선인출 메커니즘을 이용한 효율적인 H.264/AVC 복호화 기법

지신행° 박정욱 김신덕
연세대학교 컴퓨터과학과
{twoze°, pjppp, sdkim}@yonsei.ac.kr

An Efficient H.264/AVC Decoding Technique Using Prefetching Mechanism

Shinhaeng Ji° Jungwook Park. Shindug Kim.
Department of Computer Science, Yonsei University.

요 약

H.264/AVC는 SoC/IEC MPEG와 ITU-T Video Coding Experts Group에서 함께 발표한 비디오 코딩을 위한 가장 최근의 표준이다. 기존의 표준들 보다 적은 비트로 높은 압축률과 좋은 화질을 제공하고 있다. 그러나 1/4 화소 움직임 예측과 보상의 지원과 7가지의 가변블록에 대한 움직임예측과 블록모드별 RD(Rate-Distortion)를 수행하고 CAVLC등 H.264/AVC 표준에서 채택한 여러 가지 비디오 압축방식으로 인해 그 복잡도가 훨씬 증가하였다. 이 논문에서 H.264/AVC의 복호화기에서 복잡도의 약 40% 이상을 차지하는 움직임보상 모듈을 효율적으로 수행하고 최적화하기 위한 방법을 제안한다. 예측된 모션벡터에 따라 참조하는 프레임에서 매크로블록을 만들어 내는 움직임 보상 과정을 수행하는 데 있어서 접근 지연시간이 큰 외부 메모리 참조를 선인출 메커니즘을 이용하여 미리 예측하여 수행함으로써 전체 수행시간을 줄이는 기법을 적용하였다. 이를 통하여 가변길이 복호화 모듈과 움직임 보상모듈을 수정하여 반복적으로 읽고, 쓰기를 수행해야 하는 횟수를 줄였다. 실험 결과 본 논문에서 제안하는 방법을 이용하여 복호화 과정을 수행했을 때 PSNR(Peak Signal to Noise Ratio)의 손실은 전혀 없으면서 복호화기의 전체 실행시간을 약 5% 향상시키고, 핵심 모듈인 움직임 보상과정에서 약 20% 정도 실행시간을 향상시키는 등 높은 성능 향상을 보였다.

1. 서 론

H.264/AVC는 ISO/IEC Moving Picture Experts Group와 the ITU-T Video Coding Experts Group에서 함께 발표한 비디오 코딩을 위한 가장 최근의 표준이다. MPEG-4나 H.263+ 등 기존의 표준들 보다 적은 비트로 높은 압축률과 좋은 화질을 제공하고 있다. 그러나 1/4 화소 ME(Motion Estimation: 움직임 예측)와 MC(Motion Compensation: 움직임 보상)의 지원과 7가지의 가변블록에 대한 ME와 블록모드별 Rate-Distortion[1]을 수행하고 CAVLC(Context-based Adaptive Variable Length Coding: 컨텍스트 적응형 가변길이 부호화기) 등 H.264/AVC 표준에서 채택한 여러 가지 비디오 압축방식으로 인해 그 복잡도가 훨씬 증가하였다[2][3][4]. 또한 16X16 크기의 매크로블록 단위로 복호화가 이루어지는 H.264/AVC는, 각 매크로 블록에 대응하여 재구성되어질 서브 매크로블록들을 외부 메모리로부터 반복적으로 읽어 들이는 과정이 수행된다. 이렇게 각 움직임 벡터에 해당하는 서브 매크로블록을 외부 메모리에서 찾아서 읽어 들이는 부분이 움직임 보상 모듈에서 수행되고 이 과정이 전체 복호화기에서 가장 많은 부하가 요구된다.

이 논문에서는 이러한 움직임 보상과정에서의 메모리 참조에 따른 문제점을 줄이기 위해 가변길이복호화기와 움직임 보상과정에서 선인출 메커니즘을 이용하여 프레임 재구성에 필요로 하는 서브 매크로블록을 미리 시스템에 갖다 둠으로써 전체 복호화기의 실행시간을 줄이고 외부 메모리에 대한 접근 횟수와 읽기의 부하를 줄이는 데 사용되었다.

본 논문의 구성은 다음과 같다. 2장에서는 디코더에서의 메모리에 대한 연구들과 기존의 디코더에서 메모리 참조 기법에 대해 설명하고, 3장에서는 보다 효율적인 복호화를 위한 새로운 기법을 제안한다. 4장에서는 새로운 기법을 H.264의 JM에 적용하여 대표적인 여러 영상에 대해 실험해 보고 기존의 디코더와 제안하는 디코더의 전체 실행시간과 핵심 모듈에서의 실행 시간(Clockticks)을 Intel VTune Performance Analyzer 프로그램을 사용하여 비교 분석하였다. 마지막으로 5장에서는 결론을 기술하였다.

2. H.264/AVC 복호기

H.264/AVC의 복호화기는 세 가지 단계, 즉 VLD(Variable Length Decoder), IDCT(Inverse Discrete Cosine Transform), MC(Motion Compensation)로 나뉜다.

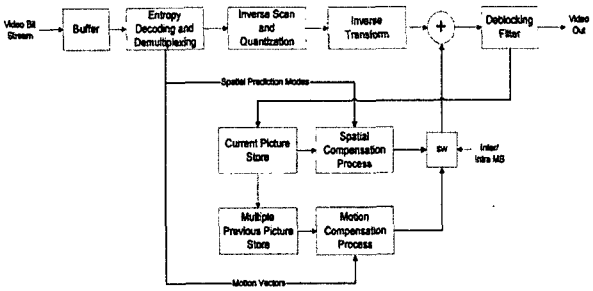


그림1. H.264/AVC 복호화기 구조

VLD(가변길이 복호화기) 단계에서는 NAL 단위로 들어오는 비디오 스트림의 부호화된 데이터를 복호화 하는 단계이다. 이때 움직임벡터 차이 (Motion Vector Difference : 'mvd') 값, 매크로블록 타입('mb_type'), 매크로블록 예측모드('mb_pred'), 양자화 파라미터('mb_qp_delta')와 예측 후의 오차 이미지 샘플에 해당하는 코딩된 변환 계수(Residual) 등의 값들을 얻게 된다. IDCT 과정에서는 VLD에서 구한 나머지 값들을 이용하여 비디오 압축 알고리즘에서 양자화 및 압축이전에 이미지 또는 오차 데이터의 상관관계를 제거하기 위해 사용된다. 마지막으로 움직임보상 단계에서는 VLD과정에서 구한 mvd와 mb_type 등의 정보를 가지고 움직임 벡터 예측(Motion Vector Prediction) 과정을 거친 후 구해진 서브 파티션에 대한 움직임 벡터를 가지고 외부 메모리로부터 데이터를 가져오게 된다. H.264/AVC에서는 이러한 동작들이 4x4 서브 매크로블럭 단위로 이루어진다. 이러한 외부 메모리 참조 동작으로 인해 복호화기의 핵심 모듈인 움직임 보상(MC)과정에 많은 부하가 생기게 된다.

3. 선인출 메커니즘을 이용한 복호기

JM 9.5에서 구현된 복호화기는 각 매크로 블록(16x16) 단위로 VLD 과정을 수행한다. 매크로블록의 헤더를 해석하여 복호화기 수행에 필요한 데이터들을 구하고 필요에 따라서 IQ와 IDCT를 수행한다. VLD과정을 마치고 나면 VLD에서 구한 'mvd', 'mb_type', 'ref_index' 등의 정보를 가지고 움직임 보상단계에서 각 서브 매크로블록에 대한 움직임 값(MV)을 구하고 이를 가지고 외부 메모리를 참조하게 된다.

이 논문에서 제안하는 기법은 VLD과정에서 구해지는 변환계수들과 MC과정을 수행하면서 계산되는 움직임 벡터 예측 값 또는 픽셀 값들을 저장하기 위해 사용하는 임시버퍼의 사용을 최소화하고 이들 데이터를 프레임 버퍼에 직접 저장하게 함으로써 메모리 부하를 줄이고 보다 빠르게 동작하도록 구현 하였다.

표1. 기존의 알고리즘와 제안하는 알고리즘 비교

기존의 알고리즘	제안하는 알고리즘
For each picture	For each picture
For each Slice	For each Slice
For each MB	For each MB
Decode MB header	Decode MB header
If a block is coded	Do mv_prediction
Decode and IQ	Prefetch sub-MBs
IDCT the block	If a block is coded
Motion Compensation	Decode and IQ
Do mv_prediction	IDCT the block
While(mb_type)	Motion Compensation
Accesses External Mem	
Load sub-MBs	

또한 JM 9.5[5]에서 구현된 VLD와 MC모듈을 수정하여 움직임 보상과정에서 참조하게 될 각 서브 매크로블록에 해당하는 움직임 벡터를 VLD과정에서 미리 계산하여 구하고, 동시에 해당 데이터를 미리 내부메모리에 가져다 놓도록 구현하였다. 이때 외부 메모리로부터 서브 매크로블록들을 반복적으로 읽어오는 과정을 프로세서에서 지원하는 명령을 이용하여 한 사이클에 읽어 오도록 하였다. 결국 움직임 보상과정에서 반복적인 메모리 참조에 따른 부하를 줄일 수 있었다.

표1은 JM 9.5에서 구현된 복호화 방법과 본 논문에서 제안하는 방법을 나타내고 있다.

4. 성능분석

4.1. 실험 방법

본 실험은 H.264/AVC의 베이스라인 프로파일에서 참조 소프트웨어인 Joint Model(JM) 9.5를 이용하여 수행되었다. 실험은 펜티엄 IV-2.6GHz PC에서 salesman과 carphone, container, slient, Mother & Daughter 그리고 claire QCIF(176X144) 영상에 대해 수행되었다.

본 논문에서는 기존의 디코딩 과정에서의 메모리 참조 기법과 제안하는 기법을 사용해 각 영상을 디코딩하는 시간과 메모리 참조와 관련된 핵심 모듈의 실제 실행 사이클을 비교하였다.

4.2. 실험 결과 및 분석

입력 영상을 디코딩하는 시간과 메모리 참조와 관련된 핵심 모듈의 실제 실행 사이클을 측정하고 이를 평균화하여 기존의 복호화기와 본 논문에서 제안한 선인출 기법을 이용한 복호화기의 성능을 비교 분석하였다.

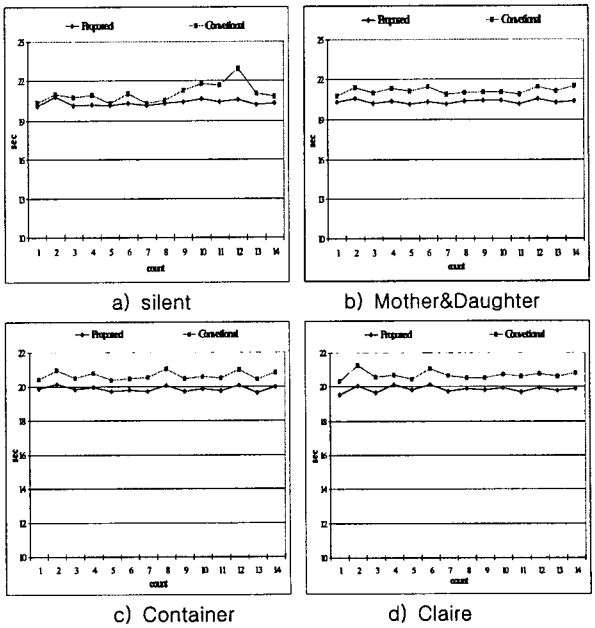
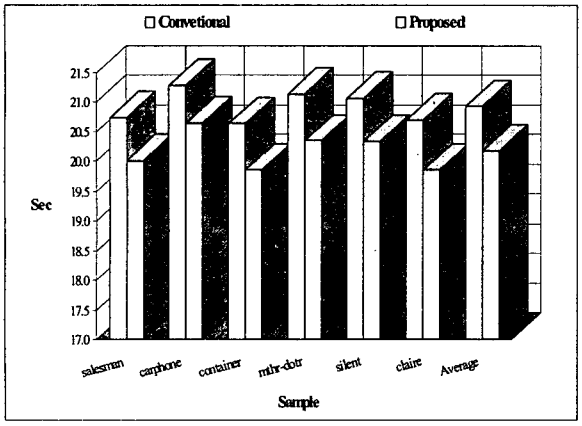
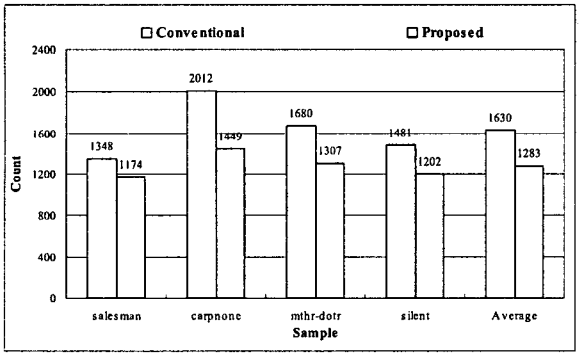


그림2. 비디오 샘플(Silent, Mother&Daughter, Container, Claire)에 대한 평균 복호화 시간비교

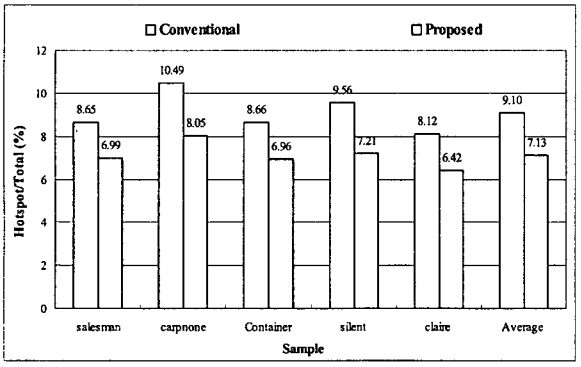
그림2는 입력 영상들에 대한 핵심 모듈의 복호화 시간을 VTune Performance Analyzer을 사용하여 측정한 결과를 보여 주고 있다. 각 영상에 대한 결과를 보면 기존의 복호기보다 약 5%정도씩 성능이 향상 된 것을 볼 수 있다. 전체 복호기의 실행시간에 대해서도 알아보자. 그림3은 실험에서 나온 결과를 도표화 한 것이다.



a) H.264/AVC 복호기 전체 실행시간



b) H.264/AVC 핵심 모듈에 대한 Clockticks 비교



c) H.264/AVC 핵심 모듈에 대한 실행시간 비교

그림3. 입력영상들에 대한 복호기 전체실행 시간(a), 핵심모듈에 대한 클럭 수(b)와 실제 실행시간(c) 비교

그림3은 H.264/AVC의 핵심 모듈을 실행하는데 걸린 시간을 측정한 결과를 도표화 한 것이다. 실험 결과 본 논문에서 제안하는 기법을 이용하여 복호화기를 수행하고 이를 Intel VTune Performance Analyzer[7]를 이용해 측정하였을 때, PSNR(Peak Signal to Noise Ratio)의 손실은 전혀 없으면서 전체 실행 시간은 평균적으로 약 5% 개선시키고, 핵심 모듈인 움직임보상 과정에서는 약 20%의 성능 개선 효과를 볼 수 있었다. 또한 실행 클럭 수와 비교하기 위해 실제 복호화기의 실행 시간을 측정한 결과 약 20% 줄어든 것을 볼 수 있었다.

5. 결론

이 논문에서는 H.264/AVC 복호화기 복잡도의 약 40%를 차지하는 움직임 보상과정의 부하를 줄이기 위한 방법으로 반복적으로 수행되는 외부 메모리의 참조에 따른 오버헤드를 줄이기 위해 선인출 매커니즘을 적용한 방법을 제안하고 있다.

실험결과를 통해서 그 효과를 알 수 있는데, PSNR(Peak Signal to Noise Ratio)의 손실 없이 H.264/AVC 복호화기의 전체 실행시간을 약 5% 개선되었으며, 외부 메모리로부터 참조 데이터를 읽어 들이는데 많은 부하가 요구되었던 움직임 보상(Motion Compensation) 과정에서의 실행 시간(Clocks)도 약 20% 개선되는 등 높은 성능 향상을 보였다.

참고문헌

[1] Antonio Ortega and Kannan Ramchandran, "Rate-Distortion Methods for Image and Video Compression", IEEE SIGNAL PROCESSING MAGINE, pp23-50, November 1998

[2] Joint Video Team(JVT) of ISO/IEC MPEG & ITU-T VCEG, "Draft ITU-T recommendation and Final Draft International Standard of Joing Video Specification (ITU-T Rec. H.264-ISO/IEC 14496/10 AVC)", May, 2003.

[3] International Telecommunication Union(ITU-T) recommendation H.264, "Advanced Video Coding for Generic Audiovisual Services (ITU-T Rec. H.264(05/2003)", May, 2003.

[4] Lain E.G. Richardson, "H.264 and MPEG-4 Video Compression (Video Coding for Next-generatin Multimedia)", John Wiley & Sons Ltd, 2003.

[5] Joint Video Team (JVT), "Reference Software JM9.5," 2005.

[6] Joint Video Team (JVT) reference software home page- http://iphome.hhi.de/suehring/tml

[7] Intel Corp., Intel VTune Performance Analyzer, Version 7.1, 2004.