HEVC 의 SAO 병렬화 성능 비교

조현호, 심동규
광운대학교
idjhh@kw.ac.kr, dgsim@kw.ac.kr

Comparison of Parallelization for HEVC SAO

Hyunho Jo, Donggyu Sim
Kwangwoon University

요 약

본 논문에서는 HEVC (High Efficiency Video Coding) SAO (Sample Adaptive Offset)의 병렬화 성능을 비교한다. HEVC 의 참조 소프트웨어인 HM-10.0 에서는 SAO 수행 과정의 연산량 및 메모리 접근을 최소화하고 카테고리 계산 과정에서 SAO 수행 전의 픽셀값을 사용하기 위해서 라인 버퍼를 사용한다. 그러나 이러한 라인 버퍼의 사용은 SAO 에 대해 데이터-레벨의 병렬화를 적용하기 어렵게 만드는 주요 요인이다. 본 논문에서는 HEVC 디블록킹 필터가 적용된 픽쳐를 추가 메모리에 복사하는 구현 방식과 HM-10.0 의 SAO 구현 방식 각각에 대해 데이터-레벨 병렬화를 적용하고 각각의 성능을 비교·분석하였다. 실험 결과, HEVC 디블록킹 필터가 적용된 픽쳐를 추가 메모리에 복사하는 구현 방식은 데이터-레벨 병렬화의 구현은 쉽지만, 디블록킹 필터링 된 픽쳐를 추가 메모리에 복사하는 부분 때문에 HM-10.0 기반의 병렬화보다 복호화 성능이 저하될 수 있음을 확인하였다. 이에 반해 CTU 의 행 단위로 병렬 수행될 영역을 분할하는 방식은 구현의 용이성과 병렬화 성능을 동시에 얻을 수 있음을 확인하였다.

1. 서론

ISO/IEC 의 MPEG (Moving Picture Expert Group)과 ITU-T 의 VCEG (Video Coding Expert Group)은 2013 년 1 월 차세대 비디오 압축 표준 기술인 HEVC (High Efficiency Video Coding)의 표준 제정을 완료하였다. HEVC 는 H.264/AVC 와 같은 이전의 동영상 압축 표준 기술과 동일하게 예측 (Prediction), 변환 (Transform), 양자화 (Quantization), 엔트로피 코딩 (Entropy coding)이라는 기본 구조를 그대로 사용한다. 그러나 HEVC 는 각각의 코딩 단위에서 세부적으로 새로운 기술들을 사용함으로써 H.264/AVC High 프로파일보다 약 50%의 비트율을 감소시킨다. 그뿐만 아니라 HEVC 는 복원된 영상에서 양자화 에러 때문에 발생하는 블록킹 현상 (Blocking artifact)과 링잉 현상 (Ringing artifact)를 제거하기 위하여 디블록킹 필터와 SAO (Sample Adaptive Offset)라는 두 개의 인-루프 (In-loop) 필터를 사용한다. 그 중 SAO 는 디블록킹 필터가 H.264/AVC 에 사용되었던 것과 달리 HEVC 에 처음으로 소개된 기술이다. SAO 는 링잉 현상을 효과적으로 제거하여 주관적 화질을 향상 시키면 뿐만 아니라 SAO 가 적용된 픽쳐를 화면 간 예측 모드에서 참조 픽쳐로 사용하게 함으로써 부호화 성능도 향상 시킨다. 그러나 이러한 HEVC 인-루프 필터의 사용은 복호화기의 연산량을 증가시키고, 복원 픽쳐의 픽셀에 오프셋값을 더하는 SAO 의 특성상 메모리 접근에 의한 속도 저하를 일으킨다.

본 논문에서는 HEVC 의 SAO 에 데이터-레벨 병렬화 (Data-level parallelism)을 적용하여 복호화기에서 SAO 의 수행 속도를 향상시키는 방법을 소개한다. 특히, HEVC 의 참조 소프트웨어인 HM (HEVC Test Model)의 SAO 구현 방식에 데이터-레벨 병렬화를 적용하는 방식과 디블록킹 필터가 적용된 픽쳐를 추가 메모리에 복사하여 데이터 의존성을 제거한 후 데이터-레벨 병렬화를 적용하는 방식 각각에 대한 병렬화 성능을 비교·분석한다.

본 논문의 구성은 다음과 같다. 2 장에서는 HM-10.0 에 구현된 SAO 에 데이터 레벨 병렬화를 적용하는 방식과 디블록킹 필터가 적용된 픽쳐를 추가 메모리에 복사한 후 데이터 레벨 병렬화를 적용하는 방식에 대해서 자세히 설명한다. 3 장에서는 HEVC SAO 의 구현 방식에 따른 병렬화의 성능을 평가 및 분석하고, 4 장에서는 본 논문에 대한 결론 및 향후 연구 방향을 제시하고 본 논문을 마치도록 한다.

2. HEVC SAO 의 구현 방식에 따른 병렬화

HEVC 의 SAO 는 CTU 단위로 수행되며, 한 CTU 내에서 하나의 휘도 성분과 두 개의 색차 성분에 대해서 독립적으로 SAO 기술이 적용된다. HEVC 의 SAO 에는 밴드 오프셋과 에지 오프셋이 있는데 이중 에지 오프셋은 각 픽셀 단위에서 카테고리 (Category)를 계산한다. 카테고리는 그림 1 과 같이 CTB (Coding Tree Block) 단위로 부호화기가 복호화기로 명시적으로 알려주는 클래스의 값에 따라 선택되는 픽셀값들로부터 계산된다. 그림 1 에서 c 는 CTB 내에서 카테고리가 계산될 픽셀이고, a 와 b 는 c 픽셀의 주변에 있는 픽셀이다. 카테고리 값은 CTB 내의 모든 픽셀 위치에서 계산되는데 이때 a, b, c 값은 SAO 가 적용되기 전의 픽셀값들이 사용된다. 따라서 디블록킹 필터가 적용된 픽쳐를 임의의 영역으로 분할한 후 각 분할 영역에서 병렬적으로 SAO 를 수행하는 경우, 분할 영역 경계에서 데이터 의존성

문제가 발생한다.

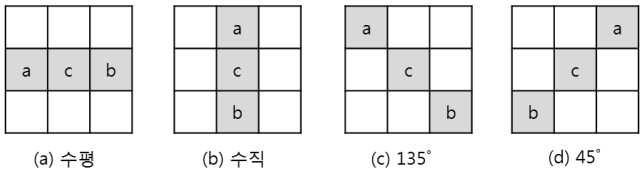

그림 1. 에지 오프셋의 클래스

병렬화를 위해 코딩할 픽쳐를 CTB 의 개수가 같도록 여러 영역으로 분할하는 경우, 각 영역 사이의 수평 경계와 수직 경계지점에서 데이터 의존성 문제가 발생한다. 본 논문에서는 픽쳐 내에서 CTB 의 행의 개수를 사용하여 영역을 분할하는 방법을 제안한다. 제안하는 영역 분할 방식 (방식-1)에서는 그림 2 와 같이 행 단위의 라인 버퍼만을 사용함으로써 데이터 의존성을 더 쉽게 해결할 수 있다. 또 다른 제안 방식 (방식-2)은 그림 3 과 같이 디블록킹 필터가 적용된 픽쳐를 추가 메모리에 복사하는 방식이다. 이러한 구현 방식은 라인 버퍼를 사용하는 것보다 메모리를 많이 사용하지만 에지 오프셋 과정에서 발생하는 데이터 의존성을 쉽게 해결할 수 있다. 또한, 각 CTU 에 대해서 SAO 를 수행할 때에도 라인 버퍼에 픽셀값들을 백업하는 부분을 제거할 수 있어 소프트웨어 최적화 관점에서 더 유리하다.

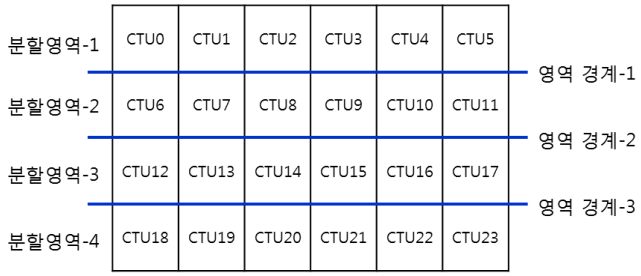

그림 2. CTB 의 행 단위로 영역을 분할하는 방식

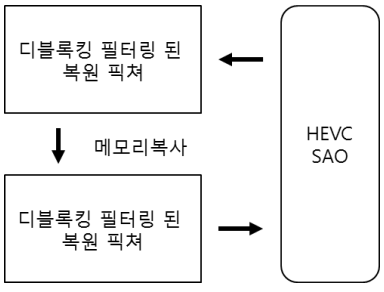

그림 3. 픽쳐 메모리 복사를 통한 HEVC SAO 구현 방식

3. 실험 결과

HEVC SAO 의 구현 방식에 따른 병렬화 성능을 비교하기 위하여 HM-10.0 기반에서 방식-1 과 방식-2 를 각각 구현하였다. 병렬화의 구현은 OpenMP (Open Multi-Processing)를 사용했으며 디코더의 성능은 HEVC SAO 의 수행 시간을 측정하였다. 실험 영상은 공통 실험 조건 (Common test condition)에서 사용하는 Full-HD 실험 영상 2 개를 사용했으며, 4 개의 QP 값을 사용하여 다양한 조건에서 성능을 비교하였다. 병렬화의 성능은 헥사 코어 CPU 상에서 평가되었다. 표 1 은 SAO 의 구현 방식에 따른 각각의 병렬화 성능을 순차적으로 수행되는 SAO 의 성능과 비교한 것이다. 표 1 을 참조하면 방식-1 은 순차수행 방식보다 약 2.25 배 수행 속도가 향상되지만, 방식-2 는 병렬 수행을 했음에도 순차수행보다 느려지게 된다. 이는 방식-2 가 픽쳐 단위의 메모리 복사를 수행하는데 해당 과정에서 많은 시간이 소요되기 때문이다.

표 1. 구현 방식에 따른 SAO 수행 시간 비교

실험 영상	QP	SAO 수행 시간 (sec)		
		순차수행	방식-1	방식-2
Cactus	22	2.64	0.91	1.99
	27	1.46	0.53	1.59
	32	0.68	0.45	1.54
	37	0.42	0.31	1.33
BQTerrace	22	3.04	0.96	2.22
	27	2.62	0.93	2.21
	32	1.06	0.48	1.73
	37	0.72	0.55	1.58

4. 결론

본 논문에서는 HEVC SAO 의 구현 방식에 따른 병렬화 성능을 비교하였다. 디블록킹 필터링 된 픽쳐를 메모리 복사하여 수행하는 방식은 병렬화의 구현은 용이하나 픽쳐 단위의 메모리 복사 때문에 QP 가 높은 영상에 대해서는 순차 수행 방식보다 성능이 저하된다. 이에 반해 CTB 의 행 단위로 영역을 분할하는 방식은 구현이 쉽고, 병렬화를 통한 성능 향상을 얻을 수 있다. 추후에는 SAO 수행 과정을 픽셀 디코딩과 디블록킹 필터링 과정을 고려하여 병렬화하는 방법을 연구할 것이다.

감사의 글

이 논문의 일부는 2012 년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (NRF-2010-0025325). 본 연구의 일부는 미래창조과학부 및 정보통신산업진흥원의 대학 IT 연구센터 육성지원 사업의 연구결과로 수행되었음 (NIPA-2013-H0301-13-1011)

참고문헌

[1] G. J. Sullivan, J.-R. Ohm, W. Han, and T. Wiegand, “ Overview of the high efficiency video coding (HEVC) standard,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649-1668, Dec. 2012.

[2] C. Fu, E. Alshina, A. Alshin, Y. Huang, C. Chen, C. Tsai, C. Hsu, S. Lei, J. Park, and W. Han, “ Sample adaptive offset in the HEVC standard,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1755-1764, Dec. 2012.