Daubechies D4 웨이블릿 필터를 이용한 유닛 (Unit) 기반 볼륨 데이터 압축 기법

허영주[1], 박상훈[2]
한국과학기술정보연구원(KISTI)[1]
동국대학교 영상대학원[2]

A Unit-Based Volume Data Compression Scheme Using Daubechies D4 Wavelet Filter

Young Ju Hur[1], Sang Hun Park[2]
Korean Institute of Science and Technology Information[1]
Graduate School of Digital Image & Contents, Dongguk University [2]

요 약

데이터 압축 기술은 대용량의 데이터를 효율적으로 저장할 수 있게 해주는 기술로, 여러 분야에서 생성되는 데이터의 용량이 커지고 네트워크를 통한 데이터 전송에 대한 필요성이 증가함에 따라 그 중요도가 점점 더 커지고 있는 추세다. 특히 다양한 과학 분야에서 시뮬레이션의 결과로 산출되는 볼륨 데이터는 컴퓨팅 기술의 발전에 힘입어 점점 더 용량이 방대해지고 있는 추세이기 때문에 볼륨 데이터 압축에 대한 요구는 계속 커지고 있다.

본 논문에서는 Daubechies 의 D4 기저함수를 이용한 웨이블릿 필터 변환과 zerobit 인코딩 기법을 응용한 유닛 기반의 볼륨 데이터 압축 기법을 제안한다. 유닛 기반 인코딩 기법은 복원 데이터의 손실율이 낮기 때문에 적은 웨이블릿 변환 계수로 화질이 좋은 이미지를 얻을 수 있다. 따라서 정밀한 영상을 요구하는 대용량 데이터의 압축 및 렌더링에 유용하게 사용할 수 있을 것이다.

Keyword : 데이터 압축, wavelet

1. 서 론

각 분야에서 생성되는 데이터의 용량이 커지고, 이런 대형 데이터를 네트워크를 통해 전송해야 하는 필요성이 증가함에 따라 데이터 압축 기술에 대한 중요도도 높아지고 있다. 그러나 볼륨 데이터는 다른 유형의 데이터, 특히 멀티미디어 데이터에 비해 압축 기법에 대한 연구가 그다지 활발하지 않았다. 볼륨 데이터는 컴퓨팅 기술의 발전에 힘입어 최근에는 점점 더 대형화되는 추세에 있기 때문에 이런 데이터를 압축하는 기법, 그리고 이런 압축 기법을 표준화하는 데 대한 연구의 중요성은 점점 더 강조되고 있다.

일반적으로 데이터 압축 기법은 손실 압축(lossy compression)과 무손실 압축(lossless compression)으로 나눌 수 있다. 일반적으로는 데이터가 일부 손실되더라도 가시화 결과로 생성된 이미지에는 큰 영향을 미치지 않으면서 높은 압축 효율을 구현할 수 있기 때문에 손실 압축 기법을 많이 사용한다.

본 논문에서는 Daubechies 의 D4 기저 함수를 이용한 웨이블릿 변환과 zerobit 인코딩 기법을 응용한 유닛 기반의 볼륨 데이터 압축 기법을 제안한다. 본 논문은 다음과 같이 구성된다. 2 절에서

는 유닛 블록 인코딩 기법과 관련된 기존 연구에 대해 설명하고 3 절에서는 유닛 블록 인코딩 기법에 대해 상세하게 설명하겠다. 그리고 4 절에서는 압축 및 복원 결과를 설명한 뒤, 5 절의 결론으로 끝을 맺기로 한다.

2. 관련 연구

일반적인 손실압축 기법에서는 데이터의 값이 분포하는 범위가 작으면 작을수록 압축 효율이 높아지기 때문에, 인코딩에 앞서 데이터를 변환함으로써 데이터의 분포 범위를 줄인다. 데이터 변환 방식은 매우 다양한데, 최근에는 계산이 용이하고 기저함수가 다양하다는 장점으로 인해 웨이블릿 변환 기법이 많이 사용되고 있다. 웨이블릿 변환을 사용하는 압축 기법에서는 어떤 기저(basis)함수를 사용하느냐에 따라 압축률, 복원 화질, 복원 속도 등이 달라진다. 지금까지의 대부분의 압축 기법에서는 계산상의 편의를 이유로 Haar 기저함수를 많이 사용해 왔다. 이 기저함수는 최소한의 계산으로 압축과 복원을 수행할 수 있기 때문에 실시간 복원이 중요시되는 분야에서 많이 활용되고 있다. [6]에서는 Haar 기저함수를 이용한 3 차원 웨이블릿 변환 방식을 소개하고 있는데, Haar 기저함수는 계산은 간편하지만 복원 데이터와 원래 데이터 사이의 오차가 크다는 단점을 가진다.

Daubechies[12]는 Haar 기저함수보다 복원 데이터의 손실율이 적은 새로운 웨이블릿 기저함수군을 제안했다. 이 기저함수군은 2 개에서 20 개의 인수를 사용하는 변환방식을 제공하는데, 그 중 4 개의 계수를 사용하는 D4 방식은 계산의 복잡도가 그다지 크지 않으면서도 Haar 기저함수보다는 복원 화질이 좋은 이미지를 제공한다.

이렇게 변환된 데이터는 데이터의 범위 및 특성에 맞는 방식으로 인코딩할 수 있으며, 인코딩 방식 역시 매우 다양하다.

[9]와 [13]에서는 웨이블릿을 적용해서 변환한 데이터의 계층적 구조를 이용한 인코딩 방식을 소개했으며, [11]에서는 비트스트림에서 비트의 중요도에 따라 인코딩의 우선순위를 달리하는 EBCOT 이라는 방식을 소개했다. EBCOT 은 JPEG2000 에 채택된 인코딩 방식이기도 하다.

[3]에서는 Haar 웨이블릿 기저함수로 변환한 3 차원 볼륨 데이터 압축에 초점을 맞춘 zerobit 인코딩 방식을 소개했다. Zerobit 인코딩 방식은 웨이블릿으로 변환된 데이터의 특성, 즉 웨이블릿으로 변환한 데이터에서는 0 이 많이 발생하며, 이 0 값은 몰려있는 경향이 있다는 특성을 이용해서 데이터의 용량을 줄인 압축 기법이다. 특히 데이터 값의 범위에 따라 1 바이트 크기의 스트림과 2 바이트 크기의 스트림을 따로 저장함으로써 데이터 공간의 낭비를 최소화하고 압축률을 높였다.

본 논문에서는 [14]에서 제안한 Daubechies D4 기저함수를 이용한 3 차원 볼륨 데이터 인코딩 기법을 구현하고 API 형태의 사용자 인터페이스를 설계, 구현한 과정을 설명한다. 이 인코딩 방식은 zerobit 인코딩 방식의 기본 개념을 이용해서 Daubechies D4 방식의 특성에 맞게 변형한 것으로, 16 x 16 x 16 크기의 유닛 블록(unit block)을 압축의 기본 단위로 이용한다. 유닛 기반 인코딩 기법이라는 이름이 붙여진 이 방식은 3 차원 볼륨 데이터에 대한 손실 압축을 지원할 뿐만 아니라 유닛 블록에 대한 무작위 추출 복원을 지원하며, Haar 기저함수를 사용한 압축 방식에 비해 데이터 손실율이 적기 때문에 정밀한 데이터를 요구하는 대용량 데이터 압축에 활용할 수 있을 것이다.

3. 유닛 기반 인코딩

데이터를 압축하려면 데이터 변환 과정과

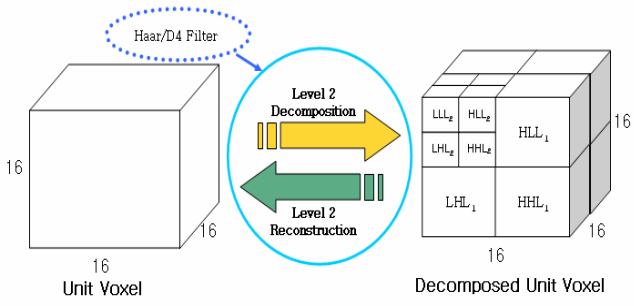

그림 1. 유닛 블록의 웨이블릿 변환

truncation 과정 및 인코딩 과정을 거쳐야 한다. 웨이블릿 변환을 이용해서 데이터를 변환하고 나면 truncation 과정을 거치는데, truncation 과정은 데이터 복원에 상대적으로 영향을 덜 미치는 의미 없는 데이터를 모두 0 으로 바꾸는 과정을 가리킨다. 실제로 저장해야 할 데이터의 개수는 이 truncation 과정에서 현저하게 줄어들며, 실제로 데이터 손실도 이 과정에서 일어난다. 이렇게 truncation 과정을 끝낸 데이터는 인코딩 과정을 거쳐서 새로운 형태로 저장되는데, 바로 이 데이터가 최종 압축 데이터다.

이렇게 압축된 데이터를 본래 상태로 되돌리는 과정은 압축 과정과 반대 순서로 진행된다. 우선 압축 데이터를 디코딩해서 압축 데이터에 저장돼 있는 0 이 아닌 웨이블릿 계수와 그 계수의 위치 정보를 파악한다. 그리고 이렇게 디코딩된 웨이블릿 계수가 웨이블릿 복원 과정을 거치면 본래 데이터로 복원된다.

이제, 데이터 압축 및 복원의 각 과정에 대해 자세히 알아보기로 한다.

3-1. 3 차원 웨이블릿 변환 및 truncation

앞서 설명했듯이, 유닛 기반 인코딩 방식에서는 Daubechies 의 D4 기저함수를 사용해서 데이터를 변환한다. 웨이블릿 변환은 전체 데이터를 16 x 16 x 16 크기의 유닛 블록으로 나눈 뒤, 각각의 유닛 블록에 대해 3 차원 웨이블릿 변환을 2 번씩 적용한다(그림 1).

이렇게 변환된 데이터는 truncation 이라는 과정을 거치게 된다. 이 과정은 임계값을 정한 뒤, 임계치 이상의 값을 가지는 계수만 남겨두고 그보다 작은 계수는 모두 0 으로 변경하는 것이다. 즉, 데이터 복원에 상대적으로 영향을 덜 미치는 값을 모두 0 으로 전환함으로써 압축률을 높이기 위한 과정이라고 할 수 있다.

일반적으로 임계값을 선정하는 과정은 다음과 같다. 우선, 웨이블릿 변환 과정의 결과로 생성된 계수를 크기 순서대로 정렬한다. 이렇게 계수를 정렬한 뒤, 사용자가 정한 계수 비율에 해당하는 위치의 계수를 임계값으로 설정하면 된다. 즉, 7%의 계수만 사용한다고 하면, 크기 순서로 정렬된 웨이블릿 계수에서 상위 7%에 해당되는 계수값을 임계값으로 설정하면 된다.

임계값을 선정하고 임계값보다 크기가 작은 계수를 모두 0 으로 전환하면 인코딩을 위한 선행 작업은 모두 완료된다. 인코딩은 이렇게 준비된 데이터를 특정 형태로 저장하는 과정을 가리키며, 이 과정에서도 데이터 용량은 현저하게 줄어들 수 있다.

3-2. 유닛 기반 인코딩

인코딩 과정은 최종 압축 데이터를 생성하는 과정으로, 인코딩 과정이 끝난 데이터는 데이터 보관이나 전송에 필요한 리소스를 절약해 준다.

인코딩 과정에서는 데이터의 '값' 뿐만 아니라 '위치' 정보도 고려해서 저장해야 데이터를 제대로 복원할 수 있다. 유닛 기반 인코딩 방식에서는 유닛 플랙 테이블(Unit Flag Table), UVBF(Unit Voxel Bit Table), 그리고 BSDT(Byte Stream Designation Table)이라는 자료구조를 이용해서 데이터에 대한 위치 정보를 저장한다. <그림 2>에서는 유닛 기반 인코딩 방식의 데이터 저장 형태를 보여주고 있다.

유닛 플랙 테이블은 전체 볼륨 데이터에 속한 유닛 중 유닛에 속한 웨이블릿 계수가 모두 0 인 유닛의 위치를 나타내는 자료구조다. 이 자료 구조는 전체 계수가 0 인 유닛을 1 비트로 표기할 수 있게 해 주며, 볼륨 데이터 특성에 따라서는 유닛 플랙 테이블의 사용만으로도 데이터의 크기를 상당량 줄일 수 있다.

유닛에 포함된 계수중 0 이 아닌 계수가 하나라도 존재하는 경우에는 UVBT 와 BSDT 를 사용해서 유닛 내에서의 계수의 위치를 표기하고 실제 데이터 값을 1 바이트 크기의 스트림과 2 바이트 크기의 스트림에 나눠서 저장한다. 웨이블릿 계수는 모두 정수 형태로 저장되기 때문에 여기에서도 약간의 데이터 손실이 발생할 수 있다.

이 중, UVBT 는 유닛 내의 계수값이 0 인지 여부를 표기하는 자료구조로, 찾고자 하는 계수의 위치에 해당하는 UVBT 비트가 0 이면 계수값은 0 이

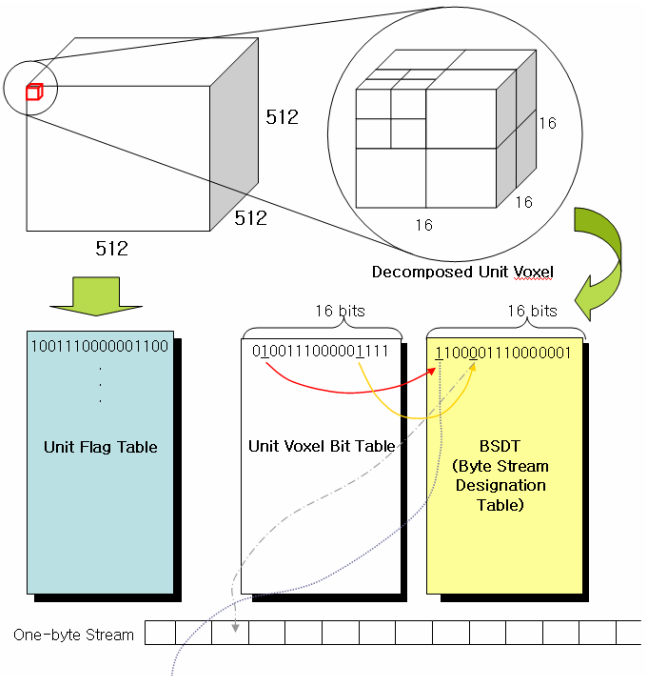

그림 2. 유닛 기반 인코딩 기법

된다. 만약 해당 비트가 1 이라면 이 계수값은 바이트 스트림에서 검색해야 하는데, 이 때 해당 계수값이 저장된 바이트 스트림을 알려주는 자료구조가 바로 BSDT 다. BSDT 의 비트는 유닛 내에서도 값이 0 이 아닌 계수에 대해서만 할당되는 비트로, UVBT 의 비트가 1 인 경우에는 BSDT 의 해당 비트값을 참조해야만 정확한 계수 위치를 알아낼 수 있다.

BSDT 의 해당 비트가 0 이면 계수값이 1 바이트 스트림에 위치한다는 것을 의미하고, 비트가 1 인 경우에는 계수값이 2 바이트 스트림에 위치한다는 것을 의미한다.

유닛 기반 인코딩 방식은 웨이블릿 변환을 거친 데이터에는 0 값이 많이 존재하고, 이 때의 0 값은 주로 한데 몰려있다는 성질을 이용한 압축 방식으로, 압축 대상인 볼륨 데이터의 특성에 따라 압축률이 크게 달라지기도 한다. 또, 1 바이트 크기의 데이터와 2 바이트 크기의 데이터를 분리해서 저장함으로써 크기가 큰 저장공간에 크기가 작은 데이터가 저장됨으로써 발생할 수 있는 데이터 공간의 낭비를 최소화함으로써 압축률을 높인다.

유닛 기반 인코딩 방식에서는 웨이블릿 계수를 모두 정수 형태로 저장하기 때문에, 정수 형태의 볼륨 데이터에 적용해야만 데이터 손실이 낮다.

유닛 기반 볼륨 데이터 인코딩에서는 실제 계수값 이외의 정보도 저장하기 때문에 약간의 오버헤드가 따른다. 이 오버헤드는 계수값이 저장된 위치를 찾아가는데 필요한 자료구조 저장에 소요되는 비용으로 유닛 내에서의 계수값의 위치와 바이트 스트림 내에서의 값의 위치를 저장하는데 사용된다. 이 오버헤드에 드는 비용은 계수 전체를 저장할 때 소요되는 저장 공간에 비해 그다지 크지 않기 때문에 큰 부담 없이 데이터를 효율적으로 압축하는 것이 가능하다.

3-3. API

유닛 기반 인코딩 기법의 API 는 상태 정보와 관련된 요소와 데이터 압축에 관련된 요소로 나눌 수 있다. 상태 정보와 관련된 요소는 데이터를 압축하는데 필요한 여러 가지 정보를 설정하거나 설정된 정보를 참조하는데 사용하는 기능 요소를 포함한다.

데이터 압축에 필요한 기능 요소는 크게 데이터 인코딩과 디코딩으로 나눌 수 있다. 데이터 인코딩 기능은 사용자의 간섭이 없는 일괄 처리 형태의 인터페이스가 가능하다. 따라서 WVEncode 라는 단일함수를 제공함으로써 데이터 변환, truncation 및 encoding 과정을 한꺼번에 처리할 수 있게 했다. 그러나 데이터 디코딩 과정은 이와 다르다. 디코딩 과정에서는 애플리케이션 수준에서 필요로 하는 데이터를 실시간으로 처리해야 한다는 필요성이 대두되기 때문이다. 이에, 디코딩에 필요한 API 에는 다음과 같이 여러가지 모드를 제공함으로써 애플리케이션이 요구하는 데이터를 처리할 수 있게 했다.

- ALL: 이 모드에서는 인코딩된 데이터 전체가 한꺼번에 디코딩된다.
- WV_VOXEL: 디코딩 함수를 이 모드에서 실행시키면 압축된 볼륨 데이터에서 사용자가 지정한 복셀(i, j, k 인덱스로 지정)값이 디코딩된다.
- WV_CELL: 디코딩 함수를 이 모드로 실행시키면, 압축된 볼륨 데이터에서 사용자가 인덱스로 지정한 복셀과 그 주변 복셀값이 디

코딩된다. 이 모드는 압축된 데이터로 직접 볼륨 렌더링을 수행하는데 유용하다.

유닛 기반 인코딩 기법에서는 인코딩이나 디코딩의 기본 단위는 16 x 16 x 16 크기의 유닛이기 때문에, WV_VOXEL 이나 WV_CELL 모드로 디코딩을 수행할 경우는 계산이 중복된다는 문제가 제기된다. 이에, 캐시를 제공하여 계산상의 효율성을 높였다.

현재 구현돼 있는 디코딩 모드는 이 3 가지 이며, 계속 WV_UNIT, WV_SLIDE 등 여러 다양한 디코딩 모드를 추가함으로써 사용성의 편이를 제공할 예정이다.

4. 결과

유닛 기반 인코딩 방식으로 압축한 데이터의 압축률과 화질은 <표 1>에서 비교해볼 수 있다. 이 실험에는 VKH(Visible Korean Human)와 VH(Visible Human) 데이터를 사용했으며, 각각 512 x 512 x 512 크기의 볼륨 데이터와 512 x 512 x 1248 크기의 볼륨 데이터를 사용, 실험을 수행하고 결과를 비교했다.

<표 1>에서 볼 수 있듯이 압축 데이터의 용량은 원본 데이터의 40% 이하로 현저하게 줄어든 반면, 원본 데이터와 복원 데이터의 오차 비율이라고 할 수 있는 PSNR(Peak-Signal-to-Noise Ratio)은 그다지 큰 차이를 보여주지 않는다. 즉, 압축율에 비해 데이터 손실은 크지 않으며, 사용한 웨이블릿 계수의 비율이 줄어들더라도 PSNR 은 크게 줄어들지 않는 것을 알 수 있다. 그러나 계수 비율이 줄어드는 데 따르는 데이터 압축 효과도 크지 않기 때문에 향후 이 부분에 대한 개선이 필요하다. 이는 실제로 저장할 계수의 개수 및 용량은 줄어들지만, 유닛 저장에 따르는 오버헤드는 그대로이기 때문에 발생하는 현상이다. 따라서, 계수의 위치 저장에 사용되는 오버헤드 데이터의 용량을 줄일 수 있어야만 개선이 가능할 것이다.

	512 x 512 x 1248 (630MB)					
Data	Visible Korea Human (VKH)					
Truncation Ratio (%)	100 %	10%	7%	5%	3%	1%
Comp.Data Size (MB)	174	57	47	39	32	21
PSNR (dB)		57.78	55.63	53.82	51.05	43.72

표 1. 압축 수행 후의 데이터 용량 및 손실율 비교

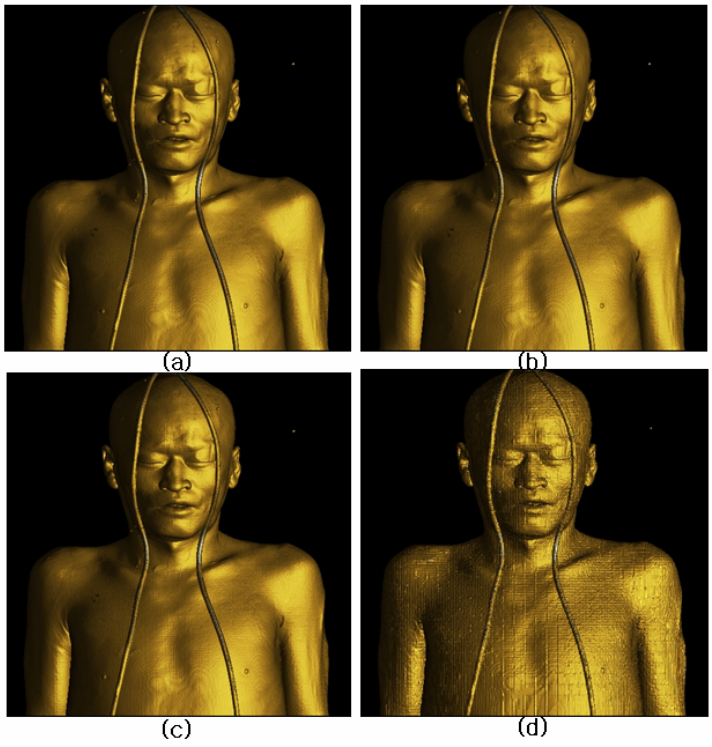

그림 3. 압축 데이터를 복원한 뒤 렌더링한 결과 이미지
(a) 10%, (b) 7%, (c) 5%, (d) 1% coefficients

<그림 3>은 각각 10%, 7%, 5%, 1%의 웨이블릿 계수로 복원한 512 x 512 x 512 해상도의 VKH 데이터를 CPU 를 사용한 레이캐스팅 기법으로 렌더링한 결과다. 결과 이미지에서 볼 수 있듯이 10%나 7%의 계수를 사용했을 경우에는 거의 육안으로 구별할 수 없을 정도로 복원 화질이 좋은 이미지를 얻을 수 있으며, PSNR 도 50dB 이상의 수치가 나오기 때문에 데이터 손실율도 거의 없는 편이다.

512 x 512 x 512 크기의 VKH 데이터 전체를 디코딩하는 데는 PC 에서 20 초 정도의 시간이 걸리며, 유닛 단위로 인코딩하는 것이 가장 속도가 빠르다. 셀 단위나 복셀 단위 디코딩에서는 캐시를 사용, 데이터 복원 속도를 높였다.

5. 결론

본 논문에서는 Daubechies 의 D4 웨이블릿 필터를 이용한 볼륨 데이터 압축 기법인 유닛 기반 인코딩 기법 및 그에 대한 API 함수를 구현했다. 이 인코딩 기법은 D4 웨이블릿 필터로 변환된 데이터를 저장하는 압축 기법이다. 유닛 기반 인코딩 기법은 높은 압축률에 비해 상대적으로 화질의 손상이 크지 않기 때문에 정밀한 영상을 필요로 하는 경우에 유용하게 사용할 수 있다. 또, 인코딩/디코딩 과정에 필요한 API 함수를 제공함으로써 사용성의 편이를 높였으며, 특히 다양한 디코딩 모드를 지원함으로써 애플리케이션의 특성에 맞는 디코딩 방식을 효과적으로 활용할 수 있게 했다.

유닛 기반 인코딩 기법에서는 계수의 위치 정보와 값을 동시에 저장한다. 향후에는 계수의 위치 정보 저장에 드는 공간을 최소화해서 압축률을 높이는 방안을 모색하고, 이 기법을 기반으로 시간에 따라 변하는 볼륨 데이터에 적용할 수 있는 기법을 고안, 적용할 계획이다.

참고 문헌

[1] C. Bajaj, I. Ihm, S. Park, “3D RGB compression for interactive applications”, *ACM Transactions on Graphics*, Vol.20, pp.10-38, 2001.

[2] I. Daubechies, *Ten Lectures on Wavelets*, SIAM, 1992.

[3] I. Ihm, S. Park, “Wavelet-based 3D compression scheme of interactive visualization of very large volume data”, *Computer Graphics Forum*, Vol.18, 1999.

[4] S. Park, “Compression-based visualization of large volume data”, *Proceedings of Korea Supercomputing Workshop*, April 2005.

[5] M. Marcelin, M. Gormish, A. Bilgin, M. Boliek, “An Overview of JPEG-2000”, *Proceedings of Data Compression Conference 2000*, pp.523-544, Mar. 2000.

[6] S. Muraki, “Approximation and rendering of volume data using wavelet transform”, *Proceedings of Visualization ’92*, pp.21-28, October 1992.

[7] S. Muraki, “Volume data and wavelet transforms”, *IEEE Computer Graphics and Applications*, pp.50-56, 1996.

[8] F. Rodler, “Wavelet based 3D compression with fast random access for very large volume data”, *Proceedings of Pacific Graphics’99*, pp.108-117, IEEE Press, 1999.

[9] J. Shapiro, “Embedded image coding using zerotrees of wavelet coefficients”, *IEEE Transactions on Signal Processing*, Vol.41, pp.3445-3463, Dec. 1993.

[10] E. Stollnitz, T. DeRose, D. Salesin, *Wavelets for Computer Graphics: Teory and Applications*, Morgan Kaufmann Publishers, 1996.

[11] D. Taubman, “High Performances Scalable Image Compression with EBCOT”, *IEEE Transactions on Image Processing*, Vol.9, no 7, July 2000.

[12] I. Daubechies, D4 Wavelet Transform, http://www.bearcave.com/misl/misl_tech/wavelets/daubechies/.

[13] C. Valens, EZW encoding, http://perso.wanadoo.fr/polyvalens/clemens/ezw/ezw.html

[14] 허영주, 박상훈, Daubechies 웨이블릿 변환을 이용한 볼륨 데이터 압축, 제 24 회 한국정보처리학회 추계학술발표대회, 제 12 권 2 호, 2005