한국산학기술학회논문지
Vol. 11, No. 8 pp. 3076-3082, 2010

임베디드시스템 환경에서 하드웨어 기반 H.264 Encoder 최적화

조정현[1*], 이명수[1], 정한수[2], 김창석[2], 조대제[1]
[1]안동대학교 대학원 멀티미디어공학과, [2]공주대학교 대학원 컴퓨터과학과

Optimization of H.264 Encoder based on Hardware Implementation in Embedded System

Jung Hyun Cho[1*], Myung Soo Lee[1], Han Soo Jeong[2], Chang Suk Kim[2] and Dae Jea Cho[1]
[1]**Multimedia Engineering, Andong National University**
[2]**Computer Science, Kongju National University**

요 약 영상 압축 코덱(Codec)을 활용하여 군 혹은 민간 분야에서 다양한 기술과 제품들이 출시되고 있다. 기존 고성능 PC환경 하에서 영상 압축 코덱의 프로세스는 큰 문제가 되지 않았지만, 제한적인 시스템 자원을 가지는 임베디드 시스템 환경에서는 고해상도의 영상을 고밀도 압축하면서 발생하는 시스템 부하로 인하여 성능 및 활용도가 제한되는 문제가 부각되고 있는 상황이다. 본 논문에서는 임베디드 시스템 환경 상 기존 소프트웨어 알고리즘 형태의 영상 압축 방식에 대한 성능 및 주변 장치 연동 인터페이스 제약에 대한 해결책으로서 하드웨어 방식의 영상 압축 코덱성능 최적화, 외부 장치 연동의 편의성 및 확장성을 부각하기 위한 DirectShow 필터 인터페이스화를 제안하였고 검증을 위해 임베디드 시스템을 구현해서 시뮬레이션 하였다.

Abstract The techniques and the products which use various video compression codec are come out from army or civil field. In existing high-end PC environment, process of the video compression codec does not become a problem, but in embedded system environments which limited system resources, because the system load due to the high-resolution images compressed by high-density, issues of performance and utilization are highlighted. This paper proposes the DirectShow Filter interfaces which are a hardware method in order to solve the problem existing software algorithms for image compression performance and peripheral interfaces.

Key Words : Video Compress, Multimedia, Codec, H.264, DirectShow

1. 서론

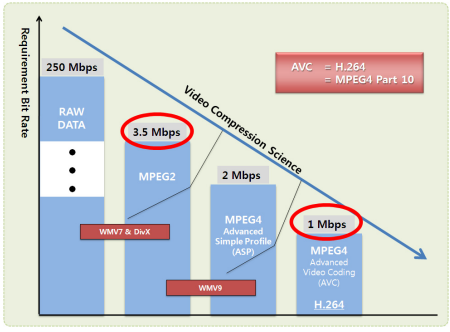
영상을 처리하는데 있어서 중앙처리장치, 메모리 등 시스템 자원(System Resource)이 어느 정도 갖추어진 PC 환경에서는 제약이 덜하지만 임베디드(Embedded) 시스템 환경에서는 자원의 제약사항이 많아서 단말기에서 영상을 처리 기술은 더 많은 기술을 요구하고 있다. 또한 영상처리 관련 기술이 빠르게 발전하면서 사용자들이 요구하는 영상에 대한 QoS 수준이 상당히 높아진 상태이며, 이에 따른 영상 데이터 크기 역시 비례하여 증가하였다[1,2]. 압축 이전의 원시(Raw) 영상 데이터의 경우 임베디드 환경에서는 처리가 어려울 정도로 데이터가 크지만, 그림 1과 같이 영상 압축 코덱(Codec)을 이용하면 1/250 이상 데이터의 크기를 줄일 수 있다[3-6].

[표 1] 영상 압축 방식 간 비교

항 목	Software 방식	Hardware 방식
압축코덱	MPEG2	MPEG4 AVC (H.264)
압축률	낮음 (1/70)	높음(1/250)
시스템 점유율	높음 (다중작업 어려움)	낮음 (다중작업 수월)

*교신저자 : 조정현(jhcho33@lycos.co.kr)
접수일 10년 07월 22일 수정일 10년 08월 09일 게재확정일 10년 08월 10일

표 1과 같이 영상 압축 방식도 순수 소프트웨어 압축 알고리즘 방식의 경우에는 임베디드 환경의 시스템 상에서 고화질 영상처리 시 성능 상 리소스의 한계점을 들어내기 때문에, 강제로 영상의 압축 품질을 크게 낮추는 방식으로 문제를 회피하였다[9,10]. 하지만 최근에는 소프트웨어기반의 압축 방식이 아닌 하드웨어 기반의 영상 압축 기술이 발전하면서 임베디드 환경의 시스템에서도 최신 사용자의 요구에 충분히 만족할만한 영상처리가 가능하게 되었다.



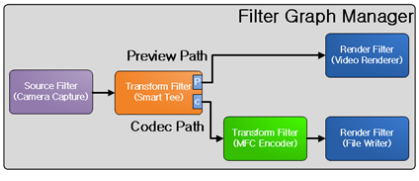
[그림 1] 코덱 별 압축 효율성 비교

또한 마이크로소프트사 DirectX의 하부에 존재하는 다이렉트쇼 기술을 사용한다면 멀티미디어 처리 과정에 대한 효율적인 제어가 가능하다. 본 논문은 하드웨어 방식의 영상 압축을 수행하는 마이크로프로세서를 기반으로 하는 임베디드시스템에서 입력 스트림 영상에 대한 압축 패스 내부 메모리 복사 과정을 최소화 하여 기존 소프트웨어 영상 압축 방식과 대비하여 처리 성능을 개선하였으며 이러한 전반적인 처리과정을 다이렉트쇼 기술로 제어함으로써, 외부 장비와의 연동 및 사용자의 코덱 제어 효율성을 극대화 측면에서 기술되었다.[7,8]

2. 본론

2.1 다이렉트쇼 필터를 이용한 코덱 연동 처리

카메라 모듈 영상 압축을 위하여 필터그래프의 기본적인 구성은 그림 2와 같은 필터그래프로 구성된다.

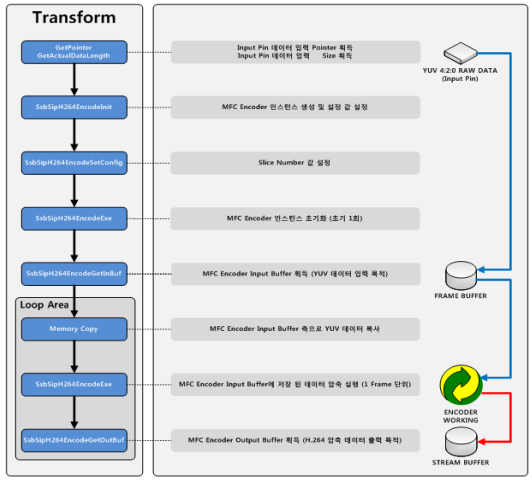


[그림 2] 영상 압축 Filter 구성

H.264 압축 코덱 Filter의 경우 YUV형태의 원시스트림(Raw Stream) 영상 데이터를 입력 받아 H.264 압축 후 Compress 데이터로 출력을 처리하는 변환 필터의 특징을 가지고 있다. 변환 필터의 핀은 입력과 출력 특성을 모두 가지게 되며 실질적인 데이터의 입출력 인터페이스로서 사용이 된다.

2.2 H.264 하드웨어 영상 압축 코덱 제어

영상 압축 필터 구성 상 실질적인 변환 구간으로서 그림 3과 같이 입력 핀 측으로 전달 된 카메라 모듈의 원시 스트림 데이터에 대한 압축 과정에 대한 전반적인 처리가 이루어지진다.



[그림 3] 영상 압축 코덱 제어 방식

영상 압축 파라미터 중 GOP(Group of Picture) 설정항목은 압축 시 레퍼런스 프레임인 IDR(Instantaneous Decoding Refresh) 구간과 다음 IDR 구간 사이에 발생되는 예측 구간 인 예측(Prediction) 프레임 의 개수를 설정하는 파라미터로서 영상처리 환경에 맞게 능동적인 수치 조절이 필요하다. 영상 통신 환경에서는 본 GOP 수치를 높게 설정하여 손실되는 영상 프레임 대한 에러 복구율을 높일 수 있다.

입력 버퍼 측으로 소스 영상 데이터 복사 및 압축 코덱 구동 후 압축 처리가 완료 된 데이터에 대한 메모리 측 저장을 위하여 출력 버퍼 주소를 획득하게 된다. 본 출력 버퍼는 스트림 버퍼로서 Line 형식으로 구성되어 있으며 한 개의 프레임 단위로 순차 저장된다.

2.3 영상 압축 코덱 최적 메모리 할당

카메라 모듈 측으로부터 입력되는 원시 스트림 데이터

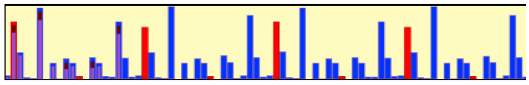
에 대한 압축 시 경유되는 메모리에 대한 최적 사이즈가 할당되어야만 효율적인 영상 압축이 이루어지게 된다. 카메라 모듈 측으로부터 데이터를 전달받게 되면 DMA 버퍼 상으로 저장 되며 총 4개의 구간으로 설정되어 있으며 압축 코덱의 출력 버퍼 측으로 메모리 복사가 이루어지며 이는 3~4 프레임의 원시 스트림 데이터 사이즈 이상을 만족해야만 한다.

[표 2] 코덱 버퍼별 크기 할당

	기준 용량		설정 용량
설정 항목	SD	CIF	VGA
STRM_BUF	307,200	307,200	409,600
FRAM_BUF	1,866,240	912,384	4,147,200

2.4 프레임 손실방지 및 압축 성능 개선

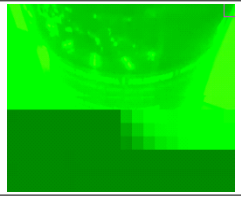
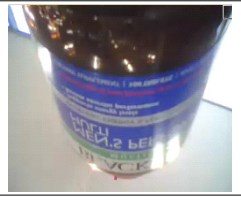
영상 압축을 위한 구성이 완료된 상태에서 카메라 모듈의 스트림 데이터에 대한 H.264 압축을 수행하기 전 카메라와 영상 압축 코덱에 대한 내부 버퍼 사이즈는 필수적으로 최적화가 이루어져야 한다. 또한 카메라 모듈 측 데이터 출력 메모리와 영상 압축 코덱의 입력 메모리 주소 매핑이 정상적으로 이루어져야지만 정확한 카메라 모듈 영상 데이터 전달이 가능하게 된다. 만일 이러한 메모리 주소 구간에 대한 매핑 상태가 상이하거나 미묘한 차이가 발생하게 되면 영상 압축 시 그림 4와 같이 프레임 손실 문제가 발생하기 때문에 확인 절차가 반드시 필요하다.



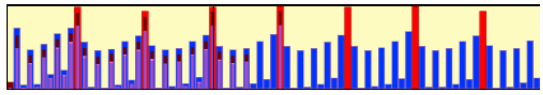
[그림 4] 비 정상 압축 스트림 그래프

메모리 주소 매핑 문제가 있는 상태로 영상 압축을 구동할 시 주기적으로 생성되어야 할 IDR 구간과 전체적인 예측 프레임 구간의 데이터가 불안정하여 영상 복원 시 표 3과 같은 화질 문제가 발생한다.

[표 3] 메모리 매핑에 따른 압축 영상 비교

메모리 매핑 문제 발생	정상적인 메모리 매핑
	

반면 정상적인 메모리 주소 매핑이 설정되었다면 그림 5와 같이 IDR 구간과 예측 프레임구간에 대한 주기와 용량이 균일하게 압축이 되는 스트림 결과를 확인할 수 있다.

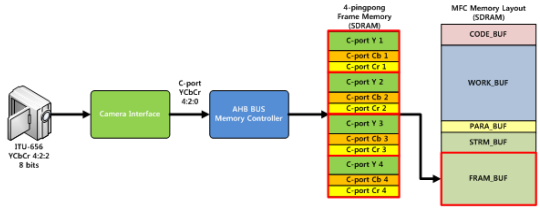


[그림 5] 정상 압축 스트림 그래프

카메라 모듈 측 데이터 출력 시점부터 최종 H.264 영상 압축 스트림 출력 시점까지 메모리 상태를 분석한 결과 불필요한 구간에 대한 제거가 효율적일 것으로 판단되었다.

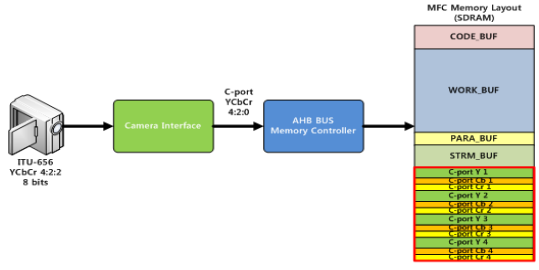
즉 카메라 모듈 측에서 출력되는 데이터가 DMA 버퍼를 경유하지 않고 직접적으로 코덱 버퍼 측에 출력되기 때문에 한번의 복사 과정을 줄일 수 있는 최적화 요소가 적용된다.

그림 6과 같이 최적화가 되지 않은 상태에서는 메모리 복사가 한번 이루어지게 된다.



[그림 6] 카메라 모듈 압축 데이터 이동 경로

하지만 그림 7과 같이 영상 압축 코덱 버퍼 상에 DMA 버퍼를 메모리 매핑을 하게 되면 설정된 구간은 카메라 측에서 출력되는 원시 스트림 데이터의 저장영역과 동시에 영상 압축 코덱이 직접적으로 관리하게 되는 메모리 영역으로도 공유하게 된다.



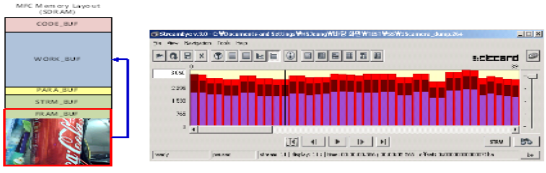
[그림 7] 메모리 매핑 데이터 최적 이동 경로

2.5 프레임 전송율 저하 및 Motion Block Error 방지 대책 적용

카메라 인터페이스와 영상 압축 코덱 간 타이밍 문제가 발생하게 될 시에는 다양한 문제점들에 노출이 된다. 실제로 카메라 원시 스트림 데이터 출력 시점과 영상 압축 동작 시점 간의 미묘한 타이밍 비동기 문제로 인하여 압축되는 영상 전체가 프레임 전송률 저하 및 모션 블럭 에러(Motion Block Error)와 같은 현상이 발생을 하였다.

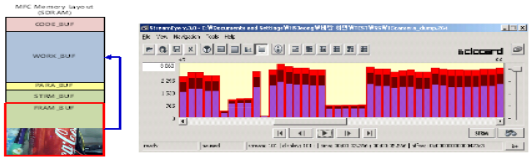
이와 같은 타이밍 문제는 현재 연구 과정에서 사용되는 카메라 모듈 과 같은 실시간 스트림 데이터에서만 발생되는 고유의 문제이며 파일과 같이 모든 데이터가 고정 할당되어있는 상황에서는 타이밍 문제를 크게 고려할 필요가 없다. 실제 테스트를 진행한 결과 메모리상에 저장 된 원시 스트림 데이터에 대한 영상 압축 시 정상적인 H.264 패턴으로 압축이 되는 것을 확인하였지만, 카메라 모듈 측으로부터 전달받은 실시간 데이터 압축 시 영상이 손실되는 현상이 발생하였다.

위와 같은 문제 사항에 대하여 분석 결과 카메라 모듈 데이터를 처리하는 환경에서는 메모리 대 메모리 복사 과정에서도 영상 압축 과정이 수행되면서 카메라 모듈 데이터 출력 시점과 영상 압축 시점간의 미묘한 타이밍 차이가 발생하게 된다. 이러한 원인으로 인하여 그림 9와 같이 힌 프레임에 대한 모든 데이터 처리가 이루어지지 않으며, 심하게 타이밍 오차가 생기는 상황에서는 2~3 프레임의 영상이 손실되는 심각한 문제가 발생하였다.



[그림 8] 정상 압축 시 영상 스트림 상태

영상 코덱 프레임 버퍼 측으로 카메라 데이터 복사 타이밍과 영상 압축 처리 타이밍에 대한 동기화가 부정확할 시, 정상적으로 한 프레임 데이터를 압축하지 못하기 때문에 처리되지 않은 빈 공간의 영상 까지도 압축이 수행된다. 이러한 비정상적인 압축 영상을 파일로 덤프하여 재생하게 되면 대형 모션 블럭 에러가 발생하게 된다. 또한 동기화 타이밍이 늦을 시에는 불완전한 프레임 데이터를 압축하게 되며, 반대로 동기화 타이밍이 지나치게 빠르게 되면 동일한 프레임의 영상을 2~3번 중복 압축하게 된다.

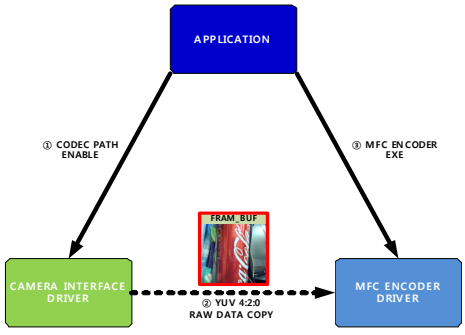


[그림 9] 비 정상 압축 시 영상 스트림 상태

카메라 원시 스트림 데이터 출력 속도가 영상 압축 코덱 처리 속도를 상회하여 동작하게 되면 압축 된 스트림 데이터 중간에 손실이 발생하게 된다. 설정한 비트 전송율 값으로 압축이 진행 되는 과정에서 용량이 감소 된 구간이 문제시 되는 타이밍 비동기 발생 구간이다.

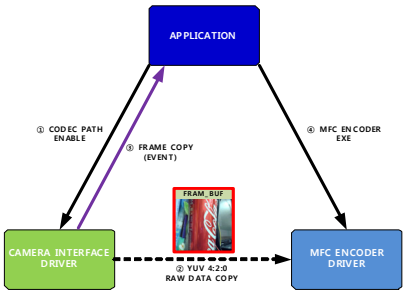
위와는 반대로 영상 압축 코덱 처리 속도가 카메라 원시 스트림 데이터 출력 속도를 상회하여 동작하게 되면 동일한 프레임을 중복하여 압축하게 된다. 스트림 그래프 결과와 같이 동일한 용량의 프레임이 중복되어 압축 된 구간들이 존재한다.

특히 이와 같은 중복 현상이 발생하게 되면 영상 복원 및 재생 시 프레임 저하를 유발하는 원인이 된다. 이러한 영상 압축 타이밍 문제를 해결하기 위해서는 카메라 인터페이스와 영상 압축 코덱 간의 타이밍을 동기화 할 수 있는 이벤트 처리가 반드시 필요하다.



[그림 10] 기존 영상 처리 이벤트 구조

그림 10과 같이 기존에는 카메라 코덱 패스 활성화와 동시에 영상 압축 프레임 버퍼 측으로 카메라 원시 스트림 영상 복사를 수행하며, 어플리케이션 측에서 최종적인 영상 압축 실행 명령이 전달되면 프레임 버퍼 측에 저장 된 데이터를 압축하는 구조로 이루어져 있다. 본 방식에서 가지는 문제점으로는 카메라 원시 스트림 영상 복사 시점과 영상 압축 수행 시점 간 타이밍 제어가 존재하지 않는다. 즉, 이러한 구조는 카메라 영상 출력과 영상 압축 구동 간 타이밍 문제 발생 원인이 된다.



[그림 11] 신규 영상 처리 Event 구조

타이밍 비 동기화 문제를 제거하기 위해서는 그림 11과 같이 카메라 모듈 측에서 출력되는 데이터에 대한 이벤트 처리가 필수적이다. 출력의 기준은 한 프레임 단위이며 카메라 모듈 측에서 한 프레임의 데이터가 출력되면 어플리케이션 측으로 이벤트를 전달하는 구조가 필요하다. 어플리케이션 측에서 이러한 이벤트를 전달받는 시점을 기준으로 카메라 영상 데이터가 정상적인 출력 및 메모리 상 저장이 완료 되었다는 것을 판단한 후 영상 압축 코덱을 동작시키는 구조이다. 이러한 구조는 카메라 모듈 측에서 한 프레임을 출력할 때마다 반복되며 디버깅 메시지로 확인하게 되면 카메라 데이터 출력과 영상 압축이 1:1로 수행되는 것을 확인할 수 있다.

3. 시뮬레이션 및 결과

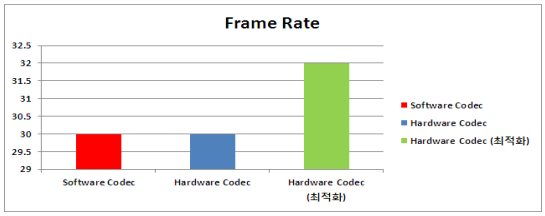
3.1 시뮬레이션 환경

영상 압축 시 카메라 모듈 측 메모리와 영상 압축 코덱 메모리 간 최적화를 통하여 복사구간을 최소화 하였으며, 압축 프레임 손실 및 모션 블럭 에러 등을 유발하는 동기화 문제에 대한 분석 및 개선 방안 적용하였으며 이에 대한 성능 분석 시 프레임 전송율와 시스템 점유율에 대한 수치를 기준으로 설정하였다. 영상 처리의 경우 640 x 480 해상도를 기준으로 30 프레임 주기로 1,000 kbps의 비트 전송율로 모든 테스트 환경에 적용하였다.

측정 데이터에 대한 신뢰성을 높이기 위하여 1회 측정 시 3,000 프레임 (약 100초) 가량 압축을 진행하였으며 동일한 환경에서 항목 당 총 6번 씩 반복 테스트를 진행하여 평균값을 산출하였다.

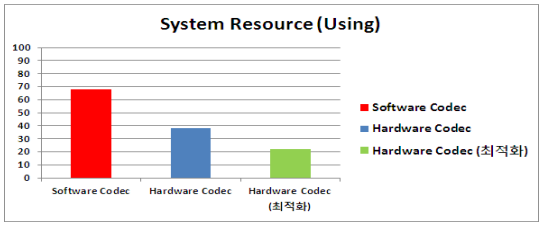
3.2 시뮬레이션 결과

카메라 모듈 측 영상을 압축하였을 시에는 그림 12와 같이 연구에서 최적화 한 방식이 약 2 프레임을 추가적으로 압축 할 수 있는 성능 개선결과를 보이고 있다.



[그림 12] 압축에 따른 프레임 전송율

반면 시스템 점유율 기준을 통한 비교 테스트 시 확연한 차이를 보이고 있음을 알 수 있다.



[그림 13] 압축에 따른 시스템 점유율

기존 소프트웨어 코덱 방식에서는 70%에 육박하는 시스템 자원을 점유함으로서 기타 어플리케이션 및 처리 환경 상 잔여 자원이 거의 없음을 알 수 있다. 그에 반면 하드웨어 코덱 방식을 이용하여 되면 그림 13과 같이 39%까지 낮아지게 되며, 본 연구에서 수행한 최적화 사항들을 적용할 시 22%까지 시스템 점유율을 최적화 할 수 있으며, 이로써 영상 압축 처리 이외 작업에도 충분한 시스템 자원을 보유할 수 있음을 알 수 있다.

기존 드라이버 형태로 카메라 영상 압축을 구성할 시 초기 카메라 모듈 측 원시 스트림 영상 입력 시부터 최종 영상 압축 후 출력 시 까지 하위 계층 소스에 대한 직접적 수정이 필요하다. 만약 카메라 모듈 영상 압축이 아닌 네트워크 스트림 데이터 입력에 따른 압축 혹은 압축 후 후처리에 따른 작업사항이 발생하게 될 시 모든 사항들은 별도의 인터페이스 구성 작업을 수행해야만 사용이 가능하였다. 함축적으로는 영상 압축 코덱에 대한 확장성이 크게 떨어지는 단점으로 풀이할 수 있다. 이는 실질적인 개발 시 시간적 낭비가 큰 부분이기 때문에 이에 대한 개선이 필요함에 따라 영상 압축 코덱에 대하여 다이렉트쇼 필터화를 진행하였다. 영상 압축 코덱이 COM(Component Object Model) 기반의 다이렉트쇼 필터로 동작하게 되면 다양한 COM 라이브러리 및 인터페이스를 사용할 수 있으며, 필터에 대한 입력 및 출력 단에

대한 자유로운 개발이 가능하다. 또한 필터 기반에서는 다이렉트쇼 상에서 제공하는 다양한 WDM 연동장치를 필터 형식으로 연결할 수 있기 때문에 단순히 어플리케이션 구성을 하는 작업만으로도 개발 기간을 상당 수 줄일 수 있다.

4. 결론

본 논문에서는 임베디드 시스템 환경에서 기존 소프트웨어 알고리즘 형태의 영상 압축 방식에 대한 성능 및 주변 장치 연동 인터페이스 제약에 대한 해결책으로서 하드웨어 알고리즘 영상 압축 코덱에 대한 성능을 최적화함과 동시에 외부 장치 연동의 편의성 및 확장성을 부각하기 위한 다이렉트쇼 필터 처리 구조를 제안하였다. 영상 압축 처리를 수행하면서 발생하는 타이밍 비동기화 문제 및 고용량 데이터의 잦은 메모리 이동 구조에 대한 최적화를 진행하여 기존 환경에서 가지는 문제점들을 제거하고, 제한 된 성능의 임베디드 시스템 자원을 최대한 활용할 수 있도록 진행하였다. 연구 결과와 같이 기존 환경 대비 압축 프레임 전송율의 경우 2 프레임 이상의 성능 차이를 보이고 있다. 또한 시스템 리소스 측면에서는 모든 환경 상 40% 정도의 최적화가 된 결과를 확인하였다. 또한 다이렉트쇼 필터 구성을 통하여 각종 WDM 장치 연동과 압축 데이터에 대한 후처리에 개발을 드라이버 형식이 아닌 필터 연결만으로 구성이 가능하도록 하여 개발 측면에서의 편의성 및 확장성을 개선하였다.

최근 출시되는 마이크로프로세서 칩들은 기본적으로 영상 코덱을 자체적으로 내장하고 있으며, 나노 공정 기술이 발달함에 따라 더 복잡한 연산일 수행이 가능한 고성능 칩들이 개발되고 있다. 임베디드 시스템의 코어 클럭 속도 역시 1GHz를 상회한 상태이기 때문에 본 연구에서 비교 대상으로 사용 된 소프트웨어 알고리즘 방식의 영상 압축 방식도 굳이 하드웨어 영상 압축 방식에 비하여 단점만 가지지는 않을 것으로 예상된다. 하지만 영상 처리만을 단독으로 수행하는 제품들은 현 시장에서 경쟁력이 없으며, 화려한 UI 환경과 다양한 멀티태스킹 환경을 부가적으로 요구 하고 있는 실정이다. 이러한 처리들은 시스템 리소스 사용이 심하며 영상 압축을 소프트웨어 방식으로 처리 할 여력이 아직은 부족하다는 판단이다. 이러한 이유 때문에 하드웨어 구동 방식의 영상 코덱 사용을 통하여 최대한의 시스템 리소스를 절약하는 것이 부가적인 서비스 개발에 있어 효율적인 환경으로 인식되고 있다.

이처럼 앞으로도 하드웨어 방식의 영상 코덱은 널리 사용될 것이며 본 연구에서 진행된 사항처럼 구동 환경에 대한 최적화 및 편의성을 고려한 인터페이스 연동성은 앞으로도 큰 비중을 차지할 것으로 판단된다. 현재는 윈도우즈 계열의 임베디드 시스템 상에 이러한 최적화 사항들을 적용하였지만, 추후에는 안드로이드와 같은 신규 운영체제 환경에 대한 다양한 최적화 연구가 필요하다.

참고문헌

[1] 봉정식, "H.264 코덱에서 동영상 성능개선 연구", 대한전기학회집, 2005.

[2] 신승호, 김경남, 김태용, "멀티미디어 방송(DMB)에서의 H.264/AVC압축 파라미터 성능연구", 방송공학회지 제12권 제4호 pp.28-39, 2007.

[3] 김대연, 임성창, 이영렬, “.264/AVC의 화면 내 예측을 위한 새로운 고속 모드 결정 방법” 한국방송공학회학술대회, pp. 117-120, 2006년 11월.

[4] Akiyuki Tanizawa, Shinichiro Koto, Takeshi Chujoh, "Fast rate-distortion optimized coding mode decision for H.264", Fundamental Electronic Science pp.41-55, 2007.

[5] Loren Merritt and Rahul Vanam," Improved rate control and Motion Estimation for H.264 Encoder", ICIP 2007, pp V-309~312, 2007

[6] H.264/AVC encoder reference software 13.2 (http://iphome.hhi.de/suehring/tml/)

[7] 이일주,임성준, 채현석,“.264 코덱을 사용한 고성능 DVR 시스템 개발에 관한 연구” 한국산학기술학회 논문지, Vol 10. No.1, pp 110~116, 2009

[8] 서기범, “H.264 율제어 알고리듬의 하드웨어 설계” 한국 산학기술학회 논문지, Vol.11, No.1, pp175~181, 2010

[9] 신화선, "DirectShow 멀티미디어 프로그래밍", 한빛미디어, 2002.

[10] 호요성, "H.264 / AVC 알고리즘 이해와 프로그램 분석", 두양사, 2009.

조 정 현(Jung Hyun Cho) **[준회원]**

- 2009년 3월 ~ 현재 : 안동대학교 대학원 멀티미디어공학과 석사과정

<관심분야>
정보통신, 멀티미디어

이 명 수(Myung Soo Lee) **[준회원]**

- 2009년 3월 ~ 현재 : 안동대학교 대학원 멀티미디어공학과 석사과정

<관심분야>
정보통신, 멀티미디어

정 한 수(Han Soo Jeong) **[준회원]**

- 2008년 2월 : 백석대학교 정보통신학부 졸업
- 2008년 3월 ~ 현재 : 공주대학교 대학원 컴퓨터과학과 석사과정

<관심분야>
멀티미디어, 영상처리, 3D

김 창 석(Chang Suk Kim) **[정회원]**

- 1983년 2월 : 경북대학교 전자공학과 졸업
- 1990년 2월 : 경북대학교 전자공학과 석사
- 1994년 8월 : 경북대학교 컴퓨터공학과 박사
- 1983년 1월 ~ 1994년 12월 : ETRI 선임연구원
- 1998년 3월 ~ 현재 : 공주대학교 컴퓨터교육과 교수

<관심분야>
지능정보시스템, 데이터베이스, XML

조 대 제(Dae Jea Cho) **[정회원]**

- 1986년 2월 : 경북대학교 전자공학과 석사
- 2001년 8월 : 경북대학교 컴퓨터공학과 박사
- 2002년 3월 ~ 현재 : 안동대학교 교수

<관심분야>
멀티미디어, 멀티미디어콘텐츠, 보안