Daubechies 웨이블릿 변환을 이용한 볼륨 데이터 압축

허영주*, 박상훈**
*한국과학기술정보연구원(KISTI)
**동국대학교 영상대학원 멀티미디어학과
e-mail : popea@kisti.re.kr, mshpark@dongguk.edu

Volume Data Compression Using Daubechies Wavelet Transforms

YoungJu Hur*, Sanghun Park **
*Korean Institute of Science and Technology Information
**Dept. of Multimedia, Graduate School of Digital Image & Contents, Dongguk University

요 약

볼륨 데이터는 시뮬레이션 통해 생성되거나 고성능 측정 장비를 이용해 측정된 값으로 구성되는 고차원 데이터의 한 형태로서 다양한 자연과학과 공학분야에서 폭넓게 활용되고 있다. 최근에는 각 분야에서 생성되는 계산 데이터의 용량이 점점 더 증가하고 있기 때문에 이런 대용량의 볼륨 데이터를 효과적으로 처리하기 위한 기법들에 관한 연구가 수행되고 있으며, 특히 대용량 볼륨 데이터 압축 기법에 대한 필요성이 증가하고 있다. 본 논문에서는 Daubechies 웨이블릿 변환과 zerobit 인코딩 스킴을 응용한 새로운 볼륨 데이터 압축 기법을 제안한다. 이 방법은 기존의 압축 방법에 비해 복원 데이터의 손실이 낮기 때문에 정밀한 영상을 요구하는 대용량 데이터 압축에 유용하게 사용될 수 있다.

1. 서론

데이터 압축 기술은 대용량 데이터를 효율적으로 저장하고 전송할 수 있게 해주는 기술이다. 각 분야에서 생성되는 데이터의 용량이 커지고 네트워크를 통한 데이터 전송에 대한 요구가 증가되면서 데이터 압축 기술의 중요성이 높아지고 있다. 다양한 형태의 멀티미디어 데이터들을 효과적으로 압축하기 위한 여러 가지 다양한 기법들이 지난 몇 년 고안되었으며, 대용량 볼륨 데이터 압축 기법에 관한 연구 결과들도 발표되어 왔다.

손실 압축(lossy compression) 가운데 대표적인 방법으로, 웨이블릿 변환을 이용하는 기법이 2 차원 영상과 3 차원 이상의 볼륨 데이터를 압축에서 널리 이용되고 있다. 웨이블릿 변환에 기반을 둔 압축 기법에서는 어떤 기저(basis)를 선택하여 사용할 것인가의 문제가 압축율, 복원화질, 복원속도에 큰 영향을 미친다. 다양한 웨이블릿 변환의 기저들 가운데 Haar 기저는 매우 적은 계산 비용으로 압축과 복원을 수행할 수 있다는 장점을 갖고 있기 때문에, 실시간 수행이 중요한 응용에서 활용되고 있다. 실제로, [3]에서는 Haar 기저함수를 이용하여 매우 적은 계산으로 변환을 수행하며, 변환 후 생성된 계수들의 계층구조와 0 인 계수들이 특정 위치에 밀집하게 되는 성질을 이용하여 3 차원 볼륨 데이터에 대한 zerobit 인코딩 스킴을 설계하였다. 이 방법은 매우 빠른 속도로 전체 데이터를 복원하지 않고, 임의의 위치의 볼륨 데이터를 복원할 수 있는 구조를 갖고 있기 때문에 복원화질보다 복원속도가 중요한 실시간 그래픽스 응용을 위해 효과적으로 이용될 수 있다. 그러나 다른 기저를 사용한 경우에 비해 복원화질이 좋지 않다는 단점을 내포하고

있다.

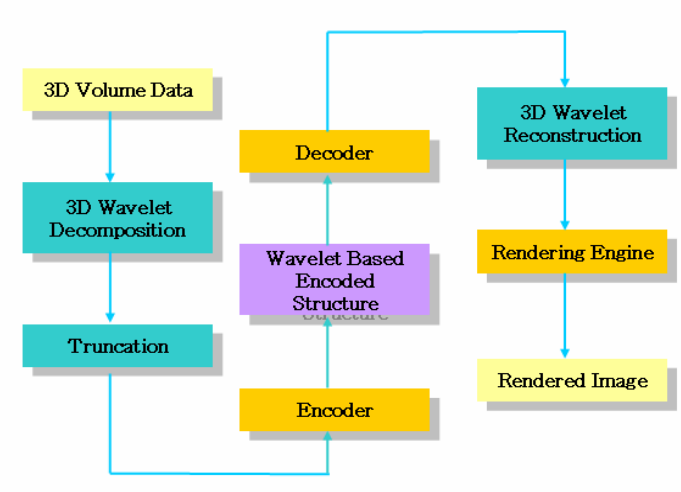


그림 1. 웨이블릿 변환을 이용한 볼륨 데이터 압축/복원과정 [4]

본 논문에서는 복원화질이 뛰어난 Daubechies D4 기저를 사용해서 데이터를 변환하고, 생성된 계수의 효과적인 인코딩을 위해 zerobit 스킴을 D4 변환 방식에 맞게 변형한 새로운 인코딩 기법을 제안한다. 개발된 기법은 손실 압축이며, unit 이라 불리는 16x16x16 크기의 부분 볼륨에 대한 무작위 추출 복원을 지원한다.

2. 시스템 개요

웨이블릿 변환 방식을 이용한 볼륨 데이터 압축 및 복원 과정은 (그림 1)과 같다.

우선 3 차원 볼륨 데이터를 웨이블릿 필터를 이용해서 데이터를 변환한다. 데이터 변환에 사용하는 기저함수는 Haar 기저함수건 D4 기저함수건 간에 원하는 방식을 적용할 수 있다. 이렇게 변환된 데이터는 x, y, z 방향에 대해 low-pass 필터와 high-pass 필터 중 적용한 필터에 따라 LLL, LLH, LHL, LHH, HLL, HLH, HHL, HHH 영역으로 나눌 수 있다. 웨이블릿 필터는 기저 함수에 따라 여러 종류가 존재하는데, 가장 많이 사용되는 방식은 Haar 기저함수를 활용한 방식이다. 이 방식은 빠르고 효율적으로 계산을 수행할 수 있다는 장점[3][10] 때문에 널리 사용되고 있지만 복원 데이터의 정확도가 떨어진다는 단점이 있다. 이에 반해 Daubechies 의 D4 필터[2][10][12]는 Haar 필터보다는 계산이 복잡하지만 복원 데이터가 훨씬 정확한 수치

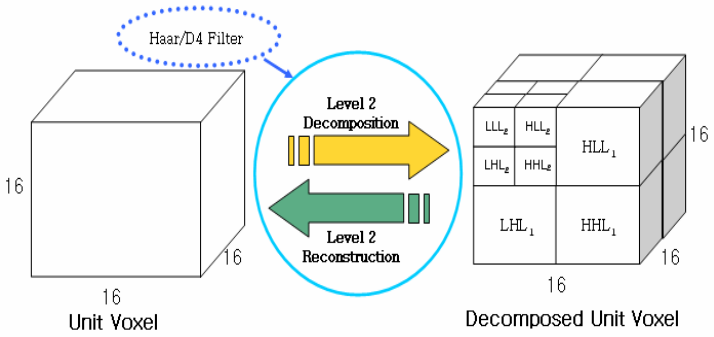


그림 2. Unit Voxel 의 웨이블릿 변환

를 지닌다는 이점이 있다. 본 논문에서는 Daubechies 의 D4 필터를 사용해서 웨이블릿 변환을 수행했으며, D4 필터의 경계 문제를 해결하기 위해 전체 볼륨 데이터를 16 x 16 x 16 크기의 unit 으로 나누고 각각의 unit 에 대해 D4 웨이블릿 변환을 2 단계씩 적용했다 (그림 2).

그런 다음, 변환 데이터에서 임계치 이상의 값을 가지는 데이터 값만 남겨두고 나머지 데이터를 모두 0 으로 전환한다. 이 과정을 Truncation 이라고 하는데, Truncation 과정은 데이터 복원에 상대적으로 영향을 미치지 않는 의미 없는 데이터를 모두 0 으로 바꾸는 과정을 가리킨다. 물론 이 때의 임계값은 압축률에 직접적인 영향을 미친다.

이렇게 Truncation 과정까지 거친 데이터는 인코딩 과정을 거쳐서 새로운 데이터 형식으로 변환된다. 이렇게 인코더를 거친 데이터가 바로 최종 압축 데이터이며, 일반 사용자는 이 압축 데이터를 저장하거나 네트워크 전송에 사용함으로써 자원을 절약할 수 있다. 본 인코딩 방식에서는 데이터를 1 바이트 또는 2 바이트의 정수 데이터로 저장함으로써 저장 공간을 최소화했다. 물론 이로 인해 복원 데이터에 손상이 생기기는 하지만 전체 데이터의 정확도에 큰 영향을 미칠 정도의 손실은 발생하지 않는다. 인코딩 방식에 대해서는 뒤에서 자세히 다룰 것이다.

이렇게 압축된 데이터를 본래 상태로 복원하는 과정은 데이터 압축 과정과 동일하다. 우선, 디코딩을 실행, 압축 데이터를 일반 데이터 형태로 변환한 다음, 웨이블릿 복원 필터를 이용해서 데이터를 복원한다. 복원은 16 x 16 x 16 짜리 unit 에 웨이블릿 필터를 z, y, x 방향에 대해 2 단계로 적용해서 수행하며, 이렇게 복원된 볼륨 데이터를 렌더링하면 최종 이미지를 얻을 수 있다.

이 복원 과정은 16 x 16 x 16 크기의 unit voxel 단위로 수행할 수 있기 때문에, 현재 렌더링하는 부분에 해당하는 unit 을 복원하면서 렌더링을 수행하는 것도 가능하다. 즉, 압축 데이터로부터 필요한 부분만 unit 단위로 복원하는 것이 가능하다.

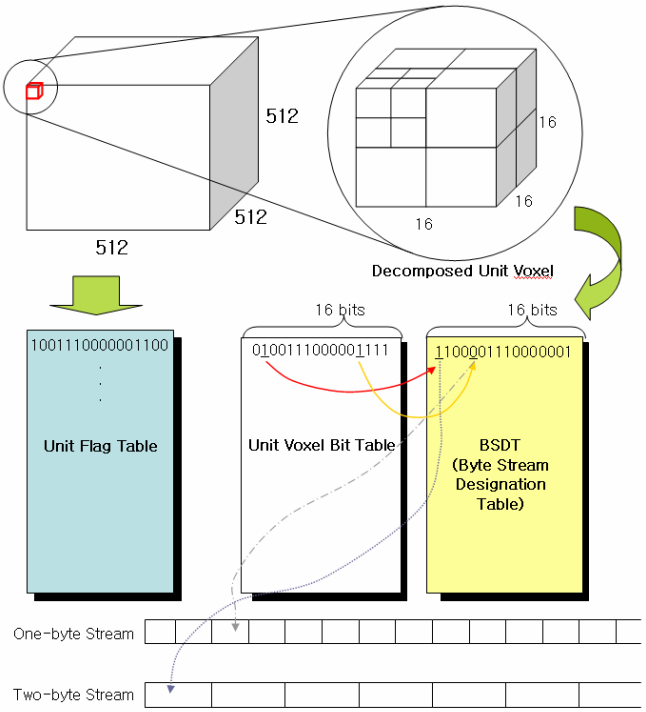
3. Unit 기반 인코딩

이제 unit 을 기본 단위로 하는 인코딩 방식에 대해 알아보자. 데이터를 본래 형태로 복원하기 위해서는 2 가지 정보가 저장돼야 하는데, 이 2 가지 정보는 임계치보다 절대값이 큰 데이터의 '위치'와 '값'이다.

본 논문에서 이 2 가지 정보를 저장하는 데 사용한 방식은 (그림 3)과 같다.

앞서 설명했듯이 웨이블릿 변환을 거친 데이터에서는 0 값이 빈번하게 나타나며, 인접해있는 데이터에서는 값의 변화가 크지 않기 때문에 0 값이 몰려있는 현상도 볼 수 있다. Unit 기반 인코딩 기법은 웨이블릿 변환의 이런 특성을 이용했으며, [3]의 zerobit 인코딩 방식을 D4 변환의 특성에 맞게 변경했다.

우선, unit 에 포함된 웨이블릿 계수가 모두 0 인 unit 의 위치를 표시하기 위해 Unit Flag Table 이라는 자료



구조를 만들었다. 이 자료구조는 전체 볼륨 데이터에서의 unit 의 위치를 일일이 나열한 것으로, unit 내 계수값이 모두 0 인 unit 을 0 으로, 0 이 아닌 계수가 하나라도 존재하는 unit 을 1 로 표기한다. 따라서 포함한 계수값이 모두 0 인 unit 에 존재하는 16x16x16 개의 계수를 1 비트로 나타낼 수 있다.

0 이 아닌 값이 하나라도 존재하는 unit 은 UVBT(Unit Voxel Bit Table)와 BSDT(Byte Stream Designation Table)라는 자료구조를 생성해서 데이터의 위치를 지정하며, 실제 계수는 값의 크기에 따라 1 바이트 스트림과 2 바이트 스트림에 나뉘어 저장된다.

앞의 [2. 시스템 개요] 부분에서 언급했듯이 웨이블릿 계수는 모두 정수 형태로 저장되는데, [-128, 128) 범위 내에 존재하는 값은 1 바이트 스트림에, 이외의 값은 2 바이트 스트림에 저장된다.

UVBT(Unit Voxel Bit Table)와 BSDT(Byte Stream Designation Table)는 바이트 스트림에 저장된 데이터의 위치를 나타내는 데 사용되는 자료구조로 unit 마다 각각 하나씩 존재한다. UVBT 는 해당 unit 에 포함된 웨이블릿 계수가 0 인지 혹은 값이 있는지를 나타낸다. 따라서 UVBT 에서 찾고자 하는 계수의 위치에 해당하는 비트가 0 이면 사용자가 찾는 계수값은 0 이다. 그러나 UVBT 의 해당 비트가 1 이면 1 바이트 스트림이나 2 바이트 스트림에서 값을 찾아야 한다. 이 때 값이 1 바이트 스트림에 있는지, 혹은 2 바이트 스트림에 있는지를 알 수 있게 해주는 것이 바로 BSDT 다. BSDT 는 unit 내에서 값이 0 이 아닌 계수에 대해서만 할당되는 값으로, BSDT 의 해당 비트가 0 이면 1 바이트 스트림에, 1 이면 2 바이트 스트림에 계수가 위치한다는 것을 의미한다.

(그림 3)을 예로 들어보자. (그림 3)의 오른편 아래에 위치한 UVBT 는 16x16x16 개의 값이 각각 0 인지 아닌지를 나타내는 비트 스트림이다. UVBT 에서 볼 수 있듯이, 이 unit 의 첫번째 계수값은 0 이다. UVBT 의 두번째 비트는 1 이므로 이 unit 의 두번째 계수는 값을 가지고 있다. 이 값이 1 바이트 스트림과 2 바이트 스트림 중 어디에 저장돼 있는지에 관한 정보를 담고 있는 자료구조가 바로 BSDT 다. BSDT 는 0 이 아닌 계수에 대해서만 바이트 스트림 정보를 저장하고 있다. UVBT 의 두번째 비트는 0 이 아닌 비트로는 처음 나오는 비트이기 때문에 두번째 계수값이 위치하는 스트림을 찾으려면 BSDT 에서 처음 나오는 비트를 참조해야 한다. BSDT 에서 처음 나오는 비트는 1 이다. 따라서, 두번째 계수는 2-바이트 스트림에 위치하는 것을 알 수 있다. 2-바이트 스트림에서의 값의 위치는 BSDT 에서 해당 비트보다 앞에 나오는 1 의 개수를 세면 알 수 있다. 이 예제에서는 해당 비트보다 선행하는 1 은 없으므로 2-바이트 스트림에서 처음 나오는 값이 바로 두번째 계수 값이라는 것을 알 수 있다.

이 인코딩 방식은 2 가지 면에서 데이터 저장 공간을 줄인다. 우선, 값이 0 인 계수가 몰려있다는 성질을 이용, 전체 계수값이 0 인 unit 을 단일 비트로 표현함으로써 데이터 저장공간을 줄인다. 또, 0 이 아닌 계수가 포함된 unit 에서도 값이 0 인 계수는 단일 비트로 표현하고 0 이 아닌 계수만 1 바이트, 혹은 2 바이트의 크기로 저장함으로써 0 값이 많은 unit 의 저장공간을 최소화한다. 물론, 여기에는 UVBT 와 BSDT 저장에 대한 오버헤드(16x 16x2 +⌈전체 데이터 개수/16⌉ 바이트)가 따른다. 그러나 이 오버헤드는 전체 데이터를 저장하는데 필요한 공간과 비교해 보면 상당히 적기 때문에 이정도 오버헤드는 감수할 수 있다.

4. 결과 및 결론

이 인코딩 기법을 사용해서 압축한 데이터의 용량과 원본 데이터와의 차이는 (표 1)과 같다.

실험에는 VKH(Visible Korea Human)과 VH(Visible Human) 데이터를 사용했으며, 전체 슬라이스 중 512 개와 1248 개의 슬라이스만 추출, 512x512x512 짜리 데이터와 512x512x1248 짜리 데이터 각각에 대해 실험을 수행하고 결과를 분석했다.

(표 1)에서 볼 수 있듯이 압축 데이터의 용량은 원본 데이터의 40% 이하로 현저하게 줄어들었지만, 그에 반해 원본 데이터와 복원 데이터의 차이를 보여주는 PSNR(Peak- Signal-to-Noise Ratio)은 크게 줄어들지 않은 것을 볼 수 있다. 즉, 데이터 압축률에 비해 수치상으로 본 화질의 변화는 그다지 크지 않다는 것을 알 수 있다.

(그림 4)는 일정 비율의 웨이블릿 계수를 0 으로 만들고 인코딩 과정을 거쳐서 압축한 512^3 VKH 데이터를 다시 복원해서 렌더링한 결과를 보여준다. 그림에서 (a)는 10%, (b)는 7%, (c)는 5%, (d)는 1%의 계수만 남겨두고 모두 0 으로 만든 다음 인코딩 과정을 수행해서 압축한 뒤, 이 데이터를 다시 복원해서 렌더링한 결과인데, 10%나 7%의 계수만 복원해도 이미지 화질이 크게 손상되지 않는 현상을 확인할 수 있다.

	512^3 (260MB)						512 x 512 x 1248 (630MB)					
Data	Visible Korea Human (VKH)											
Truncation Ratio (%)	100%	10%	7%	5%	3%	1%	100%	10%	7%	5%	3%	1%
Comp.Data Size (MB)	69	24	19	16	13	8.3	174	57	47	39	32	21
PSNR (dB)		56.62	53.88	51.65	48.80	41.58		57.78	55.63	53.82	51.050	43.72
Data	Visible Human (VH)											
Truncation Ratio (%)	100%	10%	7%	5%	3%	1%	100%	10%	7%	5%	3%	1%
Comp.Data Size (MB)	95	27	21	18	15	8.2	229	63	51	44	35	22
PSNR (dB)		59.87	56.71	54.40	50.33	41.71		60.15	57.67	55.24	51.76	42,90

표 1. 압축 결과: 압축 데이터 용량 및 품질

이 512^3 크기의 데이터를 복원하는 데는 10% 계수 데이터 복원에 20 초가 걸렸고 1% 데이터 복원에는 14 초가 걸렸다. 물론 이 시간은 unit 단위로 데이터 복원 과정을 수행할 때 걸리는 시간이다.

이상의 결과에서 볼 수 있듯이, D4 웨이블릿 변환 데이터를 전제한 unit 기반 인코딩 기법은 높은 압축률에 비해 화질의 손상이 크지 않다는 장점을 가질 뿐만 아니라 데이터 복원 시간도 상대적으로 빠르기 때문에 대용량 데이터 압축에 유용하게 활용할 수 있다. 또, unit 단위로 복원작업을 진행할 수 있기 때문에 렌더링에 직접 필요한 부분을 선택적으로 복원해서 활용할 수도 있다. 향후에는 이 기법을 확장함으로써 시간에 따라 변하는 볼륨 데이터에 적용하는 방안을 고안, 구현할 계획이다.

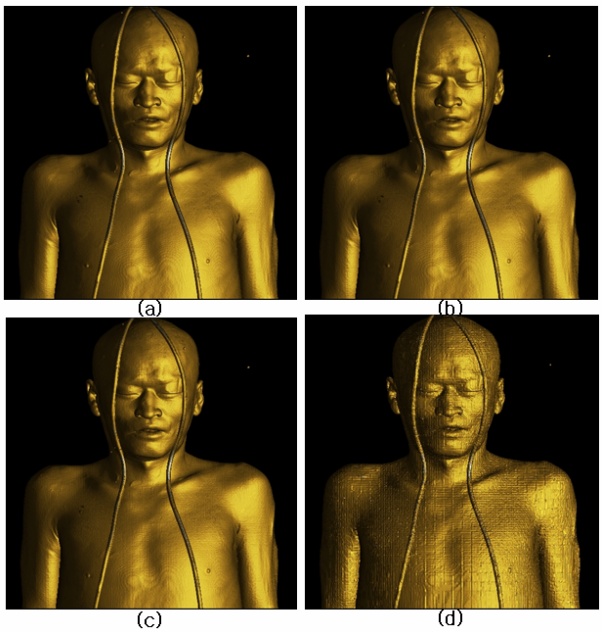


그림 4. 복원한 VKH 데이터로 렌더링한 이미지:
(a) 10%, (b) 7%, (c) 5%, (d) 1%.

참고문헌

[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.