Journal of Korea Multimedia Society Vol. 15, No. 5, May 2012(pp. 624-631)

GPGPU를 이용한 고속 의료 볼륨 영상의 압축 복원

계 희 원†

요 약

많은 의료영상 시스템에서 의료 볼륨 데이터는 압축된 형태로 저장되어 있으며, 압축된 데이터는 가시화 이전에 압축 복원을 수행해야 한다. 압축 복원은 상당한 시간이 소모되기 때문에 본 연구는 삼차원 의료영상의 고속 복원 방식을 제안한다. 제안 방법은 의료영상의 특수성에 대한 사용자 요구를 감안하여, 손실과 무손실 압축을 모두 제공하며 점진적 개선(progressive refinement) 복원 속성을 갖는다. 그리고 그래픽스처리장치(GPU)를 이용한 병렬화를 수행하여 매우 짧은 시간 내에 압축 복원이 수행된다. 마지막으로 압축 복원과 볼륨 가시화를 연계하여 선택적 압축 복원 방법이 가능하며, 이를 통하여 볼륨 압축 복원의 추가적 성능 향상을 얻었다.

Fast Medical Volume Decompression Using GPGPU

Heewon Kye†

ABSTRACT

For many medical imaging systems, volume datasets are stored as a compressed form, so that the dataset has to be decompressed before it is visualized. Since the decompression process takes quite a long time, we present an acceleration method for medical volume decompression using GPU. Our method supports that both lossy and lossless compression and progressive refinement is possible to satisfy variable user requirements. Moreover, our decompression method is well parallelized for GPU so that the decompression takes a very short time. Finally, we designed that the decompression and volume rendering work in one framework so that the selective decompression is available. As a result, we gained additional improvement in volume decompression.

Key words: Medical image decompression(의료영상 압축복원), embedded zerotree wavelet(내장형 0-트리 웨이블릿), CUDA(쿠다), volume rendering(볼륨 가시화)

1. 서 론

볼륨가시화는 볼륨데이터를 가공하여 유용한 영상 정보를 생성하는 기법이다[1]. 볼륨가시화는 주로 CT나 MR등의 삼차원 의료영상 장비에서 획득한 삼차원 의료영상을 관찰하는 방법으로 널리 사용되고 있다. 현재 다수의 의료영상 시스템에서 볼륨가시화를 수행하는 작업은 다음과 같다. 우선, 영상 서버에 저장된 의료 볼륨데이터를 진단을 수행하는 가시화 클라이언트로 전송한다. 이때, 의료영상은 보안, 저장 용량, 관리의 편리를 위하여 별도의 저장 서버에 보관되어 있는 것이 일반적이다. 볼륨데이터의 전송이 완료되면 클라이언트는 가시화 소프트웨어를 사용하여 필요한 가시화를 수행하게 된다.

영상의 용량이 커지면, 서버에서의 저장 공간과 클라이언트로의 영상 전송 시간을 절감하기 위해 삼

※ 교신저자(Corresponding Author) : 계희원, 주소 : 서울특별시 성북구 삼선동2가 389 한성대학교 정보시스템공학과(136-792), 전화 : 02)760-8014, FAX : 02)760-5886, E-mail : kuei@hansung.ac.kr

접수일 : 2011년 10월 20일, 수정일 : 2012년 1월 7일
완료일 : 2012년 3월 16일
† 정회원, 한성대학교 정보시스템공학과 조교수
※ 본 연구는 한성대학교 교내연구비 지원과제임.

차원 의료영상을 압축된 형태로 저장, 전송하게 된다. 따라서 클라이언트는 전송된 압축 영상을 복원한 이후 볼륨가시화를 수행한다. 대용량 볼륨데이터의 압축 복원은 오랜 시간이 소요되기 때문에, 이를 줄이기 위한 노력이 필요하다. 영상 압축과 복원은 매우 오랫동안 광범위하게 다루어진 주제이나, 많은 기존연구는 동일한 화질에 대해 더 우수한 압축률을 얻는 기법에 초점이 맞추어져 있다[2,3]. 본 연구의 목적은 삼차원 의료영상 시스템에 적합한 고속 의료영상 압축 복원 방법을 제시하는 것이다.

의료영상 처리의 목적은 질환을 찾아내는 것이므로 고주파 영역과 특이점이 주요 관심대상이다. 따라서 일반 영상 압축 방법인 손실 압축으로 고주파 성분을 제거하게 되면, 임상 진단에 오류를 만들 수 있다. 따라서 본 연구는 무손실 압축을 지원하려 한다. 기존의 많은 고속 압축 방법은 벡터 양자화를 이용한 코드북(codebook) 방식이다[4]. 코드북은 벡터 양자화를 통해 자주 발생하는 영상 조각을 사전 형식으로 나열한 것으로, 압축하고자 하는 영상 조각을 가장 유사한 사전의 페이지 번호로 치환하여 압축을 수행한다. 관련연구[5]와 같이 시간 변화를 포함한 사차원 데이터에 대해서 팔레트를 생성하고 팔레트 에니메이션을 수행하거나, 코드북을 이용하는 방식은 영상 조각의 복원이 사전 참조 한번으로 종료되므로 속도가 매우 빠르다는 장점이 있다. 그러나 이러한 방법으로 무손실 압축을 수행하려면 모든 경우의 수를 사전에 수록해야 하므로 압축 효율이 매우 나빠지고 압축이 불가능해지는 단점이 있다. 그밖에 영상 손실 없이 데이터를 줄이는 방법으로, 전처리로 투명한 부분을 모두 메모리에서 제거하고 남는 부분을 재배치 하는 방법[6]이 있으나, 사용자가 투명도를 한 번 결정하면, 이후 변경하지 못한다는 단점이 있다.

본 연구는 무손실 압축을 지원하기 위해, 연산량이 많지만 영상의 중복된 정보를 추출하여 압축을 수행하는 엔트로피 기반 방법[7-9]을 적용하려 한다. 영상을 삼각함수 또는 웨이블릿을 이용한 주파수 변환을 하여 주성분과 잔차 성분을 분리한 다음, 잔차 성분의 중복된 정보를 제거하면 압축 효율을 향상시킬 수 있다. 그리고 엔트로피 기반 방법은 오랜 시간이 소요되므로 고속 병렬 처리가 가능한 그래픽스 처리장치(graphics processing unit, GPU)를 이용하여 시간을 단축하려 한다.

GPU에는 수백 개에 달하는 병렬 처리 장치가 내장되어 있으며[10], 각 장치에 대해 사용자는 논리적 작업 단위인 쓰레드를 할당하여 병렬 연산을 수행할 수 있다. 각 쓰레드는 병렬 수행되므로, 본 연구는 쓰레드간 데이터 종속성이 없고 부하 관리(load balancing)가 원활하도록 시스템 설계를 수행하였다. 또한 그래픽스 처리장치를 효율적으로 사용하기 위하여, 압축 복원이 볼륨가시화를 위해 사용되는 점에 착안, 가시성 검사를 이용한 선택적 블록 복원 방법을 제안한다.

본 연구에서 제안하는 시스템은 다음의 속성을 만족한다. 우선 손실 압축뿐만 아니라 무손실 압축과 복원을 지원한다. 그리고 데이터 용량이 너무 크거나 전송 속도가 너무 느린 경우, 경우에 따라 우선 낮은 화질의 복원 영상이 생성되나 시간이 지나며 개선될 수 있다. 즉, 고속의 손실 복원과 점진적 개선이 가능하다. 그리고 삼차원 가시화에 적합한 압축 구조를 갖추어 고속 압축 복원을 가능하게 한다. 이후 2장에서 본 연구에서 사용하는 압축 방법인 내장형 0-트리 웨이블릿 변환에 대해 요약하고 3장에서 본 연구의 제안 방법에 대해 설명하며 4장에서 실험 결과를 보이고 5장에서 결론을 맺는다.

2. 내장형 0-트리 웨이블릿 변환

본 연구는 삼차원 의료영상의 압축과 복원을 위하여 내장형 0-트리 웨이블릿 변환[11] (embeded zerotree wavelet, EZW)을 GPU 기반으로 구현하고 볼륨 가시화와 연계한 고속화 방법을 제안한다. EZW의 특징은 중요한 계수가 먼저 부호화되고 작은 계수는 이후에 부호화되기 때문에, 압축 복원 과정에서 중요 계수의 우선 복원이 가능하다. 따라서 빠른 손실 복원을 수행할 수 있으며, 이후 계수를 더 전송하고 복원하면 무손실 복원도 수행할 수 있다. 유사한 압축 방법으로 SPIHT[8]나 EBCOT[9] 같은 방법이 있으며, 이들의 압축률이 EZW에 비해 높은 것으로 알려져 있으나[3], 알고리즘이 더 복잡하여 복원하는데 시간이 오래 걸린다는 단점이 있어 선택하지 않았다.

EZW는 웨이블릿 변환된 영상의 자기 반복성을 이용하여 부모 노드 값의 변화가 적으면 자식 노드에서도 변화가 적을 것이라는 관찰을 응용한 방법이다. 상세한 내용은 참고문헌[11]에 소개되어 있으며, 본

연구는 그 개요만을 간단하게 설명한다. EZW에서 블록을 압축하는 방법은, 설정된 문턱 값보다 큰, 중요한 계수 값을 순차적으로 부호화하는 것이다. 예를 들어 그림 1에서 초기 문턱 값이 32로 지정된 경우 32보다 큰 값만 부호화되어 63, -34, 49, 47이 순차적으로 검색되어 P(양수) N(음수) P P 로 부호화된다. 그리고 문턱 값보다 작은 값은 Z (zero)로 부호화되는데, 영상을 사진 트리(quad-tree) 형태로 파악했을 때 모든 노드가 Z인 부분 트리(sub tree)를 T(tree root)로 대치하여 부호 수를 줄이게 된다. 추출된 각 부호는 P, N, Z, T중 하나이므로 각 부호에 대해 2bit를 부여하여 저장하고, 문턱 값보다 큰 중요 부호의 정밀도 부분을 추가로 저장하면 지정된 문턱 값에서의 처리가 종료된다. 그리고 문턱 값을 반씩 줄여 나가며, 나머지 값을 연속적으로 부호화 하면 상세 정보들이 차례로 부호화된다. 본 연구에서 EZW의 계수는 문턱 값이 1이 될 때까지 반복 수행하여 저장되므로, 무손실 압축과 복원이 가능하다.

EZW의 복원과정은 순차적으로 저장되어 있는 부호 P, N, Z, T의 배치와, 중요 부호의 정밀도 값을 이용하여 수행된다. 복원은 마찬가지로 큰 문턱 값부터 수행되는데, 문턱 값이 1보다 큰 상태에서 도중에 중단하면 원래 영상이 아닌 근사값이 복원된다. 이를 이용하여 빠른 시간에 손실 복원을 수행하고자 하는 경우, 문턱 값을 크게 유지한 상태에서 복원을 중지하면 된다.

63 P	-34 N	49 P	10 T	7 Z	13 Z	-12	7
-31 Z	23 T	14 T	-13 T	3 Z	4 Z	6	-1
15 T	14 Z	3	-12	5	-7	3	9
-9 T	-7 T	-14	8	4	-2	3	2
-5	9	-1 Z	47 P	4	6	-2	2
3	0	-3 Z	2 Z	3	-2	0	4
2	-3	6	-4	3	6	3	6
5	11	5	6	0	3	-4	4

그림 1. EZW 압축의 예제

3. 볼륨 데이터의 압축과 복원 과정

제안 알고리즘의 전체 구성을 그림 2에 나타내었다. 삼차원 영상의 복원은 압축 과정의 역순으로 이루어지므로 본 논문에서는 압축 과정을 위주로 설명하며 복원 과정은 역순으로 이해할 수 있다. 압축과정은 시간적 여유가 있으므로, GPU가 장착되지 않은 곳에서도 압축 수행이 가능하도록 CPU상에서 동작한다. 복원과정은 고속의 GPU상에서 수행되도록 설계 및 구현 하였으며 이후 각 단계에 대해 상세하게 설명한다.

3.1 블록화 및 이차원 지역 매핑

삼차원 볼륨 영상은 블록 단위로 압축된다. 전체 볼륨 데이터를 작은 조각의 블록으로 분해하면 데이터 분포의 지역성이 좋아지고 압축 효율이 향상되기 때문이다. 그러나 블록의 크기가 너무 작으면 전체 블록의 개수가 증가하여, 각 블록마다 저장해야 하는 헤더 크기를 고려하면 압축 효율이 저하된다. 본 연구는 압축률과 수행효율을 고려하여 하나의 블록을 8×8×4의 크기로 정의하였고 이를 타일로 배열하여 16×16으로 변형하였다. 그림 3에서 B로 표현된 하나의 블록은 8×8×4의 해상도이며, 하나의 블록을 구성하는 4개의 8×8 영상을 이차원으로 나란히 배열하면 그림 3의 우측과 같은 16x16 크기의 영상 1개를 생성할 수 있다. 기존 연구의 많은 이차원 영상 압축에서 블록의 크기는 8x8의 타일을 사용하지만, 삼차원 데이터는 평면적 분포 외에 공간적 분포도 고려하여야 하기 때문에 블록의 크기가 약간 크게 설계되었다.

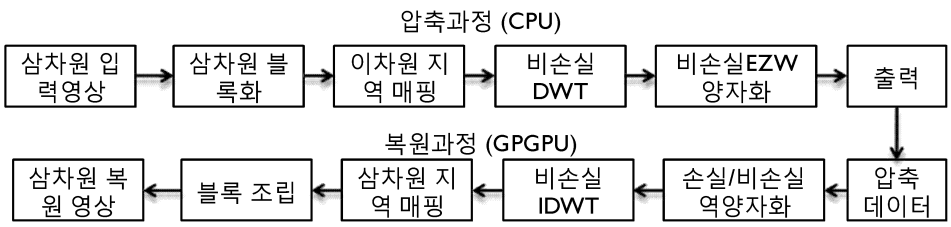

그림 2. 제안 시스템의 순서도

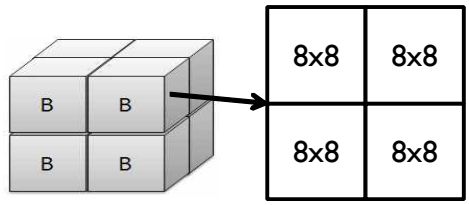

그림 3. 블록화 및 이차원 지역 매핑

이후 생성된 각 타일 영상에 웨이블릿 변환이 적용된다. 한편, 압축의 복원은 이차원 타일 데이터를 볼륨 데이터의 일정 영역에 값을 복사하는 과정이며, 각 블록간에는 교집합이 없으므로 병렬화하기 유리하다.

3.2 웨이블릿 변환 및 EZW 과정

각 타일 영상은 웨이블릿 변환을 통해 저주파 성분과 고주파 성분으로 분리된다. 연구 목표는 속도가 빠르면서도 화질 손실이 발생하지 않는 것이다. 우선, 가장 간단하고 빠른 웨이블릿 변환인 Haar 웨이블릿을 선택하였다. 다만, Harr 웨이블릿은 변환 과정에서 화질 손실이 발생하는 경우가 있는데, 연산과정에서 발생한 실수 값을 정수로 변환할 때 오차가 생기거나 양자화(quantization)과정을 거치기 때문이다. 본 연구는 무손실 압축을 지원하기 위해 양자화 없이 (식 1)과 같은 정수형 무손실 변환을 사용하였다. (식 1)에서 부모 노드 j+1단계의 잔차 부분 d와 신호 부분 s는 자식 노드 j단계에서 얻을 수 있다. 복원 과정은 (식 2)를 반복적으로 적용하면 손실 없이 역변환이 구현된다.

$$d_{j+1,i} = s_{j,2i+1} - s_{j,2i}$$
$$s_{j+1,i} = s_{j,2i} + \lfloor \frac{d_{j+1,i}}{2} \rfloor \quad (1)$$

$$s_{j-1,2i} = s_{j,i} - \lfloor \frac{d_{j,i}}{2} \rfloor$$
$$s_{j-1,2i+1} = d_{j,i} + s_{j-1,2i} \quad (2)$$

주파수 영상으로 변환된 결과는 EZW 변환을 거쳐 출력되며, 이후 역순으로 복원된다. 복원 과정은 EZW 와 웨이블릿 변환 모두 각 블록에 대해 연산이 독립적이므로 병렬처리가 가능하다.

3.3 GPU 병렬화

본 연구는 GPU를 이용한 병렬처리를 구현하기 위해 nVidia의 GPGPU 언어인 CUDA[12]를 이용하여 병렬화를 수행하였다. 영상압축은 매우 복잡한 연산 단계를 거치기 때문에, 웨이블릿 변환과 같이 부분적으로만 GPU를 이용하는 경우[13]가 아니라면, DirectX[14]나 OpenGL과 같은 그래픽스 API의 확장보다 직접적인 GPGPU 언어의 사용이 적합하다. 병렬처리를 수행하기 위해 CUDA는 쓰레드(thread)와 쓰레드 블록(thread block)의 논리적 계층구조를 제공하여 병렬화를 쉽게 처리한다.

CUDA에서 가장 작은 논리적 병렬 작업단위가 쓰레드이며, 하나의 쓰레드는 주어진 명령들을 순차적으로 수행한다. 사용자는 그림 4와 같이 일정한 수의 쓰레드를 하나의 쓰레드 블록으로 지정할 수 있다. 같은 쓰레드 블록에 속한 쓰레드는 동기화와 공유 메모리를 통한 정보 교환이 가능하다. 그리고 전체 작업은 다수의 쓰레드 블록으로 구성된다. 각 쓰레드 블록은 GPU에 내장되어 있는 물리적 병렬처리 장치인 MP(multi-processor)에 할당되어 병렬적으로 수행된다.

본 연구는 효과적인 병렬화를 통해 알고리즘의 성능을 향상시키려 한다. 영상 압축의 복원과정은 블록에 대해 독립적으로 수행되므로 각 블록에 대한 복원을 단일 쓰레드로 구성하였다. 다음으로 쓰레드 블록을 구성하기 위해 하나의 쓰레드 블록에 속하는 쓰레드의 개수를 결정하는 문제를 해결해야 한다. 하나의 MP에는 제한된 용량의 레지스터(register)가 있기 때문에 하나의 쓰레드 블록을 구성하는 쓰레드의 수

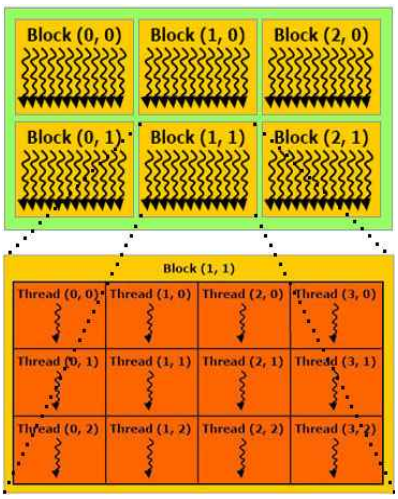

그림 4. CUDA의 병렬 계층 구조[12]

가 너무 많으면 레지스터의 부족으로 성능 저하가 발생하고 심하면 수행이 불가능하게 된다. 반대로 쓰레드 블록을 구성하는 쓰레드의 수가 너무 적으면 MP를 구성하는 처리장치가 유휴상태가 되므로 효율이 감소하게 된다. 본 연구는 쓰레드 블록당 쓰레드의 개수를 변화해가면서 성능 변화를 관찰하고 최적점을 찾아, 이후 연구에서 적절한 쓰레드의 수를 결정하는 데 참고가 되도록 하였다.

3.4 볼륨가시화와 연계한 효과적인 압축 복원

복원된 볼륨 데이터는 볼륨가시화에 이용되며, 볼륨가시화 역시 GPGPU에서 수행된다. 본 연구에서는 일반적으로 가장 많이 사용되는 볼륨가시화 알고리즘인 광선추적법[1]을 적용하였으며 각 광선에 대해 쓰레드를 할당하여 병렬화를 수행[15]하였다. 본 연구는 압축 복원과 가시화가 모두 GPU에서 수행되므로 복원된 데이터를 추가로 전송할 필요가 없어 효율적이다. 만약 CPU로 압축을 복원한다면 복원된 데이터를 GPU로 전송하는 버스 전송시간이 추가로 필요하게 된다.

한편, 볼륨가시화를 가속화하는 중요한 방법으로 빈공간 도약 (empty space leaping)방법이 있다[16, 17]. 빈공간 도약은 전처리(pre-processing)를 통해 볼륨 데이터 내부의 투명하다고 생각되는 지역의 위치 정보를 저장하여 놓고, 가시화 단계에서 투명한 지역의 샘플링을 생략하여 속도를 향상하는 방법이다. 볼륨 데이터의 일부분이 투명한지 여부는 사용자가 지정하는 불투명도 전이함수(opacity transfer function)에 따라 결정된다[1]. 사용자는 가능한 밀도값의 범위에 대해 투명도를 정의하는데, 예를 들어 인체의 골격계를 관찰하고 싶다면, 근육과 피하지방에 해당하는 밀도값에 불투명도 0을 지정한다.

본 연구에서는 각 블록의 밀도 최댓값을 전처리로 구하여 저장하여 놓는다. 가시화 단계에서 빈공간 도약을 수행하는데, 각 블록의 밀도범위의 불투명도가 0으로 파악되면 빈공간 도약을 수행하게 된다. 이때, 빈공간 도약 기법을 압축 복원과 연계하여 고려하면, 투명한 블록은 압축 복원에서 제외해도 화질에 영향을 주지 않는다는 결론을 얻는다. 이에 착안하여 본 연구는 불투명한 것으로 인정되는 블록만을 복원하여 성능을 향상시킨다.

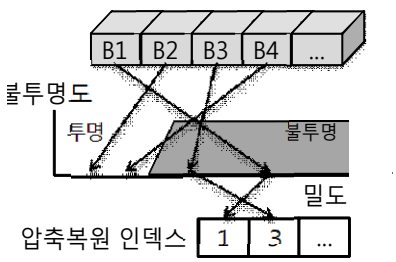

그림 5. 불투명도 전이함수와 연계한 선택적 압축 복원

이러한 선택적 복원을 구현하기 위하여, 본 연구는 각 블록의 최댓값을 압축된 데이터의 헤더에 포함시키고, 사용자가 정의한 불투명도 전이함수에 대해 헤더를 분석하여 압축 복원 여부를 미리 판단할 수 있게 하였다. 이후 불투명도 전이함수가 변경되면 이에 대응하여 추가로 압축을 복원하면 되므로 화질 저하는 발생하지 않는다.

이러한 착상을 GPGPU로 구현하는 것은 추가적인 고려가 필요하다. GPGPU는 병렬 계산 장치이므로, 다른 데이터에 대해 동일한 명령을 수행하는 병렬 수행 단위(warp)가 존재한다[12]. 하나의 병렬 수행 단위에서 일부의 블록만 압축을 복원 한다면, 복원할 필요가 없는 투명한 블록을 담당하는 연산장치는 동기화를 위해 대기하도록 설계되어 있다. 따라서 본 연구는 GPU 연산장치의 활용도를 높이기 위해, 블록의 주소지정을 간접적 방식으로 변경하였다. 그림 5와 같이 압축복원 인덱스를 생성하여 불투명하다고 판단되어 압축 복원을 수행해야 할 블록의 주소만을 따로 저장한 후, 압축복원 인덱스에 저장된 주소의 블록만을 GPGPU에서 복원하여 성능을 더욱 향상시킨다.

4. 실험결과

실험은 Intel의 Core2Duo CPU와 nVidia의 GeForceGTX 470 GPU를 장착한 개인용 컴퓨터에서 수행되었다. GeForceGTX 470은 14개의 MP를 내장하고 있으며 448(=14x32)개의 병렬처리 단위와 1.2GB의 전용 메모리를 가지고 있다. 소프트웨어 구현은 Visual Studio C++를 사용하였고, GPGPU 개발을 위해 CUDA SDK 3.2를 이용하였다. 볼륨 데이터는 256×256×225 크기의 머리 데이터와 512×512×300 크기의 복부 데이터를 사용하였고, 각 데이터는 16-비트 정밀도를 갖도록 조정하였다.

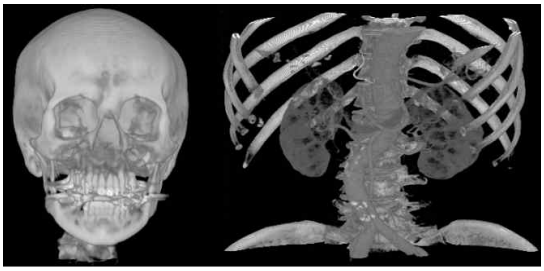

그림 6. 불투명도 전이함수와 연계한 선택적 압축 복원

무손실 압축 복원을 사용한 가시화 결과를 그림 6에 제시하였으며, 이 결과는 압축하지 않은 데이터의 가시화 결과와 완전히 동일하였다. 그리고 GPU 병렬화를 이용한 압축 복원에 소요된 시간을 측정하였다. 3.3절에서 설명한 바와 같이 CUDA의 병렬 계층구조에서 쓰레드 블록당 쓰레드의 수(thread per block, TPB)를 변화시켜가며 압축 복원 시간을 측정하여 표 1에 제시하였다. 또한 CPU 기반의 EZW 압축복원 시간 또한 측정하여 함께 제시하였다.

본 실험에서는 TPB가 64일 경우 최적의 성능을 보이는 것을 관찰할 수 있다. 이때의 결과는 CPU로 압축 복원을 수행했을 경우에 비하여 머리 데이터는 대략 36.9배(=4437/120.1), 복부 데이터는 대략 44.7배(=65865/1472.7) 정도의 성능 향상을 나타낸다. EZW 압축 복원을 GPU로 수행한 기존 연구가 없기 때문에, 이 결과를 관련 연구와 직접 비교하기는 어려운 상황이다. 다만, 허프만 비손실 압축을 GPU로 구현한 관련연구[18]에서 GPU 병렬화를 통해 CPU보다 35-50배의 속도 향상을 얻은 것으로 보고되었고, 배정도 실수형(double precision) 데이터에 대한 비손실 압축 성능을 비교한 기존 연구[19]에서 320개의 고속 코어를 사용한 GPU가 8개 코어의 병렬 CPU에 비해 4.5배 빠른 것으로 보고되어, 본 연구의 GPU를 적용하여 얻은 성능 향상은 적절한 수준으로 판단된다.

표 1에서 확인할 수 있듯이, TPB의 크기에 따라 성능이 크게 변화함을 확인할 수 있다. TPB가 하나의 MP를 구성하는 연산장치의 수인 32보다 작은 경우 병렬처리 단위의 일부가 유휴상태가 되어 성능이 크게 저하된다. TPB가 32보다 커지는 경우, 전역 메모리 접근 등의 연산으로 연산상치가 유휴상태일 때, 다른 작업으로 문맥 교환이 일어나므로 성능이 더욱 향상된다. 그러나 지나치게 큰 TPB에서는 레지스터 부족으로 성능이 저하된다.

본 연구에서 압축 복원은 레지스터를 많이 사용하는 복잡한 연산이므로, 최적의 TPB는 MP를 구성하는 연산장치의 수인 32에 근접하게 결정되는 것을 확인할 수 있다. 향후 연구에서 각 쓰레드가 수행하는 알고리즘의 특성이 본 연구에 비해 간단해진다면 최적의 TPB를 64보다 증가시키는 것이 좋다는 결론을 얻는다. 한편 TPB가 너무 작으면 전체 블록의 개수가 증가하기 때문에 용량이 상대적으로 큰 복부 데이터는 한 번에 압축복원을 수행할 수 없게 되어 수치를 기록하지 않았으며 (N/A), 이후 실험에서는 가장 효율이 좋은 TPB인 64를 기준으로 실험을 수행하였다.

EZW는 복원과정을 중간에 마치면 손실 복원을 상대적으로 빠른 시간에 수행할 수 있다. 오차 한계인 중단 문턱값을 1과 2로 설정하여 복원 시간을 측정하여 표 2에 제시하였다. 화질 손실 정도에 따라 1.5배까지 점차적으로 속도가 향상되는 것을 확인할 수 있으며, 화질의 손실을 감수하면 빠른 시간에 압축 복원이 가능하다.

마지막으로 2.5절에서 제안한 선택적 블록 복원을 이용한 성능 향상 정도를 확인하였다. 불투명도 전이함수의 설정은 그림 5의 볼륨가시화 영상을 통하여 확인할 수 있듯이 복부 데이터의 경우 많은 부분이 투명하게 설정되었으며 머리 데이터는 상대적으로 적은 부분이 투명하게 설정되었다. 선택적 블록 복원 방법을 적용하고 압축 복원 시간을 측정하였고, GPU 효율을 더욱 향상시키기 위해 압축복원 인덱스 방식을 추가하여 다시 압축 복원 시간을 측정하여 표 3에 나타내었다.

표 1. 쓰레드 블록당 쓰레드의 수(TPB)에 따른 압축 복원 시간 (시간단위: ms)

데이터	CPU	TPB (GPU)								
		1	2	4	8	16	32	64	128	256
머 리	4437	1000.0	604.7	392.3	273.1	210.3	157.4	120.1	140.7	139.8
복 부	65865	N/A	N/A	N/A	2676.8	1972.0	1528.8	1472.7	1750.5	1737.8

표 2. 손실 복원과 무손실 복원의 압축 복원 시간 비교
(시간단위: 초)

	무손실 복원	손실 복원 (문턱값 1)	손실 복원 (문턱값 2)
머리	120.1	93.8	82.5
복부	1472.7	1238.4	992.4

표 3. 선택적 블록 복원 방법의 성능 향상 결과
(시간단위: ms)

	무손실 복원	선택적 블록 복원	선택적 블록 복원+인덱스
머리	120.1	81.9	71.2
복부	1472.7	417.5	380.3

본 연구가 제안하는 선택적 블록 복원 기법은 기존 복원 방법에 비해 약 1.5배(=120.1 / 81.9)에서 3.5배(=1472.7 / 417.5) 가량의 두드러진 성능 향상을 보인다. 선택적 블록 복원의 효과는 사용자가 지정하는 불투명도 전이함수에 따라 다르게 나타나는데, 투명한 영역의 비율이 높은 복부 데이터가 더 많은 블록을 생략하여 높은 효과가 나타나는 것을 확인할 수 있다. 추가적으로 간접주소 지정을 통해 GPU의 성능을 향상시키면 10% 정도의 속도 향상을 얻게 된다. 최종 결과로서, 임상적으로 사용하는 불투명도 전이함수를 지정하는 경우, 512x512x300크기의 실용적인 데이터를 압축 복원하는데 380ms의 적은 시간만이 소요되었다. 한편, 선택적 블록 복원은 빈 공간이 많은 경우에 그 효과가 뛰어나며, 사용자가 정의한 불투명도 전이함수에서 빈 공간이 없도록 설정되는 경우, 성능 향상을 얻지 못하게 되는 한계가 존재한다.

5. 결론 및 향후 과제

본 연구에서는 의료영상의 압축과 복원을 위해 EZW 복원 알고리즘을 GPGPU로 구현하였다. 그리고 그 효율을 향상시키기 위해 볼륨가시화의 빈공간 도약 기법을 접목하여 고속의 압축 복원 알고리즘을 제안하였다. 그 결과로 실용적인 크기의 512×512×300 크기의 데이터에 대해, 볼륨가시화와 연동한 압축 복원을 380ms의 즉각적인 시간에 수행해 낼 수 있었다. 비록 투명도 적용은 직접 볼륨가시화에만 적용이 가능하고 불투명도 전이함수에 의존적인 결과이기는 하지만, 측정된 결과는 사용자의 명령에 대해 즉각적으로 압축 복원이 적용될 수 있음을 나타낸다.

본 연구의 한계로서, 압축 효율을 더욱 향상시키는 산술부호화 과정은 생략되어있기 때문에, 이를 추가하면 더욱 실용적인 결과를 얻을 수 있을 것이다. 그리고 Lindstrom 등이 언급한 바와 같이[20] 가변길이 코딩의 결과는 압축된 크기 정보를 함께 저장해야 하므로 비효율적인 경우가 있고 이를 극복하는 효과적인 압축방법의 연구가 필요하다. 추후 연구로서, 현재 의료영상 무손실 압축에 많이 사용되는 JPEG2000[21]을 CUDA로 적용하여 상업용 시스템에 적용할 수 있을 것으로 기대한다.

참 고 문 헌

[1] M. Levoy, “Display of Surfaces from Volume Data,” *IEEE Computer Graphics and Application*, Vol.8, No.3, pp. 29-37, 1988.

[2] L. Bottou, P. Haffner, P. Howard, P. Simard, Y. Bengio, and Y. LeCun, “High quality document image compression using DjVu,” *Journal of Electronic Imaging*, Vol.7, No.3, pp. 410-425, 1998.

[3] Z. Xiong, X. Wu, S. Cheng, and J. Hua, “Lossy-to-Lossless Compression of Medical Volumetric Data using Three-Dimensional Integer Wavelet Transforms,” *IEEE Transactions on Medical Imaging*, Vol.22, No.3, pp. 459-470, 2003.

[4] N. Fout and K.L. Ma, “Transform Coding for Hardware-accelerated Volume Rendering,” *IEEE Transaction on Visualization and Computer Graphics*, Vol.13, No.6, pp. 1600-1607, 2007.

[5] E.B. Lum, K.L. Ma, and J. Clyne, “Texture Hardware Assisted Rendering of Time-Varying Volume Data,” *Proc. of the Conference on Visualization '01*, pp. 263-270, 2001.

[6] M. Kraus and T. Ertl, “Adaptive Texture Maps,” *Proc. of the ACM SIGGRAPH / EUROGRAPHICS Conference on Graphics Hardware*, pp. 7-15, 2002.

[7] M. Pratt, C. Chu, and S. Wong, "Volume Compression of MRI Data using Zerotrees of Wavelet Coefficients," *Proc. SPIE Wavelet Applications in Signal and Image Processing IV*, Vol.2825, pp. 752-763, 1996.

[8] A. Said and W. Pearlman, "A New, Fast and Efficient Image Codec Based on Set Partitioning," *IEEE Trans. Circuits Syst. Video Technol.*, Vol.6, No.3, pp. 243-250, 1996.

[9] D. Taubman, "High Performance Scalable Image Compression with EBCOT," *IEEE Trans. Image Processing*, Vol.9, No.7, pp. 1158- 1170, 2000.

[10] K. Engel, M. Hadwiger, J.M. Kniss, C. Rezk-Salama, and D. Weiskopf, *Real-Time Volume Graphics*, Wellesley, Massachusetts, 2006.

[11] J.M. Shapiro, "Embedded Image Coding using Zerotrees of wavelet Coefficients," *IEEE Transactions on Signal Processing*, Vol.41, No.12, pp. 3445-3462, 1993.

[12] CUDA, Available online (http://www.nvidia.com/object/cuda_home_new.html), 2012

[13] 이만희, 박인규, 원석진, 조성대, "GPU를 이용한 DWT 및 JPEG2000의 고속 연산," 전자공학회논문지 제44권, 제6호, pp. 9-15, 2007.

[14] DirectX, Available online (http://msdn.microsoft.com/en-us/directx), 2012

[15] 계희원, 김준호, "GPGPU 환경에서 최대휘소투영 렌더링의 고속화 방법," 멀티미디어학회논문지, 14권 8호, pp. 981-991, 2011.

[16] M. Levoy, "Efficient Ray Tracing of Volume Data," *ACM Transactions on Graphics*, Vol.9, No.3, pp.245-261, 1990.

[17] W. Li, K. Mueller, and A. Kaufman, "Empty Space Skipping and Occlusion Clipping for Texture-Based Volume Rendering," *Proc. of IEEE Visualization Conference*, pp. 317- 324, 2003.

[18] A. Balevic, "Parallel Variable-Length Encoding on GPGPUs," *Proc. of the 2009 International Conference on Parallel Processing, Springer-Verlag, Berlin, Heidelberg*, pp. 26-35, 2009.

[19] M.A. O'Neil and M. Burtscher, "Floating-point data compression at 75 Gb/s on a GPU," *Proc. of the Fourth Workshop on General Purpose Processing on Graphics Processing Units*, pp. 7, 2011.

[20] P. Lindstrom and J.D. Cohen, "On-the-Fly Decompression and Rendering of Multiresolution Terrain," *Proc. of IEEE Visualization 2009, Atlantic City, NJ, USA*, 2009.

[21] T. Acharya and P.S. Tsai, *JPEG2000 Standard for Image Compression: Concepts, Algorithms and VLSI Architectures*, John Wiley & Sons, New Jersey, 2005.

계 희 원

1999년 2월 서울대학교 전산과학과 학사
2001년 2월 서울대학교 전기컴퓨터공학부 석사
2005년 8월 서울대학교 전기컴퓨터공학부 박사

2006년 1월~2007년 3월 서울대학교 컴퓨터연구소 연구원
2007년 9월~현재 한성대학교 정보시스템공학과 조교수
관심분야 : 볼륨 가시화, 실시간 렌더링, 대용량 영상처리