} &= (-4A_{-1,0} + 53A_{0,0} + 18A_{1,0} - 3A_{2,0} + 32)/64 \\
b_{0,0} &= (-A_{-1,-1} + 9A_{0,0} + 9A_{1,0} - A_{2,0} + 8)/16 \\
&\vdots \\
e_{0,0} &= (-4d_{-1,0} + 53d_{0,0} + 18d_{1,0} - 3d_{2,0} + 32)/64 \\
h_{0,0} &= (-A_{-1,-1} + 9A_{-1,0} + 9A_{-1,1} - A_{-1,2} + 8)/16 \\
&\vdots \\
r_{0,0} &= (-3n_{-1,0} + 18n_{0,0} + 53n_{1,0} - 4n_{2,0} + 32)/64
\end{aligned} \tag{12}$$

최적 fractional-pel 을 탐색하기 위해 추가적으로 각 위치별 RD-cost 값 혹은 SATD 값을 계산하는 단계를 건너 뛰고 interpolation 연산 후 비교기를 통해 즉시 최적의 fractional-pel 을 선택할 수 있다는 점은 동일하나, bilinear 모델과 비교하여 연산 저감량과 BD-rate 성능간에 trade-off 관계를 보일 것으로 예상할 수 있다.

4) 필터 탭 수의 저감을 이용한 pixel interpolation

논문에서 제안하고자 하는 Bilinear 혹은 Bicubic interpolation 모델이 기존의 pixel-interpolation 이 아닌 RD-cost 혹은 SATD interpolation 이지만 큰 범주에서 interpolation 을 위한 필터 탭 수의 저감에 속한다고 할 수 있다. 이 연장 선상에서 pixel 의 휘도 값을 이용하는 기존의 interpolation 방법을 사용하되 interpolation 필터 탭 수를 적절히 줄여 연산 복잡도 감소를 기대할 수 있다. 이전 표준인 H.264/AVC 에서는 6/2 탭을 이용하고 있는데 비해 HEVC 에서는 7/8 탭을 채용하고 있어 HEVC 의 interpolation 연산 복잡도를 기준으로 H.264/AVC 의 interpolation 연산 복잡도를 비교해보면 약 30% 남짓[16]에 불과한 것을 확인 할 수 있다.

이는 HM 인코더에서 interpolation 연산 중에서도 제일 복잡한 연산량을 차지하고 있는 quarter-pixel 들에 대한 interpolation 을 위한 interpolation 필터 탭 수가 급격히 늘어난 것에 의해 발생하는 것으로 연산 복잡도 측면을 고려하였을 경우 6/2 탭 혹은 화질 열화에 대한 성능을 고려하여 [8]와 같이 8/2 탭을 사용하는 것을 고려해 볼 수도 있겠지만 기본적으로 pixel 의 휘도 값을 이용한 interpolation 은 하드웨어 파이프라인 구조에서 지연을 발생시킬 수 밖에 없는 방법이므로 RD-cost 혹은 SATD 값을 이용한 bilinear 모델 혹은 bicubic interpolation 모델을 제일 우선 순위로 고려한 뒤 해당 방법은 후 순위로 고려하도록 하였다.

Bilinear 및 Bicubic 모델을 이용하여 RD-cost 혹은 SATD 값으로 각 fractional-pel 위치의 값으로 결정할 경우 기존의 pixel 휘도 값을 이용한 탭 필터링 방식의 interpolation 방법보다 정밀도는 낮아지게 되어 얼마간의 화질 저하는 피할 수 없다. 다만 interpolation 을 위한 연산 량은 matrix 연산이 아닌 scalar 연산만으로 대체가 되면서 획기적으로 개선할 수 있는 장점이 있으므로 적절한 조합을 통해 화질 저하를 최소화 할 수 있는 방안을 찾는 것을 본 논문에서는 목표로 하였다. Bilinear 모델과 Bicubic 모델 interpolation 은 기본적으로 생성하려는 fractional-pel 의 주변 integer-pel 의 값의 평균적인 특성을 이용하는 것이므로 해당 방법을 이용하여 최상의 성능을 기대하기 위해서는 주변 pixel 들의 분포가 균일(homogenous)하거나 단일 형태(unimodal)한 특성을 보여야 한다는 단서 조건이 필요하다. 이를 위해 아래와 같이 사전 실험을 통해 IME 에서 최적으로

선택된 integer-pel 을 기준으로 주변 pixel 들의 분포가 어떠한 특성을 갖고 있는지 확인해 보았다.

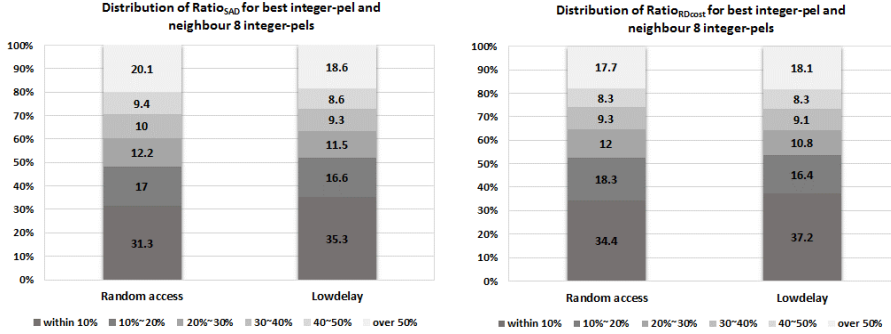


그림 23. 1080p 5 개 시퀀스에서 최적 integer-pel 과 그 주변 8 개 integer-pel 의 RD-cost 값 과 SAD 값의 분포 비율

그림 23 의 왼쪽은 IME 수행 후 최적으로 선택 된 integer-pel 과 그 주변 8 개 integer-pel 들의 SAD 값 중 최소값과 최대값의 차이에 대한 분포이고 오른쪽은 동일한 방법으로 RD-cost 값에 대한 분포로 HM 표준의 공통 평가 조건에서 Kimono, PartkScene, Cactus, BasketballDrive, BQTerrace 5 개의 1080p sequence 의 각 32 프레임의 평균 결과이다. 각 분포는 Ratio_x 로 나타내었는데 여기서 X 는 SAD 값 및 RD-cost 를 의미하고 Ratio_x 에 대한 계산식은 (13)과 같이 표현 할 수 있다.

$$Ratio_X = (X_{Max} - X_{Min}) / X_{Max} \quad (13)$$

위 그림에서 가로축의 within 10%라고 함은 integer-pel 9 개들 중 최소의 SAD 값과 최대의 SAD 값의 차이가 10%를 넘지 않을 경우를 의미하며 세로축은 각 경우에 대한 발생 비율 나타낸다. 즉,

다시 말하자면 Ratio_x 는 IME 수행 후 구해진 integer-pel 의 SAD 값 분포 혹은 RD-cost 값의 분포로 최적 integer-pel 주변의 특성이 얼마나 균일한지를 나타내는 것으로써 0%에 근접할수록 균일한 특성을 보이고 있다고 말할 수 있으며 100%에 근접할수록 균일한 특성을 가지지 못한다고 할 수 있다. 균일한 경우는 이미지의 배경이나 움직임이 거의 존재하지 않은 영역에 해당하며 균일하지 못한 영역은 object 의 edge 나 이미지의 boundary 혹은 object 의 움직임이 급격히 발생하는 지점이라고 할 수 있다.

RD-cost 값의 분포와 SAD 값의 분포가 둘 간에 큰 차이가 없는데, RD-cost 함수가 식(8),(9)에서와 같이 distortion 과 MV 의 율(Rate)의 값으로 이루어져 있어서 RD-cost 를 metric 을 하였을 경우는 integer-pel 위치 별로 다른 크기의 율 값이 더해져서 일반적으로 생각했을 때 SATD 값만으로 분포를 보았을 때와 RD-cost 값으로 분포를 보았을 때 차이를 많이 보여야 한다고 생각하기 쉽다. 하지만, RD-cost 의 전체 값에서 SAD 값이 차지하는 비중이 대부분이고 율 값이 차지하는 비율은 약 30% ~ 10% 미만으로 아주 작은 영향만을 미치게 되므로 위 그래프와 같은 결과가 나온다고 할 수 있다. 앞서 그림 23 가 bilinear interpolation 을 사용 하기 위한 사전 실험 결과라면, 그림 24 는 bicubic interpolation 방법을 사용하기 적절한지 여부를 판단하기 위한 실험 결과로 각각 SAD 값과 RD-cost 값에 대해 IME 수행 후 최적으로 선택 된 integer-pel 와 그 주변 24 개 integer-pel 들의 값 중 최소값과 최대값의 차이 분포인 Ratio_x 를 나타낸 그래프이다. 최적 integer-pel 주변 8 개 위치만을 보았을 때와는

달리 좀 더 넓은 영역에 대해 검토를 하였을 경우 주변 영역의 변화하는 특성이 더 많이 반영 되면서 균일 하다고 판단되는 비율이 상대적으로 적어지는 경향을 확인 할 수 있다. 하지만 주변 8 개 pixel 만 검토하지 않고 더 많은 24 개 pixel 를 보았을 때에도 균일한 특성이 유지 되고 있는 부분은 매우 높은 확률로 균일한 특성을 보이는 영역이다 라고 할 수 있다. 그림에서 random access 설정의 경우 약 10%, low delay 설정의 경우 약 12%의 경우가 해당이 된다고 할 수 있으며 이런 경우에 대해서 bilinear 나 혹은 bicubic 과 같은 approximated interpolation 을 적용하여 fractional-pel 을 생성할 경우 연산은 간소화 하면서 화질 저하는 최소화 할 수 있다고 가정하여도 타당하다.

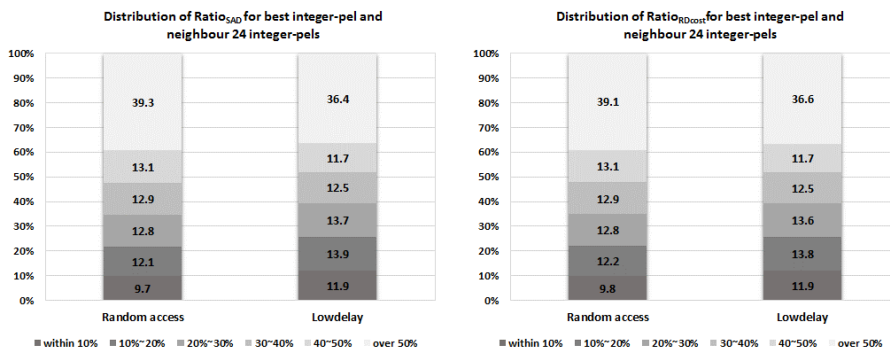


그림 24. 1080p 5 개 시퀀스에서 최적 integer-pel 와 그 주변 24 개 integer-pel 들의 RD-cost 값 과 SAD 값의 분포 비율

4. 4 Bilinear 와 Bicubic interpolation 의 적용

앞서 실험을 통해 최적 integer-pel 와 그 주변의 SAD 값 혹은 RD-cost 값의 분포를 통해 지엽적인 영역에서의 균일 특성 여부를 확인 해보았다. 이 결과를 이용하여 half-pel 과 quarter-pel

단위의 approximated interpolation 을 수행하여 interpolation 연산을 간소화 하는 방안에 대해 4 가지로 분류하여 실험해 보았다. 아래 실험은 모두 HM 공통 평가 조건[17]의 1080p class-B 에 해당하는 5 개 시퀀스 별로 각 32 프레임에 대해 QP 22, 27, 32, 37 의 조합으로 low delay 및 random access 설정으로 수행한 후 결과를 정리하였다.

4. 4. 1 Bilinear Interpolation

1) SATD 를 이용한 Bilinear Interpolation

IME 수행 후 선택이 된 최적 integer-pel 과 그 주변 8 개 pixel 의 SATD 를 계산하여 저장한 후 이를 이용하여 half-pel 단위와 quarter-pel 단위의 각 위치 별로 SATD를 계산 한다. 앞서 SAD 로 분포를 확인 하였는데 interpolation 시 metric 을 SATD 변경한 이유는 HM 인코더가 IME 는 SAD based RD-cost 를 이용하고 FME 는 SATD based RD-cost 를 이용하기 때문에 이를 준수하기 위함으로 HM 코드에서도 IME 후 최적으로 선택된 integer-pel 도 FME 를 수행하기 위해 별도의 SATD 계산을 수행하게 된다. 동일 pixel 위치에서 metric 을 SAD 에서 SATD 로 변경할 경우 hardamard transform 계산량 추가에 의해 해당 pixel 이 갖고 있는 distortion 값이 약 1.8 배 증가하게 되나 앞서 측정한 $Ratio_x$ 는 각 integer-pel 들 사이의 상대적인 비교 값이기 때문에 metric 을 SAD 에서 SATD 로 변경한다 하더라도 결과에 전혀 영향을 미치지 않는다. 오히려 IME 수행 후 $Ratio_x$ 를 계산하기 위해 모든 필요한 integer-pel 들을 SATD 로 변환한 후 계산을 수행하게 되면 불필요한 연산의 overhead 가 증가하게 되는

결과를 초래하면서 본 논문의 달성 목표와 반대되는 효과가 나타난다.

SATD를 이용한 bilinear interpolation 계산 수식은 앞서 기술한 식 (10)을 이용하게 되며 FME를 수행하기 위해 최적 integer-pel 주변의 각 fractional-pel의 SATD 값이 계산 되면 위치 별로 이미 정해져 있는 MV의 rate 값을 더하여 RD-cost 값을 간단히 계산할 수 있다. 주어진 quarter-pel의 위치에서 RD-cost 값을 비교 후 가장 낮은 RD-cost 값을 갖고 있는 위치를 최적의 fractional-pel 위치로 선정 후 FME를 종료한다.

2) RD-cost를 이용한 Bilinear Interpolation

위의 1번 방안과 동일한 방법으로 FME를 수행 하되 SATD 값이 아닌 RD-cost 값을 이용하여 bilinear interpolation을 수행한다. integer-pel의 RD-cost 값으로 interpolation을 수행 시 각각 얻어진 fractional-pel 위치의 RD-cost 들은 추가적인 연산 없이 단순 비용 값 비교를 통해 최적의 fractional-pel 위치를 선정할 수 있다.

그림 25는 사전 실험을 통해 확인 한 IME수행 후 확인 된 최적 integer-pel의 위치와 그 주변 pixel들의 분포 특성을 반영하여 SATD 및 RD-cost를 각기 이용하여 bilinear interpolation에 적용하여 fractional-pel을 생성하고 최적 fractional-pel 탐색까지 진행하여 BD-rate 성능 변화가 어떻게 되는지 확인한 결과이다.

그래프에서 가로축의 within 10%라는 의미는 $Ratio_x$ 가 10% 이내인 경우에 대해서만 bilinear interpolation을 적용하고 나머지 $Ratio_x$ 에 대해서는 HM 표준과 같이 7/8 탭 필터링을 이용하여

interpolation 을 수행 한 후 BD-rate 성능 변화를 확인한 것으로 즉, 1080p Class-B 5 개 시퀀스의 평균 결과로 $Ratio_X$ 가 10% 이내인 경우는 약 36% 정도가 발생하고 있는데 이 경우만 bilinear interpolation 을 적용할 경우 BD-rate 저하는 random access 설정에서 약 0.05%로 거의 변화가 없음을 알 수 있다. 또한 당연히도 $Ratio_X$ 값을 키워서 bilinear interpolation 의 허용 제한 범위를 늘일수록 BD-rate 성능이 점점 더 저하되는 것을 확인할 수 있고 전부 허용을 하였을 경우에는 BD-rate 성능 저하가 매우 심한 것을 확인할 수 있는데 이것은 사전 예상했던 바와 같이 최적 integer-pel 주변의 분포가 균일하지 않으면 bilinear interpolation 을 통해서 계산 량은 줄일 수 있을지라도 화질 열화로 인해 별다른 이득을 얻을 수 없음을 의미한다.

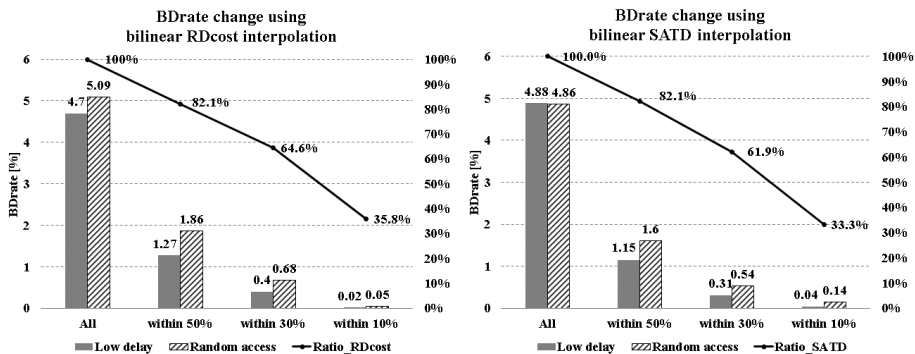


그림 25. 최적 integer-pel 주변 분포의 특성을 반영한 2 가지 방법으로 bilinear interpolation 적용 시 BD-rate 성능 변화

4. 4. 2 Bicubic Interpolation

1) SATD 를 이용한 Bicubic Interpolation

위의 1 번과 동일하게 SATD 를 이용하여 fractional-pel 위치의 interpolation 을 수행 하되 bicubic interpolation 방법을 이용한다. 이때의 수식은 위에서 기술한 식 (11)을 이용하는데 bicubic interpolation 을 사용하기 위해서는 최적 integer-pel 와 그 주변의 8 개 아닌 24 개 pixel 의 정보를 이용한다. 각 위치 별 SATD 를 계산 한 후에는 FME 수행을 위한 fractional-pel 위치 별로 rate 값을 더 하여 RD-cost 값을 계산 한 후 비교를 통해 가장 낮은 RD-cost 값을 갖고 있는 위치를 최적의 fractional-pel 위치로 선정 한 후 FME 를 종료한다.

2) RD-cost 를 이용한 Bicubic Interpolation

Bicubic interpolation 방법을 이용하여 fractional-pel 의 값을 생성하되 SATD 값이 아닌 RD-cost 값을 이용하여 approximated interpolation 을 수행한다. 앞서 수행한 방법들과 동일하게 IME 후 주변 pixel 들의 RD-cost 정보를 수집하여 식 (11)을 통해 각 fractional-pel 위치 별로 RD-cost 값을 구한 후 각 위치 별 RD-cost 값을 비교하여 최적으로 생각되는 fractional-pel 의 위치를 결정 한다.

Bilinear interpolation 방법과는 다르게 bicubic interpolation 은 최적 integer-pel 주변 8 개 pixel 들의 정보 대신 주변 24 개 pixel 들의 정보를 이용하게 되므로 주변의 linear 한 특성이 좀 더 세부적으로 반영되면서 bilinear interpolation 의 결과와 비교하여

Ratio_X 로 주어진 동일한 approximated interpolation 허용 범위에서 BD-rate 성능이 약 1.5 배 ~ 2 배 가량 향상 되는 것을 확인할 수 있다. 즉, 특정 fractional-pel 위치를 기준으로 bilinear interpolation 은 주변 2 개 pixel 의 정보를 이용하지만 bicubic interpolation 은 주변 4 개 pixel 의 정보를 이용하게 되므로 좀 더 정밀한 interpolation 이 가능해졌고 이에 비례해서 성능이 향상 되었다는 의미로 이해할 수 있다. 하지만, Ratio_X 발생 비율에 따른 pixel-interpolation 연산량을 고려하여 전체 연산 저감량을 고려하게 될 경우 bicubic interpolation 을 위한 사전 24 개 위치에 대한 RD-cost 혹은 SATD 연산으로 부가적인 연산이 증가하게 되는 것으로 인해 bilinear interpolation 방법이 Ratio_X 의 동일 발생 비율에서 연산량 저감 효과는 더 우세하게 된다. 따라서 이 후 에는 bilinear interpolation 을 위주로 개선 사항에 대해 실험을 수행 하도록 한다.

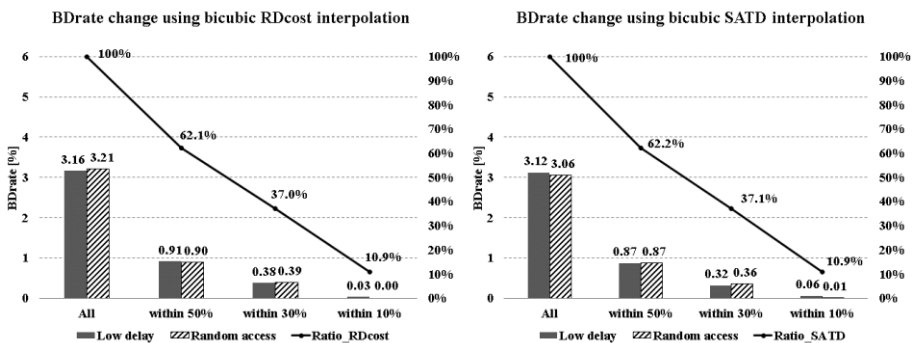


그림 26. 최적 integer-pel 주변 분포의 특성을 반영한 2 가지 방법 별로 bicubic interpolation 적용 시 BD-rate 성능 변화

4. 5 Interpolation Filter 개수 저감을 통한 연산 간소화

지금까지의 실험에서 interpolation 시 pixel-interpolation 의 방법을 수행 하지 않고 fractional-pel 의 값을 approximation 할 수 있는 방안에 대해 알아 보았다. 해당 방법들은 기존의 pixel-interpolation

의 방법과 비교하여 연산량은 대폭 절감할 수 있지만, interpolation 한 결과의 정밀도 측면에서 취약할 수 밖에 없으므로 BDRate 성능에서 열세라고 할 수 있다. 따라서, pixel-interpolation 과 앞서 수행한 SATD 혹은 RD-cost 를 이용한 bilinear 혹은 bicubic interpolation 방법 간에 연산 저감량과 BDRate 성능 사이에서 너무 한쪽으로만 치우치지 않고 중간 정도의 성능을 기대할 수 있는 방법도 고려해 볼 필요가 있다. 선행 연구에서는 pixel-interpolation 은 수행을 하되 interpolation 을 위한 필터의 개수를 줄여서 연산을 간소화 하는 방안에 대해서도 제안을 하였는데, 이전 표준인 H.264/AVC 에서는 quarter-pel 단위 interpolation 을 위해 6/2 탭 필터를 사용하였으므로 이를 그대로 HEVC 에 적용하게 되면 기존에 적용 되어 있던 7/8 탭과 비교하여 필터의 개수가 너무 큰 차이가 발생하여 화질 열화가 심하게 발생하게 되므로 화질 열화는 최소화 하고 동시에 연산량은 감소시키기 위해 [8] 에서는 8/2 탭 필터를 적용하였다. 즉, half-pel 에 대해서는 6 탭이 아닌 8 탭으로 정밀도를 증가시키고 이로부터 나온 결과를 이용하여 quarter-pel 에 대해서는 2 탭을 적용하여 간소화 된 연산을 목표로 하였는데 이것은 앞서 설명한 바와 같이 전체 복잡도에서 2D 필터링 연산이 들어가는 quarter-pel 의 interpolation 복잡도가 전체 interpolation 연산의

복잡도에서 약 75%를 차지하고 있기 때문에 quarter-pel 에 대해서만 필터의 수를 감소시키는 방안은 전체 interpolation 연산량 감소 측면에서 효과적이라고 할 수 있다. HEVC 에 6/2 필터 탭을 적용하여 interpolation 한 방안과 8/2 탭을 적용한 방안에 대해 BDRate 의 성능 변화를 확인해 보면 아래 표와 같이 정리할 수 있다.

표 6. Interpolation filter 개수 저감 방법 별 BDRate 성능 변화 비교

sequences	BDRate [%]			
	6/2 tap filtering		8/2 tap filtering	
	LB ¹	RA ²	LB ¹	RA ²
Kimono	0.57	0.73	0.47	0.61
ParkScene	2.45	2.85	2.45	2.71
Cactus	2.37	2.05	2.37	1.91
BaksetballDrive	1.05	1.54	1.05	1.42
BQTerrace	3.45	5.98	2.54	5.12
Average	1.98	2.63	1.78	2.35

¹Low delay-B, ²Random-access

지금까지 실험한 bilinear 및 bicubic interpolation 과 interpolation 탭 필터 수 저감의 종류 별로 BDRate 성능 과 이때의 interpolation 연산 복잡도를 표 7 과 같이 정리할 수 있다. 여기서 interpolation 의 연산 복잡도는 64x64 크기의 CTU 한 개를 인코딩 하기 위해 필요한 연산 량을 기준하였으며 이는 앞서 표 2 에 기술한 바에 근거하였다. 표 7 에서 interpolation 연산 복잡도는 대체로 BDRate 저하 율과 trade-off 관계를 갖고 있음을 알 수 있는데 이 것은 좀 더 정밀한 interpolation 결과를 위해 연산 량이 많아지면 많아 질수록 BDRate 성능은 좋아지고 연산 량을 줄이게 되면 BDRate 저하는 피할 수 없다는 단순한 명제를 재확인하였다고 할 수 있다. 또한, 기존의 pixel-interpolation 의

정밀도가 다른 방법들 우세하다는 점도 확인할 수 있었다. 따라서, 지금까지의 실험에서 interpolation 의 연산 복잡도를 최대한 간소화 하면서 BDRate 저하율은 최소한으로 하기 위해서는 이전 단계의 계산 정보를 최대한으로 활용하여 세밀한 설계가 필요하므로 선행 연구들과 같이 하나의 방안만 적용한 설계가 아닌 복합적인 방법 통한 interpolation 연산 간소화 방안을 설계하였다.

표 7. Interpolation 방법 별 연산 복잡도[%]와 BDRate 저하율[%] 비교

Interpolation methods	computational complexity	BDRate degradation
7/8 tap filtering for sub-pels (HM)	100	—
Quadratic model for RD-cost	7.5	5.81
Bilinear filtering for SATD	10.1	4.88
Bicubic filtering for SATD	22.0	3.12
6 tap filtering for half-pels 2 tap filtering for quarter-pels	26.5	2.63
8 tap filtering for half-pels 2 tap filtering for quarter-pels	33.9	2.35

제 5 장 실험 결과

5.1 SATD 를 이용한 조건부 Bilinear Estimation

지금까지 앞 장에서 수행한 실험 결과들을 바탕으로 HM 인코더에서 FME 를 위한 interpolation 수행 시 연산 복잡도는 최대한으로 저감하는 동시에 BRate 성능 저하는 최소한으로 하기 위한 실험을 계획 하였다.

첫 번째로, fractional-pel interpolation 을 위한 approximation 시 이용하는 metric 의 측면에서 RD-cost 값 보다는 SATD 값을 이용하는 방법이 보다 효과적인 것으로 판단하였다. RD-cost 함수는 Distortion 과 rate 의 합으로 이루어져 있는데 서로 다른 fractional-pel 의 위치 별로 각기 다른 크기의 MV 의 값을 갖고 있는 rate 이 반영되어 있는 상태에서 이 값들을 이용하여 interpolation 결과를 approximation 하게 되면 에러율이 SATD 만을 이용하였을 경우보다 좀 더 크게 표출 되기 때문이다.

두 번째로, interpolation 을 approximation 하는 방법 측면에서 기존 HM 과 동일한 방식의 pixel-interpolation 의 경우 필터 탭의 개수를 적절히 줄이게 되면 연산량은 줄어 들면서 BRate 는 상대적으로 적게 저하되는 효과를 확인 하였지만 본 논문에서는 interpolation 의 연산량을 더 감소 시킬 수 있는 방안을 위해 bilinear interpolation 과 bicubic interpolation 의 적용을 고려하였다. 이 중 bilinear 모델의 경우 interpolation 하고자 하는 fractional-pel 을 중심으로 그 주변의 2 개 pixel 위치의 정보만을 이용하지만, bicubic 모델의 경우 주변의 4 개 pixel 위치의 정보를

활용해야 하므로 부가적인 연산량도 최대한 방지하기 위해 bilinear 모델을 우선으로 고려 하였다.

세 번째로, 한가지 방안만으로 interpolation 하기 보다는 주변의 조건을 적절히 탐색하여 제안한 interpolation 연산 간소화 방안이 적절하지 않다고 판단되면 HM 인코더에서 기존에 적용되어 있는 pixel-interpolation 을 수행하여 BDRate 저하를 최소한으로 하기 위한 방안을 고려하였다.

그림 27 은 SATD 를 metric 으로 하는 bilinear 모델을 사용하되 생성하고자 하는 fractional-pel 주변 pel 들의 roughness 를 판단하여 threshold 를 결정하고, 일정 수준 이하의 uniform 한 특성이 확인되면 approximated interpolation 을 수행하되 그렇지 않으면 기존의 pixel-interpolation 을 수행하도록 하여 연산 복잡도는 낮추고 BDRate 성능 저하는 최소한으로 하고자 하는 제안 알고리즘의 전체 flowchart 를 나타낸다.

그림의 Flowchart 에서 회색으로 칠해진 박스는 기존의 HM 인코더에서 ME 를 수행하는 부분을 나타내는 것으로 앞서 기술한 바와 같이 BDRate 성능과 interpolation 연산 복잡도는 trade-off 관계를 보이고 있음을 상기한다면, 제안 알고리즘에서 정하는 threshold 의 강도에 의해 BDRate 성능과 연산 복잡도 저감 사이에서 원하는 성능은 조절할 수 있도록 하였다. 즉, threshold 강도를 낮추어 pixel-interpolation 을 많이 수행토록 한다면 BDRate 성능 저하는 낮출 수 있는 대신 연산 복잡도 저감율도 낮아지게 되고 threshold 강도를 높여서 bilinear interpolation 을 많이 수행토록 하면 연산 복잡도 저감율은 높아지는 대신 BDRate 성능 저하도 높아지게 된다.

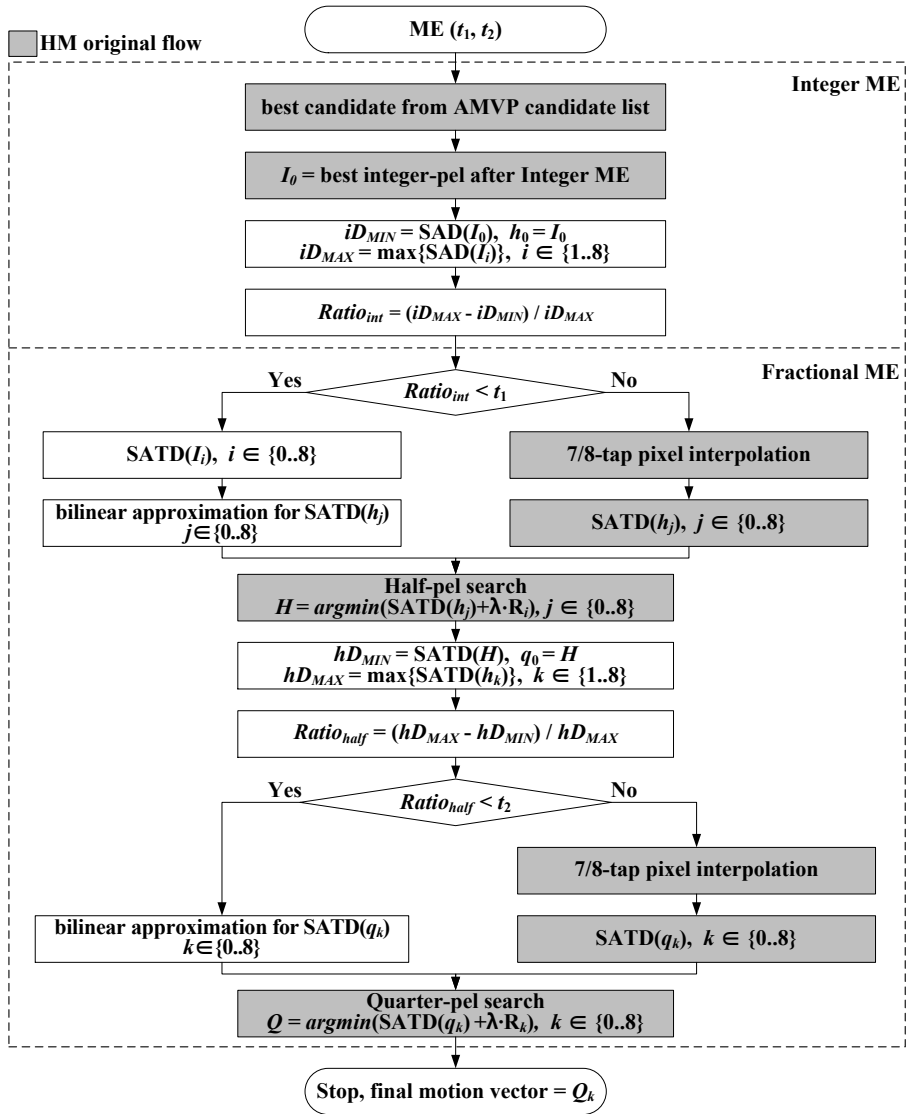


그림 27. Proposed FME algorithm using conditional bilinear SATD estimation with two thresholds

AMVP candidate list 에서 best candidate 가 선정이 되면 해당 위치의 움직임 벡터를 IME 수행의 초기 위치로 삼아 IME 를 수행한 뒤 최적의 integer-pel 이 선정 되면 해당 위치 (I_0)를 기준으로 FME 를 수행하게 되는데 IME 수행 후 최적 integer-

pel 을 찾는 시점에서는 uni-prediction 에서 test zone search 나 bi-prediction 에서 full search 를 통해 최적 integer-pel 주변의 8 개 정수 화소 위치에 대한 SAD 가 이미 계산되어 있는 상태이므로 해당 위치의 pixel 정보를 수집하여 bilinear interpolation 을 수행하기 위한 사전 동작을 수행한다. 최적 integer-pel 과 그 주변의 8 개 integer-pel 들의 위치는 그림 28 과 같다.

$I_{-1,-1}$		$I_{0,-1}$		$I_{1,-1}$
$I_{-1,0}$		$I_{0,0}$		$I_{1,0}$
$I_{-1,1}$		$I_{0,1}$		$I_{1,1}$

그림 28. IME 수행 후 결정 된 최적 integer-pel A0.0 와 bilinear interpolation 을 수행하기 위한 주변 8 개 integer-pels

총 9 개의 정수 화소에 대해 각 위치별 SAD 값을 확인 하여 이 중 최대값 과 최소값의 비율을 통해 $Ratio_{int}$ 를 확인하고 사전 실험 결과를 바탕으로 결정된 threshold t_l 을 기준으로 정해진 threshold 이하인 경우에는 SATD 이용한 bi-linear interpolation 을 수행하여 각 위치 별 half-pel 에 대한 distortion 을 구하고 아닌 경우에는 HM 과 동일하게 pel-interpolation 을 수행하게 된다. 여기서, HEVC 표준에 의거 IME 의 경우 SAD based distortion 을 기반으로 최적 integer-pel 을 찾는 동작이 수행 되지만, FME 의 경우 SATD based distortion 을 기반으로 최적 fractional-pel 을 찾는 동작이 수행되게 되므로 제안 알고리즘에서도 각 수행 단계별

distortion 함수를 동일하게 맞추어 표준을 준수하도록 하였다. 이럴 경우, 제안 알고리즘에서 최적 integer-pel 과 주변 8 개 integer-pel 의 distortion 값을 이용한 bilinear interpolation 을 사용하여 half-pel 을 생성하는 경우 IME 수행 시 계산된 각 integer-pel 의 SAD 값을 이용 하는 게 아닌 SATD 값을 이용해야 하므로 최적 integer-pel 과 그 주변 8 개 integer-pel 의 위치에 대하여 SATD 값을 별도로 구한 후 bilinear interpolation 동작을 수행해야 한다. 하지만, 이렇게 하더라도 overhead 로써 추가적인 연산 발생은 아니게 되는데 이것은, HM 에서도 최적 half-pel 탐색을 위한 half-pel interpolation 시 최적으로 결정된 integer-pel 위치와 그 주변 8 개 half-pel 위치에 대한 SATD 값을 계산하고 있는 것과 동일한 연산량 되기 때문이다.

Flowchart 의 각 단계별 수행 동작을 좀 더 구체적으로 열거하면 아래와 같이 10 단계의 동작으로 정리할 수 있다.

- Step 1. IME 수행 후 최적 integer-pel 과 주변 8 개 integer-pel 의 SAD 값 수집 후 총 9 개 SAD 에 대한 $Ratio_{int}$ 확인
- Step 2. $Ratio_{int}$ 가 사전 지정된 Threshold t_l 미만인 경우 Step 3 를 수행하고 이상인 경우 Step 4 를 수행
- Step 3. 최적 integer-pel 과 주변 8 개 integer-pel 위치에 대하여 각각 SATD 를 계산한 후 인접 pixel 간 SATD 를 bilinear interpolation 으로 8 개의 half-pel 을 생성 함. 생성된 SATD 에 rate 값을 더하여 각 위치 별 RD-cost 를 계산함

- Step 4. Pixel-interpolation 을 수행하여 half-pel 들을 생성하고 각 위치들에 대해 SATD 와 rate 값을 구하여 RD-cost 를 계산함.
- Step 5. 9 개 half-pel 들의 RD-cost 를 비교하여 최적의 RD-cost 위치를 찾아 best half-pel 로 선정 함.
- Step 6. 구해진 best half-pel 과 주변 8 개 half-pel 의 SATD 값에 대해 $Ratio_{half}$ 확인.
- Step 7. Step 2. $Ratio_{int}$ 가 사전 지정된 Threshold t_l 미만인 경우 Step 8 을 수행하고 이상인 경우 Step 9 를 수행
- Step 8. Best half-pel 과 그 주위의 pixel 들의 SATD 값을 bilinear interpolation 을 이용하여 8 개의 quarter-pel 을 생성 후 rate 값을 더하여 각 위치 별 RD-cost 를 계산함.
- Step 9. Pixel-interpolation 을 수행하여 quarter-pel 들을 생성하고 각 위치들에 대해 SATD 와 rate 값을 구하여 RD-cost 를 계산함.
- Step 10. 구해진 9 개 quarter-pel 들의 RD-cost 를 비교하여 최적의 RD-cost 위치를 찾아 best quarter-pel 로 선정 하고 FME 를 종료 함.

제안 알고리즘의 BDRate 와 interpolation 연산 복잡도에 대한 성능은 threshold t_1, t_2 의 수준에 의해 좌우 된다고 할 수 있는데 이 threshold 의 수준은 1080p class-B 시퀀스를 기준으로 한 그림 29 과 같이 IME 수행 후 발생한 $Ratio_{int}$ 와 half-pel search 후 발생한 $Ratio_{half}$ 를 바탕으로 결정되게 되므로 결국, 제안 알고리즘은 pixel 간 거리에 dependent 하게 linear 한 특성을 갖고

있는 이미지의 근본적인 성질을 이용한다고 할 수 있으며 이미지의 크기가 커지면 커질수록 이러한 pixel 간 linearity 특성이 잘 발현될 것이므로 FHD 및 UHD 등의 이미지 인코딩을 위한 architecture 에 적합하다고 할 수 있을 것이다.

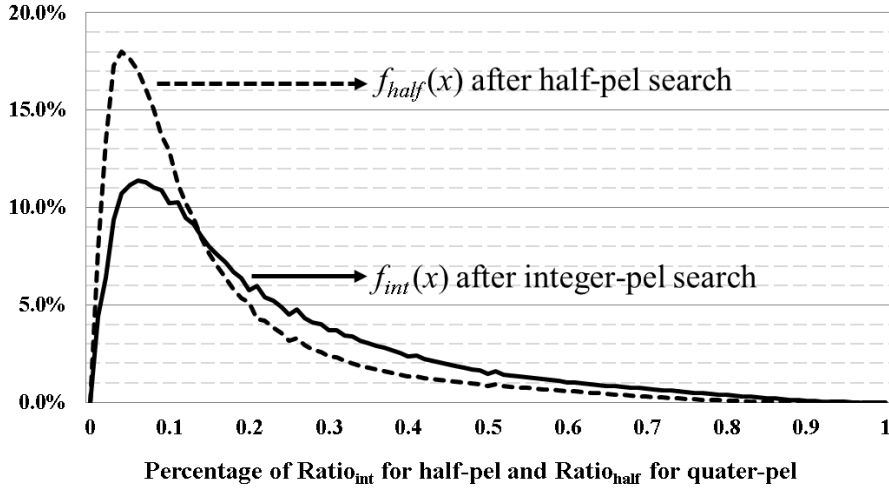


그림 29. $Ratio_{int}$ 및 $Ratio_{half}$ 에 대한 probability mass functions

적절한 threshold 를 결정하기 위해 threshold 조합 별로 BDRate 성능저하와 이때의 interpolation 연산 복잡도를 표 8 ~ 표 10 에 정리하여 보았다.

표 8 의 경우, 3840x2160(UHD) 크기의 Birthday, BirthdayFlash, Parakeets, 그리고 TableCar 의 4 개 시퀀스에 대한 평균 결과이고, 표 9 의 경우 HEVC 공통평가 조건의 2560x1600(WQXGA) class-A 시퀀스 Traffic, PeopleOnStreet, Nebuta, 그리고 StreamLocomotive 에 대한 평균 결과이며, 표 10 의 경우 HEVC 공통평가 조건에 정의된 1920x1080(FHD) 크기의 class-B 시퀀스 BasketballDrive, Cactus, Kimono, ParkScene, 그리고 BQTerrace 에 대한 평균 결과로 각 표에서

BDrate 의 경우 random access 설정에서 첫 32 frame 에 대한 결과로 각 threshold 조건 별 BDRate 저하율과 이때의 interpolation 연산 복잡도를 같이 표기 하였다. 표에서의 성능 결과는 HM 인코더의 결과를 anchor 로 하였으며 이때의 BDRate 는 0%, interpolation 연산 복잡도는 100%가 된다.

표 8. UHD 시퀀스에서 threshold 별 bilinear SATD estimation 적용 시 성능 변화 (BDRate[%], interpolation 연산복잡도[%])

$t_2 \backslash t_1$	0.0	0.1	0.2	0.3	0.4
0.0	(0.0,100)	(0.03,82.5)	(0.01,79.4)	(0.08,77.5)	(0.15,76.2)
0.1	(0.35,92.7)	(0.42,44.1)	(0.45,41.0)	(0.55,39.1)	(0.71,37.8)
0.2	(0.58,54.3)	(0.60,33.5)	(0.79,30.4)	(0.89,28.5)	(1.02,27.1)
0.3	(0.82,43.7)	(0.91,27.4)	(1.03,24.3)	(0.98,22.4)	(1.45,21.0)
0.4	(1.09,37.6)	(1.05,23.9)	(1.32,20.8)	(1.57,18.9)	(1.78,17.6)

표 9. WQXGA 시퀀스에서 threshold 별 bilinear SATD estimation 적용 시 성능 변화 (BDRate[%], interpolation 연산복잡도[%])

$t_2 \backslash t_1$	0.0	0.1	0.2	0.3	0.4
0.0	(0.0,100)	(-0.07,94.1)	(0.01,89.8)	(-0.03,86.3)	(0.03,83.2)
0.1	(0.06,99.9)	(0.09,63.2)	(0.09,58.9)	(0.16,55.4)	(0.22,52.5)
0.2	(0.19,69.0)	(0.21,47.6)	(0.26,43.4)	(0.37,39.9)	(0.47,36.9)
0.3	(0.40,53.5)	(0.42,37.9)	(0.50,33.6)	(0.65,30.1)	(0.76,27.2)
0.4	(0.71,43.7)	(0.73,31.5)	(0.83,27.2)	(0.93,23.7)	(1.17,20.8)

표 10. Full-HD 시퀀스에서 threshold 별 bilinear SATD estimation

적용 시 성능 변화 (BDRate[%], interpolation 연산복잡도[%])

$t_2 \backslash t_1$	0.0	0.1	0.2	0.3	0.4
0.0	(0.0,100)	(-0.05,91.7)	(-0.02,88.0)	(0.01,85.4)	(0.17,83.4)
0.1	(0.37,99.8)	(0.43,54.3)	(0.48,50.6)	(0.60,48.1)	(0.66,46.0)
0.2	(0.48,62.4)	(0.58,43.9)	(0.72,40.2)	(0.98,37.7)	(1.13,35.6)
0.3	(0.79,52.0)	(0.90,37.4)	(1.05,33.6)	(1.15,31.1)	(1.48,29.0)
0.4	(1.15,45.5)	(1.16,32.2)	(1.30,28.5)	(1.51,26.0)	(1.81,23.9)

앞 서 언급한 바와 같이 위 표에서 threshold 의 허용 범위에 따른 BDRate 저하율과 interpolation 연산량 저감량은 trade-off 관계를 갖고 있음을 확인할 수 있다. Threshold 수준은 높여서 bilinear interpolation 허용량을 증가시킴으로 하여 interpolation 연산을 간소화 할 수록 BDRate 성능은 점점 더 저하되고 threshold 수준을 낮출 경우 그 반대의 결과를 얻을 수 있었다. 또한, 그림 30 과 같이 인코딩 하려는 영상의 크기가 1920x1080(FHD) → 2560x1600(WQXGA) → 3840x2160(UHD)로 커질수록 interpolation 하고자 하는 주변 화소 들의 선형적인 특성은 낮은 레벨의 $Ratio_{SAD}$ 및 $Ratio_{SATD}$ 비율이 증가하는 트렌트를 통해 더욱 잘 드러남을 알 수 있는데, 주변의 선형적인 특성을 이용하는 bilinear estimation 을 적용 시 화질 열화량은 줄어들게 되어 영상의 크기가 커짐에 따라 제안 알고리즘에 이점이 더 발생하고 있음을 위의 표에서 확인 할 수 있다.

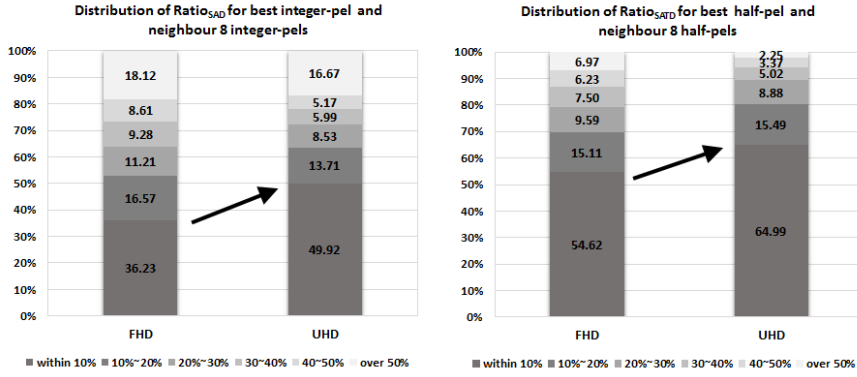


그림 30. Integer-pel 및 half-pel 해상도에서 영상 크기에 따른 주변 pixel 간의 linearity 특성 비교

위의 실험 결과를 종합하여 각 시퀀스 별 인코딩을 위한 interpolation 의 복잡도 C_{total} 은 threshold t_1 에 의해 결정되어 half-pel interpolation 시 발생하게 되는 $C_{half}(t_1)$ 과 t_2 에 의해 결정되어 quarter-pel interpolation 시 발생하게 되는 $C_{quarter}(t_2)$ 의 조합으로 식 (14)와 같이 표현할 수 있다.

$$C_{total}(t_1, t_2) = C_{half}(t_1) + C_{quarter}(t_2)$$

$$\approx C_{total}(0,0) \left\{ \alpha(1 - F_{int}(t_1)) + (1 - \alpha)(1 - F_{half}(t_2)) \right\} \quad (14)$$

여기서, $C_{half}(t_1)$ 및 $C_{quarter}(t_2)$ 는 각 half-pel search 와 quarter-pel search 단계 별 bilinear interpolation 과 pixel interpolation 연산량의 합으로 이루어져 있으므로 threshold 수준에 따른 비율 α 와 t_1 및 t_2 에 따른 probability mass function 인 $f(x)$ 로 나타낼 수 있는데, half-pel 에 대해서는 $F_{int}(t_1) = \int_0^{t_1} f_{int}(t_1)dt$ 이고, quarter-pel 에 대해서는 $F_{half}(t_2) = \int_0^{t_2} f_{half}(t_2)dt$ 로 나타낼 수

있다. 식에서 α 의 경우 $C_{half}(0)/C_{total}(0,0)$ 를 의미하며 이 값은 약 0.24 를 갖게 됨을 확인하였다.

5.2 선행 연구와 성능 비교

FME 에서의 연산 복잡도를 낮추기 위한 시도는 궁극적으로 FME 를 수행하면서 복잡 다단한 연산으로 인해 hardware 설계 시 pipeline 의 병목현상을 유발하여 throughput 을 저하시키고 hardware 측면에서도 많은 gate count 를 필요로 하기 때문에 cost 관점에서 불리하기 때문이다. 이를 위해 선행연구에서는 앞서 언급한 바와 같이 throughput 향상을 위해 resource 를 많이 투입하거나, 연산 방법을 단순화 하는 방법을 동원하여 빠른 연산을 목표로 하였다. Interpolation 연산 복잡도(IC)와 FME 를 위한 hardware 복잡도(HC) 및 BD-rate 성능 저하(BDBR)에 대해 HEVC 공통 평가조건에서 정의된 1080p class-B 5 개 시퀀스의 random access 설정에서 각 32frames 의 평균 실험 결과로 표 11 에 비교 정리 하였다. Interpolation 연산 복잡도(IC)는 4.1 절에서 기술한 interpolation 연산 복잡도 계산 방법에 근거하여 64x64 크기의 CTU 한 개의 interpolation 연산 하기 위한 연산량을 비교하였다. Hardware 복잡도(HC)의 경우는 4.2 절에서 기술한 hardware 복잡도 계산 방법에 근거하였으며 역시 CTU 한 개의 FME 연산을 위해 필요한 cycle 수에 대해 비교 정리하였다. 이때, Hardware implementation 은 앞서 기술한 바와 같이 3840x2160 크기의 UHD 영상을 400Mhz CLK frequency 로 30fps 를 처리하기 위한 구조로 가정하였다.

표 11. Interpolation 방법 별 IC[%]와 HC[%] 및 BDBR[%]비교

Algorithms	Interpolation method	IC	HC	BDBR
Tsung [7]	7/8-tap filtering for sub-pels	114.8	147.1	0.09
He [8]	8-tap filtering for half-pels 2-tap filtering for quarter-pels	33.9	53.6	2.35
Jou [9]	quadratic model for RD costs	14.8	36.1	2.95
This work	bilinear filtering for SATD	31.1	49.2	1.15

Tsung[7]의 방법은 fractional-pel search 시 HM 과 같이 half-pel 후 quarter-pel 을 수행하는 2 단계로 나누어 하지 않기 때문에 interpolation 도 half-pel 과 quarter-pel 을 동시에 생성하는 방법으로 interpolation 필터 탭 수는 HM 과 동일하게 유지하게 되므로 BD-rate 저하는 거의 없는 대신, resource 는 많이 필요하게 되어 연산 복잡도 측면에서는 HM 보다 오히려 열세인 특성을 갖는다. [7]의 연구를 보완하여 He[8]의 경우 동일한 방식으로 half-pel 과 quarter-pel 을 동시에 생성 하되, interpolation 필터 탭 수를 적절히 조절하여 BD-rate 성능은 약간 손해를 보았지만 연산 복잡도는 비약적으로 개선할 수 있는 방안을 제시하였다. Jou[9]의 경우, 기존의 interpolation 과는 완전히 다른 개념을 사용하였는데 pixel interpolation 이 아닌 RD-cost interpolation 기법을 도입하여 복잡 다단한 matrix 연산을 단순한 scalar 연산만으로 수행 할 수 있게 되어 interpolation 의 연산 복잡도는 매우 간소화 되었지만 너무 단순한 연산 기법으로 인해 BD-rate 저하가 매우 심하게 되었다. 본 논문에서는 BD-rate 저하는 최소화 하면서 interpolation 연산 복잡도는 최대한으로

개선하기 위해 threshold 를 도입하였고 이를 이용하여 생성하고자 하는 fractional-pel 들의 주변 상황이 간소한 연산이 가능한 경우 bilinear 모델을 적용하여 scalar 연산만으로 interpolation 을 수행토록 하였다. 이를 통해 연산 복잡도는 선행 연구와 동등한 수준으로 개선하면서 BD-rate 성능은 He[8] 보다는 약 57% 정도 더 향상 시키는 결과를 얻을 수 있었다.

제 6 장 결 론

본 논문에서는 HM 인코더에서 연산 복잡도가 제일 높은 inter prediction 에서의 연산 복잡도를 낮출 수 있는 방안을 연구 하였다. Inter prediction 중에서도 단일 모듈로는 제일 연산량이 많은 FME 에서의 연산 복잡도를 효과적으로 낮출 수 있는 방안을 중점적으로 살펴 보았으며 기존의 연구들은 다소 단순한 방법을 통해 연산을 간소화 하여 side effect 로 BDRate 성능 저하를 심하게 발생 시키거나, hardware cost 를 많이 써서 BDRate 성능은 보전을 하되 연산을 빨리 하는 방법으로 latency 측면으로 접근하여 연산 복잡도와 BD-rate 성능 간에 한쪽으로 치우치는 경향을 보였다면 본 논문에서는 BDRate 성능도 고려하면서 연산 복잡도에 대해서도 선행 연구들과 대등한 수준으로 개선할 수 있는 방안을 제시 하였다. IME 수행 후 이미 계산 된 pixel 정보를 이용하여 half-pel interpolation 연산을 간략화 한 뒤, 다시 half-pel search 후 수집한 주변 pixel 들의 정보를 이용하여 quarter-pel interpolation 의 계산을 간소화 하는 방법을 통해 HM 과 비교하여 전체 interpolation 의 연산을 약 69%를 절약할 수 있었으며 이때 BDRate 저하도 약 1.1% 정도로 제한할 수 있었다. 또한 제안 알고리즘은 다른 연구들과는 달리 BD-rate 성능과 interpolation 연산 복잡도 및 hardware 복잡도의 trade-off 관계를 통해 쉽게 목표 성능을 변경할 수 있는데 이것은 threshold 기법을 도입함으로 하여 가능하게 된 것으로 따라서 제안 알고리즘은 hardware 인코더뿐만 아니라 software 인코더에도 적용이 가능한 방안임에 의의가 있다. 또한, 제안 알고리즘은 영상의 크기가

커질수록 더 좋은 성능 결과를 보이는 방식이므로 방송 영상 서비스가 FHD 에서 UHD 로 급격히 전환되는 현재의 영상 산업 환경에서 더욱 의미를 갖는다 하겠다.

참고 문헌

- [1] G. Correa, P. Assuncao, L. Agostini, and L. Cruz, "Coding tree depth estimation for complexity reduction of HEVC," Data Compression Conference (DCC), 2013, pp. 43–52.
- [2] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H. 264/AVC video coding standard." Circuits and Systems for Video Technology, vol.13, no7, 2003, pp 560–576.
- [3] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," . Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 1649–1668, Dec. 2012
- [4] L. Lu, J. V. McCanny, and S. Sezer, "Subpixel interpolation architecture for multistandard video motion estimation," Circuits and Systems for Video Technology., vol. 19, no. 12, pp. 1897–1901, Dec. 2009.
- [5] V. Afonso, H. Maich, L. Agostini, and D. Franco. "Low cost and high throughput FME interpolation for the HEVC emerging video coding standard," 2013 IEEE Fourth Latin American Symposium Circuits and Systems (LASCAS), pp. 1–4, Feb. 2013.
- [6] H. Lv, R. Wang, X. Xie, H. Jia, and W. Gao, "A comparison of fractional-pel interpolation filters in HEVC and H. 264/AVC." Visual Communications and Image Processing (VCIP), 2012.

- [7] P. K. Tsung, W. Y. Chen, L. F. Ding, C. Y. Tsai, T. D. Chuang, and L. G. Chen. "Single-iteration full-search fractional motion estimation for quad full HD H. 264/AVC encoding," 2009 International Conference Multimedia and Expo (ICME), pp. 9-12, June 2009.
- [8] G. He, D. Zhou, Y. Li, Z. Chen, T. Zhang, and S. Goto. "High-Throughput Power-Efficient VLSI Architecture of Fractional Motion Estimation for Ultra-HD HEVC Video Encoding," Very large scale integration (VLSI) Systems, Vol 23, pp. 3138-3142, Dec 2015.
- [9] S. -Y. Jou, C. Silvia, and C. T.-H. Chang. "Fast Motion Estimation Algorithm And Design For Real Time QFHD High Efficiency Video Coding," Circuits and Systems for Video Technology, Vol 25, pp.1533-1544, Sep. 2015.
- [10] J. -F. Chang, and J. -J. Leou. "A quadratic prediction based fractional-pixel motion estimation algorithm for H. 264," Journal of Visual Communication and Image Representation, Vol 17, pp. 1074-1089, Oct. 2006.
- [11] S. Srinivasan, "Approximate bicubic filter." U.S. Patent No. 7,110,459. 19, Sep. 2006.
- [12] J. -B. Lee, and K. Hari, "The VC-1 and H. 264 video compression standards for broadband video services." Vol. 32. Springer Science & Business Media, pp293-297, 2008.
- [13] N. Purnachand, L. N. Alves, and A. Navarro, "Improvements to TZ search motion estimation algorithm for multiview video

coding," 2012 19th International Conference on Systems, Signals and Image Processing (IWSSIP), pp. 388–391, 2012.

[14] T.-C. Chen, Y.-W. Huang, and L.-G. Chen, "Fully utilized and reusable architecture for fractional motion estimation of H. 264/AVC," in Acoustics, Speech, and Signal Processing (ICASSP'04), pp. V-9–12 vol. 5, 2004

[15] J. Lee, S. Kim, K. Lim, and S. Lee, "A Fast CU Size Decision Algorithm for HEVC." Circuits and Systems for Video Technology, vol. 25, no.3, pp. 411–421, 2015.

[16] V. Sze, B. Madhukar, and G. J. Sullivan. "High Efficiency Video Coding (HEVC)." Springer, pp. 135 – 137, 2014

[17] F. Bossen, "Common test conditions and software configurations," JCT-VC-L1100, JCT-VC, Geneva, Switzerland, Jan. 2013.

Abstract

A Complexity Reduction Algorithm of the Fractional Motion Estimation in HEVC using Conditional Bilinear SATD Estimation

Dae-Bin Lim

Dept. of Electrical and Computer Engineering

The Graduate School

Seoul National University

The process of the inter prediction in the HEVC encoders finds the best partition mode for the each coding tree unit (CTU) and prediction for each prediction blocks (PU) in the CTU and then, encodes the residual data which is the difference between the original PU and its best prediction. To find the best prediction and its motion vector from previously reconstructed pictures, motion estimation which is consist of integer motion estimation (IME) and fractional motion estimation (FME) is performed. In the HEVC encoder, IME selects the best integer motion vector (IMV) through the test zone search algorithm and FME also finds the best quarter-pel motion vector (FMV) after obtaining predicted PU for the fractional positions with 7/8 tap interpolation.

The coding efficiency of the HEVC is improved supporting 7/8-tap interpolation compared to the previous standard of H.264/AVC that uses 6/2-tap interpolation for sub-pel prediction. However, the computational complexity of the FME is substantially increased due to the repeated interpolation for each PU partitions of every coding units (CU) in all coding tree units (CTU) with all reference pictures. Therefore, it is necessary to reduce the complexity for FME without substantial coding efficiency degradation in the HEVC encoders.

In this paper, we propose to estimate SATD values with bilinear filtering for eight search positions of a PU only if the SATD values for the neighbors of the motion vector after the integer or half-pel search are smooth enough. According to the simulation results, the proposed algorithm reduces the computational complexity for interpolation by about 69% on the average and incurs 1.15% BD-rate degradation for the five class-B video sequences.

Keywords : HEVC, Inter Prediction, Fractional Motion
Estimation, Complexity reduction, Interpolation
Student Number : 2014-21602