효율적인 메모리 관리를 이용한 ARM9 프로세서에서의 JPEG2000 코덱 구현

論 文
55D-10-3

Implementation of JPEG 2000 Codec on ARM9 Processor Using Effective Memory Management

曺 時 元* · 李 東 旭†
(Shiwon Cho · Dong-wook Lee)

Abstract - In this paper, we propose an implementation of JPEG2000 codec on the ARM9 Processor which includes independent memory management facility. The codec and memory management facility together can control the encoding and the decoding process effectively within available memory area. Embedded appliances like cellular phones have very limited internal memory which can't be expanded easily. However, they should provide various applications and services using restricted memory resources. The proposed codec with memory management can provide image quality that is identical to the original image on embedded platform. The implemented codec has no memory conflict with other applications. It shows that the proposed codec can manage memory resources efficiently.

Key words : JPEG2000, Embedded System, ARM9, Memory Management

1. 서 론

높은 해상도를 지원하는 디지털 카메라가 장착된 휴대폰의 사용이 증가되면서, 대용량의 이미지 정보를 높은 압축으로 빠르게 처리해야 할 필요가 있다. JPEG는 Joint Photographic Experts Group(정지영상전문가그룹)의 약자로 현재 가장 많이 쓰이는 정지영상압축 규격 중 하나이다. 1992년 JPEG이 국제 표준으로 채택된 이후, 다양한 멀티미디어 기기에 이용되고 있다. JPEG는 호프만 코딩을 이용한 블럭 DCT 변환을 이용하기 때문에, 높은 압축율을 적용할 경우, 블록화 현상이 심하게 발생하여, 이미지의 품질이 떨어지는 단점이 있다.

JPEG 2000은 압축율을 더욱 높이면서도 이미지 품질을 보존할 수 있는 방법으로 웨이브렛과 EBCOT를 이용하여 DCT변환으로 생기는 블록화 현상을 방지한다. 기존의 JPEG은 RGB 데이터만 저장할 수 있지만, JPEG 2000은 CMYK와 같은 다양한 컬러 스페이스를 지원하고, 256채널의 생생한 컬러를 표시할 수 있을 뿐만 아니라, 다양한 정보를 파일에 삽입하는 것이 가능하다.

휴대폰이나 PMP같은 임베디드 기기에서는 일종의 멀티태스킹 환경이 만들어지면서 여러 애플리케이션의 동작을 원활하게 처리하기 위해 기본으로 탑재되는 메모리가 점점 증가하고 있다. 또, 이동 통신사에서 제공하는 서비스를 위한 메모리 상주형 프로그램들은 다른 프로그램이 실행될 때에도 백그라운드에서 메모리 자원을 할당받아 실행되고 있다. 게임, 영화등과 같은 다른 컨텐츠와 함께 실행이 되면 메모리 사용량은 더욱 늘어나게 된다. 게다가 이동통신사별로 여러 서비스를 제공하기 위해, 멀티태스킹 환경을 제공하는 추세라 메모리의 수요는 계속 늘어날 전망이다.

이동 통신사들은 컨텐츠나 서비스 별로 사용 가능한 힙메모리 용량을 제공하는 방식으로 메모리를 관리하고 있다. 휴대폰 단말기에 탑재된 메모리 용량과 애플리케이션에 따라 세부적으로 사용할 수 있는 메모리를 제한하는 방식이다. 저장용으로 사용되는 외장형 플래시 메모리는 사용자가 필요에 따라 추가할 수 있지만, 내장 메모리는 확장이 불가능하기 때문에, 향후 다양한 애플리케이션과 서비스가 제공되고, 휴대폰 단말기의 멀티태스킹 기능이 확장될 것을 고려하여 제조사 측에 메모리 탑재량을 늘릴 것을 요구하는 추세이다.

구현된 JPEG2000 코덱은 독립적인 메모리 관리 기능이 같이 포함되어, 다른 애플리케이션과 메모리 영역에서의 충돌을 제거할 수 있으며, 적용되는 플랫폼에서 메모리 자원을 효율적으로 관리할 수 있다. 또한 휴대폰 단말기와 같은 임베디드 환경에 적용할 경우, 소프트웨어 호환성과 플랫폼의 안정성을 높일 수 있다. 본 논문은 블록 인코딩/디코딩 방식을 이용하여 메모리 자원이 제한적인 ARM9과 같은 임베디드 프로세스에서 메모리 사용 효율을 더 높일 수 있도록 JPEG 2000 코덱을 독자적으로 구현하였다.

2. JPEG 2000 코덱

JPEG 2000은 JPEG 표준을 만든 ISO 산하의 JPEG 그룹이 JPEG 의 뒤를 이을 새로운 정지영상압축표준으로 만들

† 교신저자, 正會員 : 東國大 工大 電氣工學科 敎授 · 工博
E-mail : dlee@dongguk.edu
* 正 會 員 : 東國大 工大 電氣工學科 博士課程
接受日字 : 2006年 6月 15日
最終完了 : 2006年 9月 5日

었다. JPEG 기술의 기본은 DCT(District Cosine Transform)라는 일종의 푸리에 변환을 이용하여 이미지를 인코딩하는 것이다. 푸리에 변환은 임의의 신호를 여러 주파수의 삼각함수의 합으로 표현할 수 있다. 이를 이용하여 이미지를 8x8의 블록으로 쪼갠 후 DCT를 통해 64개의 기본적인 패턴으로 표현하고, 이 기본적인 패턴에 가해주는 계수 값을 양자화하고, 양자화된 값을 호프만 코딩을 이용하여 인코딩을 하게 된다. JPEG의 기본 기술은 사람의 눈으로 구분하기 어려운 주파수 성분을 제거하여 정보량을 낮추어 압축하는 방식이다. 압축을 푸는 디코딩 방법은 인코딩과 반대로 IDCT(Inverse Discrete Cosine Transform)를 거쳐 원하는 영상을 얻을 수 있다.

JPEG 2000은 DCT대신 DWT(Discrete Wavelet Transform)를 사용한다. JPEG에서 이용하는 DCT의 단점은 삼각 함수가 주기를 가지고 무한히 반복하는 함수이기 때문에 주파수 영역에서 시간 정보를 표현하지 못한다는 점이다. 시간에 따라 주파수가 변동하는 신호를 푸리에 변환으로 나타내면 무한개에 가까운 삼각 함수를 더해야만 제대로 된 신호를 재생할 수가 있다. 실제로는 무한개의 삼각 함수 즉, 무한개의 주파수로 원래 신호를 표현할 수는 없기 때문에 주로 사용되는 범위의 주파수만을 이용하여 표현을 하게 된다. 이 경우 시간에 따라 변화가 심한 신호는 원래 모양에 비해서 왜곡이 많이 발생하게 된다. JPEG에서는 이미지를 블록 단위로 압축을 하였기 때문에 사진을 확대하면 작은 블록 단위로 영상이 손상되는 것을 볼 수 있다. 그러나, JPEG 2000에서는 이미지를 웨이브렛 변환을 이용하여 압축하였기 때문에, 확대를 하여도 자연스럽게 연속적으로 매끄러운 선으로 표현될 수 있다. 웨이브렛 변환을 이용하여 이미지를 압축하면서 전체 이미지를 1/4, 1/16, 1/64 등의 낮은 해상도의 이미지로 줄일 수가 있다. 일반적인 비트맵 이미지는 축소할 때 원래의 정보가 제거되었기 때문에 원래 크기로 돌리면 단순히 블록이 확대되는 블록화 현상이 생기지만, 웨이브렛 변환의 경우는 축소할 때 고주파 성분을 따로 저장하여, 확대할 때 다시 그 정보를 이용해서 원래 이미지를 복원한다. 원본 이미지를 작은 이미지로 축소하면서, 다시 확대할 때 사용할 고주파 성분을 ROI(Region of Interest), EBCOT 등의 압축 기술로 압축한 것이 JPEG 2000이다. 고주파 성분은 같은 값들이 반복해서 나타나는 경우가 많기 때문에, 기존 JPEG보다 더 효율적으로 압축을 할 수 있다.

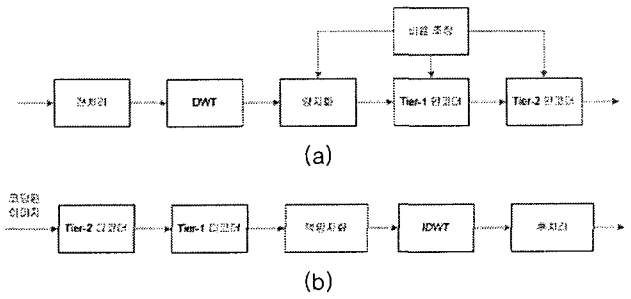


그림 1 JPEG 2000 코덱의 구조 (a) 인코더 (b) 디코더
Fig. 1 Structure of JPEG 2000 codec (a) Encoder (b) Decode

JPEG 2000의 적용 분야로 관심을 받고 있는 분야는 디지털 카메라와 디지털 카메라 폰 등의 임베디드 기기 응용 분야이다. 현재 디지털 카메라와 카메라 폰의 경우는 새로운 문화 조류를 형성할 정도로 보급이 확대되고 있으며, 작은 저장 메모리를 가지고 있는 여러 종류의 임베디드 기기들이 JPEG 2000 코덱을 많이 요구할 것으로 예상된다.

JPEG 2000 코덱[그림 1]의 상세한 사양과 프로세스는 [1] 또는 국제 규격안 [2]를 참조할 수 있다.

3. 소프트웨어 구성

구현된 JPEG 2000 코덱 라이브러리는 이미지 데이터를 처리하는 소프트웨어 툴킷(Software Toolkit) 이다. JPEG 2000 코덱 엔진(Codec Engine), 코덱 드라이버(Codec Driver), 그리고, 메모리 관리 엔진(Memory Management Engine), 세 부분으로 구성되어 있으며, 상용 제품에 적용이 가능하도록 모두 자체적으로 구현하였다[그림 2].

하드웨어로 구현된 전용 코덱을 사용할 경우, 빠른 처리 속도를 보장받을 수 있지만, 설계 단계에서 기구 설계, 동작 클럭, 소비 전력, 제조 단가 등을 고려해야 하고, 많은 개발 비용과 시간의 투자가 필요하다. 소프트웨어로 구현된 코덱의 처리 속도는 운영체제와 프로세스의 성능에 크게 의존하며, 하드웨어 코덱보다 처리 속도는 느리지만, 개발에 필요한 개발 비용과 기간을 크게 줄일 수 있는 장점이 있다. 또, 사용하는 프로세스가 빠른 성능을 가지고 있다면, 별도의 하드웨어 코덱이 필요없이 소프트웨어 코덱만으로 필요한 기능을 제공할 수 있다.

JPEG 2000 코덱 엔진은 입력 이미지를 JPEG 2000 형식으로 변환하는 기능을 수행하며, 코덱 드라이버를 통해 JPEG[3], PNM[4], BMP[5], NV21[6] 형식의 이미지를 JPEG 2000 형식으로 변환할 수 있도록 구현되어 있다. NV21형식은 YUV 형식[6]의 하나로 퀄컴의 MSN6550 계열에서 지원하는 컬러 스페이스 모델 중 하나이다. QCM411 방식이라고 부르기도 하며, MSN6550 계열의 프로세서를 사용하는 카메라 폰에서 디지털 카메라를 통해 입력되는 이미지의 기본 포맷으로 많이 사용한다. JPEG 2000 코덱 엔진에서 입력되는 데이터는 NV21[6]와 같은 YUV 형식으로 표현된 데이터 포맷을 사용한다. 인코딩을 할 경우, YUV형식의 데이터를 입력하면, JPEG 2000 형식으로 압축된 데이터를 얻을 수 있다. 반대로 디코딩을 할 경우, JPEG 2000 형식의 데이터를 입력하면, YUV 형식으로 표현된 데이터를 얻는다.

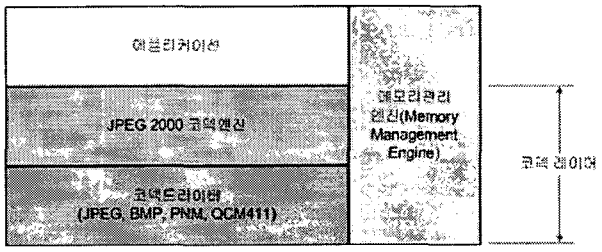


그림 2 소프트웨어 구성
Fig. 2 Software organization

YUV 형식을 사용하면 디지털 카메라 센서 장치간은 하드웨어와 직접 연결하여 데이터를 빠르게 처리할 수 있으며, RGB형식에 비해 이미지 데이터 크기가 작고, 다른 형식의 이미지로 변환하기 편리한 장점이 있기 때문에 YUV형식을 JPEG 2000 코덱 엔진의 기본 포맷으로 적용하였다.

코덱 드라이버는 JPEG 2000 코덱 엔진과 입출력 데이터 변환을 위한 인터페이스를 담당한다. JPEG, BMP, PNM, RGB565 와 같은 형식의 이미지를 YUV형식으로 변환하거나, 반대로, YUV형식으로 표현된 이미지를 JPEG, BMP, PNM형식으로 변환하기 위한 함수(API)들을 제공한다.

메모리 관리 엔진은 운영체제에서 할당한 힙 메모리를 관리하고, JPEG 2000 코덱 엔진과 코덱 드라이버에서 사용하는 메모리 영역을 관리한다. JPEG 2000 코덱 엔진은 메모리 관리 엔진을 통해 미리 할당한 힙메모리 영역 범위 내에서 이미지의 인코딩/디코딩을 처리한다. EBCOT알고리즘과 같이 메모리를 많이 사용하는 부분에서는 추가 메모리를 요구하지 않도록 메모리를 관리한다. 힙메모리는 임베디드 기기의 운영체제에서 동적으로 관리될 수 있으며, 이미지 인코딩/디코딩 과정이 끝나면 반환하여, 운영체제에서 메모리를 효율적으로 관리할 수 있도록 하였다.

미리 확보된 일정량의 힙메모리를 가지고 있으면, 데이터 저장에 관련된 처리를 좀 더 쉽게 할 수 있으며, 운영체제에 매번 메모리를 요청하는 것 보다 빠르게 데이터를 처리할 수 있다. 힙메모리가 필요한 메모리 요구량 보다 적게 할당되면, 중간에 메모리가 부족하여, 작업이 중단되거나, 부족한 힙메모리를 다시 할당하게 되면, 전체적으로 메모리 효율이 떨어지게 된다.

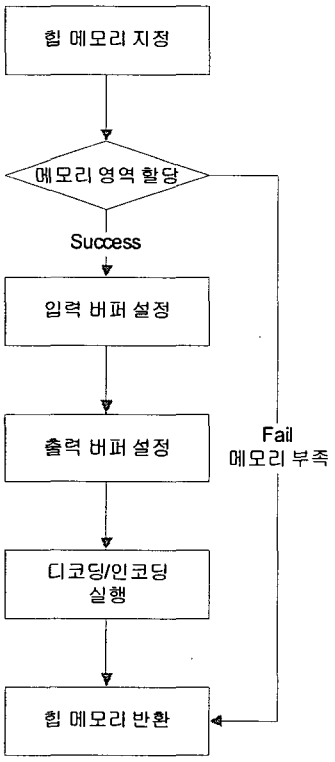


그림 3 JPEG 2000 코덱 라이브러리 알고리즘
Fig. 3 Algorithm of JPEG 2000 codec library

그림 3은 JPEG2000 코덱 라이브러리의 기본 알고리즘을 나타낸다. 먼저 사용할 수 있는 힙메모리 크기를 설정하면, 구현된 소프트웨어는 힙메모리 범위 내에서 이미지를 인코딩/디코딩을 수행한다. 이미지 변환 작업이 종료되면, 사용하지 않는 힙메모리는 반환한다. 초기에 힙메모리를 크기를 정할 때, 처리하려는 이미지의 해상도를 고려하여 힙메모리를 할당하여야 한다.

JPEG 2000 코덱 엔진의 인코더는 입력 영상을 스캔라인 버퍼(Scanline Buffer)를 이용해 일정한 블록 단위로 이미지를 처리한다. 스캔라인 버퍼에 저장된 소스 이미지는 32x32 byte의 타일(Tile)이라는 서로 겹치지 않는 직사각형의 조각으로 분리된다. 분리된 타일을 타일 컴포넌트(Tile-Components)라고 부르며, 인코더는 타일 컴포넌트 단위로 DWT를 수행한다. DWT에서 처리한 웨이브렛 계수(Wavelet Coefficients)들은 타일 버퍼(Tile Buffer) 에 저장된다[7]. 타일 버퍼에 저장된 데이터는 타일 스필리터(Tile Splitter)를 통해 최소 32x32의 코드 블록으로 분할된다. 분할된 코드 블록은 비트 평면 단위로 EBCOT(Embedded Block Coding with Optimized Truncation)[7][8][9]알고리즘에 의해 각 비트와 각 비트에 대한 정보를 산술 부호화기로 보낸다[그림 4]. EBCOT는 JPEG200코덱에서 가장 많은 계산량과 메모리를 요구하기 때문에, 메모리 관리 엔진에서 메모리를 동적으로 관리하면서, 이미지를 처리한다.

JPEG 2000 인코더는 압축율보다는 비손실 압축으로 이미지의 품질을 유지하면서, 메모리 효율을 최대한 높이는 방식으로 구현하였다.

그림 4. JPEG2000 인코더
Fig 4. JPEG 2000 encoder

JPEG 2000 코덱 엔진의 디코더는 인코더와 반대의 순서로 진행된다. 타일 스필리터 대신 타일 콤포저(Tile Composer)에서 서로 분리된 코드 블록을 합쳐서, IDWT를 위한 타일 버퍼에 저장하고, 타일 단위로 스캔라인 버퍼를 구성한다. 완성된 스캔라인 버퍼를 차례대로 디코딩한다[그림5]. 복구된 이미지의 형식은 YUV 형식이다[10].

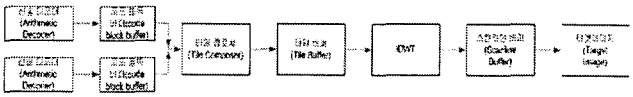


그림 5 JPEG 2000 디코더
Fig. 5 JPEG 2000 decoder

JPEG 2000 코덱 엔진은 메모리 사용량을 줄이기 위해, 스캔 라인 버퍼를 이용하여 입력 영상을 조금씩 나누어서 인코딩과 디코딩을 수행한다. 그림 6에서와 같이 이미지를 몇 개의 블록으로 나누어서 처리하기 때문에, 최소한의 기본 메모리 버퍼 용량만큼 메모리를 사용한다.

이와 같이 블록 단위로 이미지를 처리하는 방법은 적은 메모리를 가지고 큰 이미지를 처리할 수 있다는 장점이 있다. 하지만, 적당한 기본 메모리 버퍼 용량이 확보되지 못하면, 이미지를 처리하는 시간이 오래 걸리거나, 처리하는 도중에 메모리 부족으로 종료될 수 있다. 또, 블록 단위로 이미지를 처리하기 때문에, 메모리를 필요한 만큼 이용할 수 있는 환경과 비교하여 상대적으로 압축율과 품질에 한계가 있다.

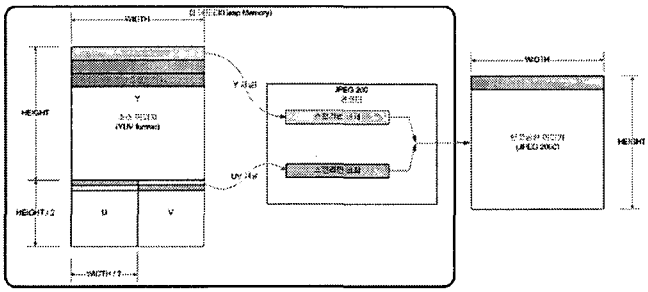


그림 6 힙메모리 블록 다이어그램
Fig. 6 Block diagram of Heap Memory

메모리 요구량은 입력 이미지의 비트율과 해상도에 의존적이기 때문에, 해상도가 높고, 비트율이 높으면 메모리 요구량도 증가한다. 이미지의 해상도에 따라 힙메모리 사이즈를 조정할 수 있으며, 힙메모리 범위 내에서 이미지를 처리할 수 없는 경우, 사용할 수 있는 메모리 영역에서 이미지를 일정한 크기의 블록 단위로 분할하여 처리를 한다. 힙메모리가 너무 작으면, 분할 인코딩/디코딩 과정의 소요 시간이 너무 오래 걸리거나, 처리 과정에서 메모리가 부족할 수 있다.

스캔라인 버퍼의 기본 사이즈는 WIDTH x 32 byte이며, 32 x 32 byte 단위로 조절할 수 있다. 스캔라인 버퍼의 크기가 너무 작으면, 반복 처리 횟수가 증가하기 때문에, 이미지를 처리하는데 처리 시간이 더 필요하게 된다.

힙메모리 영역은 YUV 데이터 보관 영역, 버퍼 블록으로 구분된다[그림 7]. 버퍼 블록을 제외한 영역은 처리하는 해상도에 따라 결정된다. 버퍼 블록은 스캔 라인 버퍼, 코드 블록 등에 필요한 메모리를 동적으로 할당한다. 본 논문에서 권장하는 최소 힙메모리 크기는 512 KB이다.

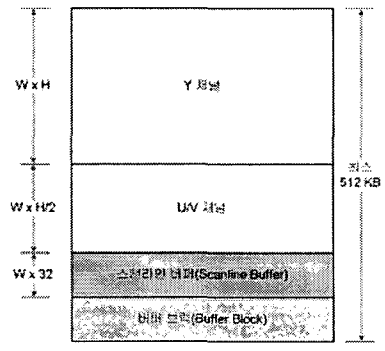


그림 7 힙메모리 맵
Fig. 7 Heap Memory Map

실제 휴대폰과 같은 상용 임베디드 제품에서는 전체 메모리를 각 프로그램 별로 사용할 수 있는 메모리를 할당하기 때문에, 할당된 메모리 내에서 이미지의 최대 해상도를 처리하는데 메모리가 부족한 경우가 자주 발생한다. 추가 메모리를 요구하지 않고, 이미지 인코딩/디코딩 처리뿐만 아니라 이미지 변환, 이미지 합성 등 다양한 특수 효과를 같이 적용하기 위해 힙메모리 내에서 처리할 수 있어야 하기 때문에, 자체적으로 할당된 메모리를 관리하면서, 이미지를 힙메모리에서 수용할 수 있는 범위 내에서 몇 개의 블록으로 나누어서 처리한다.

4. 테스트

구현된 JPEG 2000 코덱을 평가하기 위하여 이미지 해상도와 메모리 사용량에 대한 실험을 하였다. 실험에 사용된 장비는 ARM9(ARM926EJ-S) 프로세스와 64MB메모리를 장착한 개발 보드와 ADS(ARM Developer Suite), 그리고 PC로 구성되어 있다. 실험에 사용된 이미지는 320x240, 640x480, 1600x1200, 2048x1036 해상도의 24bit BMP 이미지 파일을 이용하였으며, 이미지 파일은 외부 플래시 메모리에서 읽고 저장한다. 사용할 수 있는 메모리는 최대 2MB로 제한하였다. BMP 파일을 JPEG 2000 형식으로 인코딩하는 실험과 인코딩된 파일을 다시 BMP형식으로 변환하는 디코딩 실험을 하였으며, 각 해상도별로 메모리 사용량과 압축률을 측정하였다[그림 8].

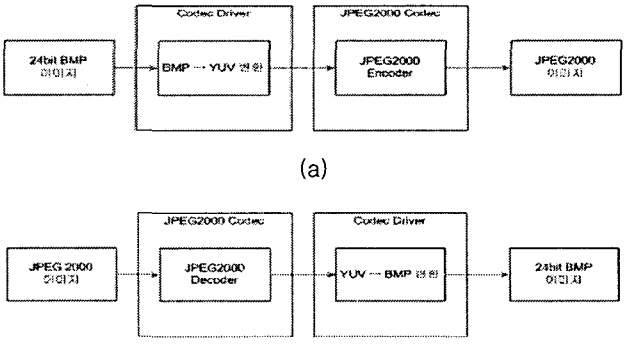


그림 8 JPEG 2000 코덱의 인코딩/디코딩 테스트 (a)인코딩 테스트 (b)디코딩 테스트
Fig. 8 Encoding/Decoding test of JPEG 2000 codec (a)Encoding test (b) Decoding test

표 1은 각 해상도별로 인코딩/디코딩할 수 있는 최소 힙메모리 요구량과 메모리 사용량을 비교하였다. 최대 메모리 사용량은 인코딩/디코딩에 사용된 스캔라인 버퍼와 버퍼 블록의 크기를 나타내며, 나머지 부분은 YUV 데이터 영역에 할당하여 사용한다. 1,536KB의 메모리를 가지고 최대 2048x1536 해상도의 이미지를 처리할 수 있음을 보여준다.

표 1 각 해상도별 메모리 사용량
Table 1 Amounts of memory used for each resolution

해상도	힙메모리 크기	최대 메모리 사용량
320x240	512KB	60KB
640x480	1,024KB	120KB
1600x1200	1,280KB	300KB
2048x1536	1,536KB	384KB

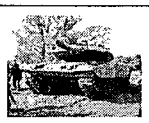
표 2 각 해상도별 인코딩 결과
Table 2 Results of encoding rate for each resolution

해상도	힙메모리	BMP 파일크기	JPEG2000 파일크기	압출율	처리시간
320x240	512KB	226KB	94KB	41.5%	68.54ms
620x480	1,024KB	901KB	304KB	33.7%	82.72ms
1600x1200	1,280KB	5,626KB	1,152KB	20.4%	658.48ms
2048x1536	1,536KB	9,217KB	1,769KB	19.1%	1186.32ms

표 2는 이미지 해상도별로 이미지 파일의 크기와 압축율을 측정한 결과이며, 320x230 이미지와 같이 힙메모리가 어느 정도 여유가 있는 경우, 이미지 압축율이 더 높다는 것을 알 수 있다. 스캔라인 버퍼를 이용하여, 이미지를 몇 개의 블록 단위로 처리하기 때문에, 고해상도 이미지에서는 메모리 자원 제한이 없는 경우보다 압축율이 상대적으로 낮을 수 있다.

표 3은 표 2에서 설정된 임베디드 환경에서 인코딩된 이미지와 메모리 제한을 두지 않은 PC환경에서 인코딩한 이미지의 품질을 비교한 결과이다. PC환경과 같이 여유있는 메모리를 사용할 수 있는 환경에서는 버퍼 메모리 영역을 더 효율적으로 사용할 수 있기 때문에, 상대적으로 더 작은 크기의 이미지를 얻을 수 있으나, 임베디드 환경과 큰 차이는 나지 않았다. 구현된 JPEG 2000코덱은 임베디드 환경에서 최소한의 메모리만 사용하여 원본 이미지와 거의 동일한 품질의 이미지를 제공하는 것을 알 수 있다.

표 3 이미지 품질 비교
Table 3 Comparison with image quality

해상도	원본이미지	ARM9 플랫폼	Windows PC 플랫폼
640x480			
이미지 크기	901KB	304KB	216KB
1600x1200			
이미지 크기	5,636KB	1,152KB	952KB

5. 결 론

본 논문에서는 구현된 JPEG 2000 코덱은 메모리 관리 기능이 같이 포함되므로, 설정된 힙메모리 영역 내에서 인코딩/디코딩을 처리할 수 있으며, 임베디드 환경에서는 최소한의 메모리만 사용하여 원본 이미지와 거의 동일한 품질의 이미지를 제공할 수 있다.

JPEG 2000코덱을 별도의 백-엔드 칩셋(Back-End Chipset)없이 ARM9계열의 임베디드 프로세스에 적용할 수 있으므로, 비용을 절감할 수 있으며, 메모리를 자체적으로 관리하기 때문에 메모리 버퍼를 독립된 프로세서로 처리할 수 있으며 다른 애플리케이션과 충돌 없이 안정적으로 적용할 수 있다. 향후 과제로는 JPEG 2000 코덱의 실시간 이미지 처리에 적합하도록 성능 향상을 위한 코드 최적화하여 실시간 동영상 코덱을 구현하는 것이다.

참 고 문 헌

[1] C. Christopoulos, A. Skodras, and T. Ebrahimi, "The JPEG 2000 Still Image Coding System: an Overview," IEEE Transactions on Consumer Electronics, Vol. 46, No. 4, pp. 1103-1127, November 2000.

[2] M. Boliek, C. Christopoulos, and E. Majani (Eds), "JPEG 2000 Part 1 Final Draft International Standard," (FDIS15444-1), ISO/IEC JTC1/SC29/WG1, N1855, August 2000.

[3] International Organization for Standardization and International Electrotechnical Commission, ISO/IEC 10918-1:1994, Information technology Digital compression and coding of continuous-tone still images: Requirements and guidelines.

[4] "Netpbm Home Page," http://netpbm.sourceforge.net,h 2005

[5] M. Luse, "The BMP file format," Dr. Dobb's Journal, vol. 9, no. 10, pp. 1822, Sept. 1994.

[6] "FOURCC.org Home Page", http://www.fourcc.org/, 2005

[7] H.C. Fang, T.C Wang, C.J. Lian, T.H. Chang, and L.G. Chen,"High Speed Memory Efficient EBCOT Architecture for JPEG2000," in Proc. IEEE International Symposium on Circuits and Systems, Vol. 2, Thailand, pp. 736-739, May 2003.

[8] D. Taubman, "High performance scalable image compression with EBCOT," IEEE Trans. Image Processing, vol. 9, no. 7, pp.1158-1170, July 2000.

[9] D. Taubman, E. Ordentlich, M.J. Weinberger, and G.Seroussi, ""Embedded block coding in JPEG2000,"" Signal Processing: Image Communication 17 (1) (2002) 49.72.

[10] K. Andra, C. Chakrabati, and T. Acharya, "A High-Performance JPEG2000 Architecture," IEEE Trans. on Circuits and Systems for Video Technol., vol.13, no.3, pp. 209-218, March 2003.

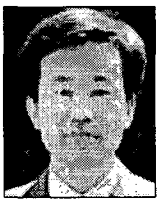
저 자 소 개

조 시 원 (曺 時 元)

1994년 동국대학교 전기공학과 (공학사). 1998년 동국대학교 대학원 전기공학과 (공학석사). 2003년~현재 동국대학교 대학원 전기공학과 박사과정

Tel : 02-2260-3350

E-mail : stsolaris@gmail.com



이 동 욱 (李 東 旭)

1983년 서울대학교 전기공학과 (공학사). 1985년 서울대학교 대학원 전기공학과 (공학석사). 1992년 Georgia Tech.대 (공학박사). 1993년~현재 동국대학교 전기공학과 교수

Tel : 02-2260-3350

E-mail : dlee@dongguk.edu