JPEG을 Animated GIF로 변환하는 과정에서 스레딩에 따른 멀티코어 모바일 디바이스의 성능 평가

우호성*, 김강석*, 김재훈*
*아주대학교 대학원 지식정보공학과
e-mail: dnghtjd86@ajou.ac.kr

Performance evaluation of mobile multicore devices on threading in converting JPEG to animated GIF

Hosung Woo*, Kangseok Kim*, Jai-Hoon Kim*
*Dept. of Knowledge Information Engineering, Ajou University

요 약

본 논문에서는 멀티코어 모바일 디바이스에서 최적의 스레드 구성을 측정하기 위해 이미지 코덱을 사용하여 다양한 환경에서 스레드 개수에 따른 인코딩 수행시간을 분석하였다. 인코딩은 Quantization을 사용하여 JPEG 파일들을 하나의 GIF 파일로 변환하는 기능을 수행하며, 듀얼코어와 쿼드코어 안에서 각각의 스레드 개수를 늘려가며 측정하였다. 듀얼코어에서는 스레드 4개였을 경우가 성능이 효율적이였으며, 쿼드 코어에서는 스레드 3개였을 경우가 성능이 효율적이였다. 분석 후 결론은 스레드 개수와 성능은 비례하는 것이 아니며 성능에 크게 영향을 미치지 않는 것으로 확인되었다. 코어와 I/O입출력의 성능 및 데이터 크기에 따라 적당한 스레드 개수를 정하여 사용하는 것이 효율적이다.

키워드 : Multicore, Thread, JPEG, GIF, Dithering

1. 서론

최근 컴퓨터 및 이동통신 기술의 발달로 모바일 장치가 크게 발전하여 사용자가 기하급수적으로 늘어나면서, 모바일 기기를 활용하는 일이 빈번해 졌다. 응용 방법으로는 이미지, 영상과 같은 멀티미디어 데이터 처리가 주를 이루고 있고, 이와 같은 고용량 데이터 처리는 모바일 장치에서 큰 시스템 부하를 야기한다.[2] 따라서 효율적인 멀티미디어 데이터를 처리하기 위해 대부분 압축하여 저장 혹은 전송을 하게 된다. 데이터 압축 방법에 따라 손실압축과 무손실 압축으로 분류 되는데 본 논문에서 사용되는 GIF의 이미지 데이터 압축방법은 무손실 압축으로 LZW를 사용하고 있다.[6] GIF의 경우 최대 8비트(256가지)의 제한된 색으로 이미지를 표현할 수 있다. 따라서 Dithering을 사용하여 좀 더 다채로운 색감을 주어 품질이 좋은 결과물을 출력할 수 있도록 하고 있다.

본 연구는 지식경제부 및 한국인터넷진흥원의 “고용계약형 지식정보공학 석사과정 지원사업”의 연구 결과로 수행되었음.

본 논문에서는 JPEG 프레임을 GIF로 변환 시 오버헤드가 큰 이미지 데이터 압축 부분과 Dithering 부분의 수행시간을 분석하고, 각기 다른 멀티코어 환경에서 스레드 개수에 따른 수행시간을 분석하여 최적화하는 기법을 제안한다.

본 논문의 구성은 다음과 같다. 먼저 2장에서는 Multicore와 JEPG 및 이미지 코덱의 결과물인 GIF, GIF의 제한된 색상을 극복할 수 있는 Dithering에 대하여 살펴보고 3장에서는 멀티코어에서의 스레드 개수에 따른 성능측정 결과를 비교 분석하였다. 마지막으로 4장에서는 결론을 맺는다.

2. 관련연구

2.1 Multicore

멀티 코어는 두 개 이상의 독립 코어를 단일 직접 회로로 이루어진 하나의 패키지로 통합한 것으로 칩 레벨 멀티프로세서라고도 한다.

멀티코어 프로세서의 장점은 중앙처리 장치(CPU)가 2개 이상 들어간 것과 마찬가지이기 때문에 멀티 코어 프로세서를 지원하는 프로그램으로 작업할 때, 싱글 코어 프로세서에 비해서 빠른 작업을 할 수 있고 동영상의 인코딩, 포토샵 작업, 높은 사양의 게임과 같은 작업에 성능이 발휘된다.[9][11]

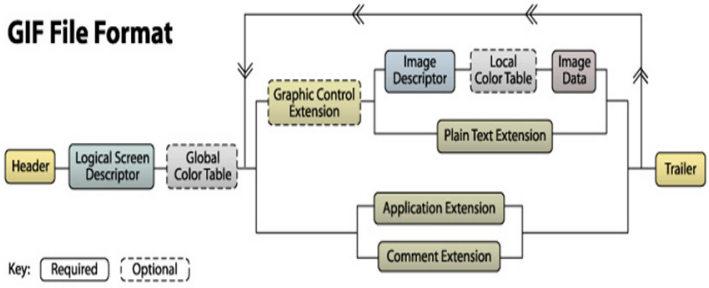
2.2 JPEG

JPEG(Joint Photographic Experts Group)은 정지화상을 위해서 만들어진 손실 압축기법을 사용하는 국제 표준으로 CCITT(Consultatve Committee International Telegraphand Telehpone)와 ISO에서 제정하였다. JPEG는 이미지를 작은 블록으로 나누어 많은 양의 이미지 정보를 줄이는 DCT(Discrete Cosine Transformer) 알고리즘에 기초를 두고 있으며. 압축률을 늘이기 위해서는 보다 많은 양의 이미지 정보를 지우고, 이에 따라 세세한 부분의 이미지가 줄어들기 때문에 이미지의 품질은 낮아진다. 손실 압축 형식이지만 파일 크기가 작기 때문에 웹에서 널리 쓰인다.[7][10]

2.3 GIF

GIF(Graphics Interchange Format)는 비트맵 그래픽 파일 포맷으로 1987년 컴퓨서브가 발표하였다. 월드 와이드 웹에서 가장 널리 쓰이는 파일 포맷이기도 하며, 특별한 플러그인을 필요로 하지 않기 때문에 다양한 환경에서 쉽게 사용할 수 있다.[7]

GIF는 그래픽이나 로고 등의 색깔이 단순한 이미지를 구현하기에 적합한 형식이며, 이미지를 투명하게 만드는 Transparent 기법, 고정된 각각의 이미지 파일을 재생시켜 움직이는 화면이 가능하게 할 수 있어 비교적 짧은 시간의 애니메이션 제작이 가능하다는 게 큰 장점이다. 또한 비손실 압축기법인 LZW 알고리즘을 사용한다.[6] 높은 압출률로 통신상에서 빠른 전송속도를 가지며 이미지의 손실 없이 압축이 가능하다. 하지만 최대 256 색까지 지원한다는 단점이 있다. 따라서 색상수가 많은 이미지의 경우 최대 24비트(16,722,216)의 색상을 표현 할 수 있는 JPEG을 주로 사용한다. 많은 색상이 사용된 사진을 GIF로 저장할 경우 256가지 외에 색은 전부 표현이 안되므로 손실이 전혀 없다고 할 수는 없지만 이를 보완하기 위해 Dithering 기법을 사용하여 색의 제한을 극복하기도 한다.



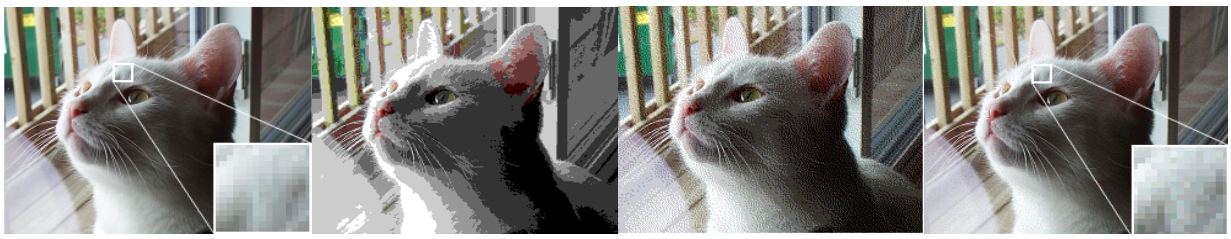
<그림 1> GIF 파일 포맷

2.4 Dithering

Dithering 기법은 컴퓨터 그래픽스에서 표시 장치나 인쇄기의 해상도를 초과하는 다계조 색의 화상을 제한된 색상으로 조합 또는 비율을 변화하여 새로운 색을 만드는 작업이다.[7]

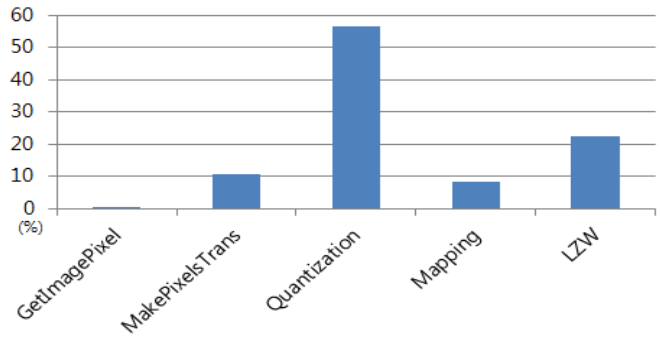
24비트의 이미지를 256색상으로만 표현할 수 있는 시스템(GIF)에서 볼 때 256색상에 존재하지 않는 색을 표현하기 위한 방식으로 저해상도에서 컴퓨터 도형 처리의 사실감을 높이고 매끄럽지 못하고 계단 모양으로 울퉁불퉁한 윤곽선이나 대각선을 눈에 띄지 않게 하기 위해 사용된다.[1]

본 논문에서는 Quantization안에 Dithering과정이 포함되어 있다.



<그림 2> 원본사진 VS. Dithering을 하지 않은 사진 VS. 216색만 플로이드-스타인버그 Dithering을 한 사진 VS. 최적화된 팔레트를 이용하여 플로이드-스타인버그 Dithering을 한 사진[7]

3. 멀티코어에서 스레드 개수에 따른 성능측정 및 분석

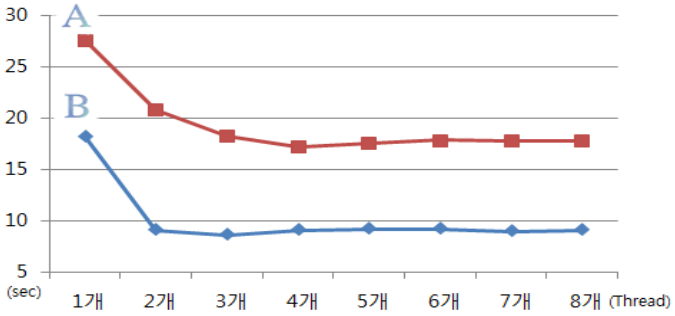


<그림 3> 전체 인코딩 비율

<그림 3>의 GetImagePixels은 JPEG 파일로부터 픽셀의 정보를 가져오는 역할을 한다.
MakePixelsTrans는 픽셀에 대해 투명도 검사를 하며, Quantization은 Dithering을 수행하기 위한 과정을 포함한다. Mapping은 Quantization과정이 끝난 후 RGB형식에 맞게 변환을 해주며, LZW는 파일을 출력하기 전에 이미지 데이터를 압축하는 과정을 포함한다.
GetImagePixels를 제외한 과정들은 병렬처리가 가능한 부분으로 본 논문에서 스레드가 동작하는 부분들이다.

스레드 / 코어	1개	2개	3개	4개	5개	6개	7개	8개
A. 듀얼	27.51	20.73	18.22	17.19	17.54	17.82	17.77	17.8
B. 쿼드	18.08	9.05	8.58	9.03	9.18	9.12	8.98	9.04

<표 1> 스레드 개수에 따른 전체 인코딩 시간
(단위 : sec.)

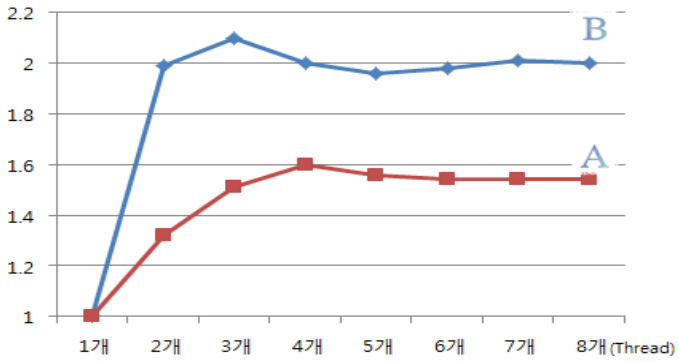


<그림 4> 스레드 개수에 따른 전체 인코딩 시간
(A. 듀얼코어 B. 쿼드코어)

A는 듀얼코어 모바일 디바이스에서 인코딩 시간을 측정한 것이다. 스레드가 4개였을 경우 가장 빠른 성능을 확인할 수 있다. 4개 이후에는 오히려 약 0.5ms 내의 지연이 발생하는 것을 확인할 수 있고, 스레드 4개는 스레드 1개였을 경우보다 처리속도가 약 1.6배 빨라졌다.

B는 쿼드코어 모바일 디바이스에서의 인코딩 시간을 측정한 것이다. 스레드가 3개였을 경우 가장 빠른 성능을 확인 할 수 있으며, 3개 이후에는 성능이 크게 증가하지 않는 것을 확일 할 수 있다.
쿼드코어 안에서 스레드 3개였을 경우 스레드 1개일 경우보다 약 2.11배 빨라졌다.

$$Speedup = \frac{\text{단일처리시간}}{\text{병렬처리시간}} = \frac{t\_single}{t\_multi}$$



<그림 5> 스레드 개수에 따른 병렬처리의 성능

듀얼코어와 쿼드코어에서 가장 성능이 좋은 시간을 비교하면 쿼드코어가 듀얼코어에 비해 약 2배 빠른 것을 확일 할 수 있다.

4. 결론

본 논문은 이미지 코덱을 사용하여 150장의 JPEG 프레임을 한 개의 GIF 파일로 변환하는 과정에서 코어 환경에 따라 각각의 스레드 개수를 증가시켜가며 전체 실행시간을 측정하였다. 50~100KB의 JPEG 파일의 파일이 무수히 많은 경우 하나의 스레드에서 작업한다면 오래 걸릴 것이다. Main 스레드가 여러 개의 스레드를 생성하여 작업을 수행한다면 속도가 빨라져 시간을 단축할 수 있다. 하지만 스레드가 너무 많아지면 속도가 빨리질 수 있지만 응답 속도가

떨어지기 때문에 문제가 발생할 수도 있다. 따라서 파일의 크기와 하드웨어의 성능을 비교하여 적당한 스레드 개수를 정해야 한다. 스레드가 필요할 때마다 생성해서 사용하기 보단 스레드풀을 이용하여 사용하고 자원이 반납되어진 대기 중인 스레드를 사용하면 효율성이 높아진다.

듀얼코어에서는 스레드 4개를 생성하였을 경우, 쿼드코어에서는 스레드가 3개였을 경우 그 이상의 스레드를 생성했을 경우보다 성능이 빠르고 효율적임을 확인 할 수 있다.

이번 실험에서는 적당한 스레드 개수 이상의 스레드를 사용 할 경우 약 0.5ms의 지연이 있었고, 이 지연은 사용되지 않는 스레드들이 대기 상태에 빠져 생긴 것으로 간주된다. 스레드 개수는 전체 성능에는 크게 영향을 미치지 않는 것으로 확인되었다.

참고문헌

[1] RoBert A. Ulichney “Dithering with Blue Noise” Proceedings of the IEEE Vol. 76, Issue 1, pp.56-79 January 1988

[2] Kangsuu You, Hanjeong Lee, Euee S.Jang, Hoonsung Kwak “An Efficient Lossless Compression Algorithm using Arithmetic coding for indexed Color image” KCI Vol. 30, pp.35-43, January 2005

[3] I. Pitas, “Digital Image Processing Algorithms and Applications” Jone Wiley & Sons, Inc., April 2000

[4] Xiaolin Wu, Nasir Memon “Context-Based, Adaptive, Lossless Image Coding” IEEE Transactions on Communications, Vol. 45 Issue: 4, pp.7-444, April 1997

[5] James D. Murray and William Vanryper, “Graphics File Formats” Reilly & Associates, Inc., April 1996.

[6] John Miano, “Compressed Image File Formats” pp.171-188, August 1999

[7] Dithering, http://en.wikipedia.org/wiki/Dither

[8] GIF Reference, http://www.martinreddy.net/gfx/2d/GIF87a.txt

[9] Multicore, http://en.wikipedia.org/wiki/Multi-core_processor

[10] Jungho Lee, Hoonjun Ko, Changmo yang, Wonhee You, “The thread scheduling method based on the priority of threads on the multithread models” Vol. 27 No. 2, October 2000

[11] Sungho Yun, Junsang Park, Myungsub Kim, “Performance Improvement of a Real-time Traffic Identification System on a Multi-core CPU Environment” KCI, Vol. 37B, No. 05, pp.348-356, June 2012