韓國 컴퓨터情報學會 論文誌
第12卷 第3號, 2007. 7.

2007-12-3-1-10

시변 볼륨 데이터의 압축과 가시화 기법

손 봉 수*

Compression and Visualization Techniques for Time-Varying Volume Data

Bong-Soo Sohn *

요 약

본 논문에서는 시간에 따라서 변하는 삼차원 볼륨의 집합인 볼륨 비디오 데이터의 압축 기법을 제안하고 각각의 프레임을 실시간으로 압축 해제하며 가시화하는 방법을 설명하였다. 시변 볼륨 데이터는 한 프레임의 크기조차도 매우 크기 때문에 그 크기를 줄이는 압축 기법을 수행할 필요가 있다. 그리고 실행 과정에서 일어나는 압축 해제는 시변 볼륨 데이터를 실시간으로 가시화 하는데 있어 병목 현상의 원인이 될 수 있다. 우리는 실행 중의 압축 해제 속도와 데이터의 압축률을 높이기 위하여 삼차원 볼륨 데이터를 작은 블록으로 분해하고 충분히 많이 변하는 블록들을 갱신하는 방법을 제안하였다. 이 방법을 구현한 결과 본 논문에서 제안된 압축 및 압축 해제 기법이 압축 해제 속도와 압축에서 복원된 데이터의 정확성, 그리고 압축률의 정도를 조절하여 대용량 시변 볼륨데이터의 대화형 가시화를 가능하게 함을 알 수 있다.

Abstract

This paper describes a compression scheme for volumetric video data(3D space X 1D time) where each frame of the volume is decompressed and rendered in real-time. Since even one frame size of volume is very large, run-time decompression can be a bottleneck for real-time playback of time-varying volume data. To increase the run-time decompression speed and compression ratio, we decompose the volume into small blocks and only update significantly changing blocks. The results show that our compression scheme compromises decompression speed and image quality well enough for interactive time-varying visualization.

▸ Keyword : 가시화(visualization), 압축(compression), 시변 볼륨 데이터(Time-Varying Volume)

• 제1저자 : 손봉수
• 접수일 : 2007.4.16, 심사일 : 2007.4.18, 심사완료일 : 2007. 4.28.
* 경북대학교 컴퓨터공학과

Ⅰ. 서 론

컴퓨터의 계산 성능과 스캐닝의 정밀성이 향상됨에 따라, 과학적 시뮬레이션과 CT/MRI와 같은 측정 장치들은 점점 더 밀도가 높은 대량의 시변 삼차원 볼륨 데이터(time-varying volume data)를 생성한다. 예를 들면, 본 논문에서 실험한 전 세계 바다 온도 분포의 변화를 나타내는 시변 볼륨 데이터의 크기는 약 240 메가바이트 (2160 X 960 X 30 float type) X 115 프레임이며 총 270 기가바이트이다.

최신 그래픽 하드웨어는 매우 빠른 볼륨 렌더링[1][2]을 가능하게 하는 반면에, 메모리와 디스크 간 또는 메모리와 그래픽 하드웨어 간 대용량의 데이터 전송은 메모리 시스템의 한정된 대역폭 때문에 병목 현상의 주된 요인이 된다. 그러므로 효율적인 데이터 관리 기법이 대용량 시간 변화 볼륨데이터의 가시화에 매우 중요한 요인이 된다. 압축 기법은 데이터의 크기를 줄이기 위하여 자주 사용되는데, 데이터 압축 기법은 일반적으로 공간적인 일관성과 시간적인 일관성을 이용하게 된다. 그러나 볼륨이미지 한 프레임조차도 그 크기가 매우 크기 때문에, 실행 중의 압축 해제는 실시간으로 가시화하는데 있어 병목 현상을 일으킬 수 있으므로 주의 깊게 고려하여야 한다.

이러한 이유로 우리는 시변 볼륨 데이터의 실시간 복원을 위한 효율적인 압축 기법을 개발하였다. 이 기법에 대한 입력은 삼차원 볼륨 비디오, 즉 이산화된 시변 볼륨 데이터 V 이며, $V=\{V_1, V_2, ..., V_T\}$로 나타내어진다. 여기서 V_t는 $V_t=\{f^t_{i,j,k} | i,j,k$는 x,y,z 축의 좌표 값$\}$로 정의되고 t번째 시간점에서의 볼륨 데이터를 의미한다. 압축 기법을 개발하는데 있어 우리는 다음과 같은 목적을 가지고 있다.

- 이미지의 왜곡을 최소한으로 하면서 시간 변화 볼륨 데이터의 크기를 최대한 줄인다.
- 압축 해제의 속도를 최대화한다.
- 압축 해제 방식이 그래픽 하드웨어로 가속화된 볼륨 렌더링 기법에 적절하도록 만든다.

우리는 데이터 내에 존재하는 공간적 일관성과 시간적 일관성을 효율적으로 이용하기 위하여 MPEG 압축 기법[3]의 아이디어를 빌린다. 그러나 이차원 비디오의 압축에 최적화 되어 있는 MPEG 압축 기법을 삼차원 볼륨 비디오의 압축에 직접적으로 확장하는 것은 우리의 압축 목표에 적절하지 못하다. 이것은 시간 변화 볼륨 데이터의 각 프레임 내 모든 블록들을 압축 해제하는 것이 대화형 가시화를 만족시키는 데 있어 충분히 빠르지 못하기 때문이다.

웨이블릿 변환(wavelet transformation)은 이차원 또는 삼차원 이미지를 압축하는 데 있어 널리 쓰이는 방식이다[4]. 웨이블릿 변환 후 중요하지 않은 계수들을 잘라 냄으로써 이미지 왜곡을 최소화하면서 높은 압축률을 이룰 수 있다. 그러나 각 볼륨 이미지 프레임에 대하여 완전한 웨이블릿 역변환을 수행하는 것은 대량의 볼륨 데이터를 대화형으로 가시화하는데 있어 너무 많은 시간을 소비하게 한다. 연속적인 볼륨 이미지 프레임 간에는 매우 작은 변화만 있는 경우가 많기 때문에, 전체 볼륨 이미지의 완전한 웨이블릿 역변환을 수행하는 대신에 전체 볼륨 이미지를 작은 블록의 집합으로 만들어 많이 변하는 블록들에만 웨이블릿 변환 및 역변환을 수행함으로써 압축률과 압축 해제 속도를 높일 수 있다.

볼륨 이미지의 각 프레임은 자체코드화 프레임(intra-coded frame)과 예언 프레임(predictive frame) 중 하나로 분류한다. 자체코드화 프레임은 독립적으로 압축 해제될 수 있는 반면, 예언 프레임은 그 앞의 프레임과의 차이를 계산하여 압축된다. 서로 다른 블록들이 서로 다른 시간적 변화를 가지고 있다는 것을 가정하면, 압축률과 압축 해제 속도를 높이기 위하여 우리는 블록들을 시간적 변화의 크기순으로 정렬하여 변화가 적어서 중요도가 떨어지는 블록들을 제거할 수 있다.

효율적인 압축에 더하여 볼륨 이미지 프레임을 빨리 그리는 것은 삼차원 비디오의 실시간 재생에 매우 중요하다. 삼차원 텍스쳐 매핑(texture mapping)에 기반한 그래픽 하드웨어를 이용하여 우리는 대량의 볼륨 데이터의 실시간 볼륨 렌더링을 구현하였다.

그림 1은 우리가 고려하는 실시간 삼차원 볼륨 비디오의 가시화 시스템을 위한 전체적인 구조를 보여준다. 디스크 드라이브의 저장 공간을 절약하고 메모리 시스템의 입출력 대역폭의 제한을 극복하기 위하여 압축된 볼륨 데이터의 각 프레임 스트림이 디스크로부터 읽혀진다. 각각의 압축된 볼륨 프레임이 읽혀지면 소프트웨어로 압축이 해제되고, 복원된 볼륨 이미지 배열은 그래픽 하드웨어의 텍스쳐 메모리로 보내져 가시화된다. 이 구조는 사용자로 하여금 볼륨 데이터를 시간과 공간을 변화시키며 대화형으로 탐구할 수 있도록 한다. 이것은 본 논문에서 제안된 시스템의 궁극적인 목

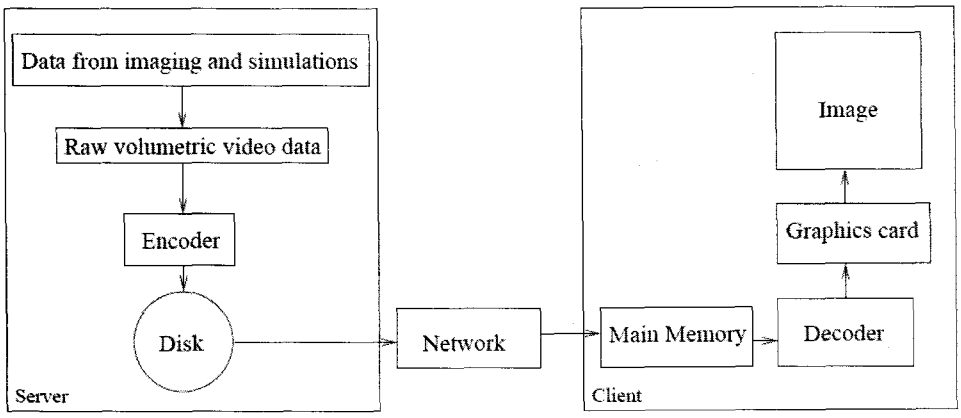

그림 1. 삼차원 볼륨 비디오 시스템의 구조 및 렌더링 과정
Fig 1. System architecture for 3D volumetric video

표이기도 하다.

본 논문의 나머지는 다음과 같이 조직되어 있다. II 절에서는 관련 연구를 설명하였다. III절에서는 삼차원 볼륨 비디오 가시화를 위한 시스템 구조를 설명하였다. IV절에서는 실시간 압축 해제를 지원하는 삼차원 볼륨 비디오의 압축 방식을 제안하였다. V절에서는 실험 결과를 설명하였다. 마지막으로 VI절에서는 결론을 도출하였다.

II. 관련 연구

시변 볼륨 데이터를 가시화 하는 것은 압도적일 정도로 큰 데이터의 크기 때문에 도전적인 일이었다. 그러한 대량의 데이터 크기를 다루기 위하여 (i) 계층적인 자료 구조, (ii) 고성능 시스템, 그리고 (iii) 압축 기법 등이 적용되어 왔다.

Time-Space Partitioning(TSP) Tree[5][6], 즉 시공간 분할 트리는 삼차원 텍스쳐 매핑 하드웨어를 사용할 수 있으며 시공간에 대한 계층적인 구조를 이용하여 시간 변화 볼륨 데이터를 빠르게 볼륨 렌더링 할 수 있도록 한다. 삼차원 공간은 팔진 트리(Octree)에 의하여 분할되고, 각 팔진 트리 노드는 시간을 재귀적으로 이분하는 이진 시간 트리를 갖는다. 그래서 각각의 이진 트리 노드는 공간과 시간의 정보를 계층적으로 가진다. 각 노드는 공간적인 분산과 시간적인 분산을 나타내는 측정치를 보유하게 되며 이 값들은 렌더링 결과에 영향을 적게 미치는 부분을 건너뛸 수 있도록 해준다. 또한 보는 시점이 고정되어 있을 때 추후 재사용을 위하여 그려진 결과 이미지를 캐쉬하여 저장할 수 있다. 이 방식은 볼륨 렌더링 속도를 매우 향상시키며 데이터의 시공간 일관성을 이용하여 입출력 오버헤드를 줄일 수 있게 해준다.

Ma와 Camp[7]의 논문은 시간 변화 데이터를 가시화하기 위하여 원거리 통신망 환경에서 원거리 가시화 시스템을 제안하였다.

과학적 볼륨 데이터는 보통 매우 크고 반복성이 많기 때문에 연구자들은 메모리와 입출력 대역폭의 효율적인 사용을 위하여 압축된 형태로 사용하기를 선호한다. 많은 개수의 이미지 압축 기법이 개발되었는데, 대부분의 방식들은 재생된 이미지의 왜곡을 최소화하고 압축률을 높이는 것을 목표로 하고 있다. JPEG[8]과 MPEG[4]은 각각 정지 영상과 이차원 비디오 데이터를 압축하기 위하여 개발되었다. 이 방식들은 이미지 왜곡과 압축률의 서로 상반된 관계를 통제하는 기능을 가지고 있다. Sengupta와 그 외 저자들[9]은 연속적인 이미지 간의 차이에 웨이블릿 변환을 적용하는 빠른 비디오 코딩을 제안하였다.

최근에는 Ihm과 그 외 저자들[4]이 대화형 실행의 목적으로 높은 압축률을 유지하면서 데이터 내 임의의 부분을 압축으로부터 재생하는 정지 영상 압축 기법을 개발하였다. Ma와 그 외 저자들[10]은 시변 볼륨 데이터의 효율적인 볼륨 렌더링을 위하여 압축을 사용하였다. Guthe와 그 외 저자들[11]은 웨이블릿 변환과 MPEG 알고리즘을 시간 변화 볼륨 데이터에 적용하였다. Lum과 그 외 저자들[12]은 빠른 렌더링과 압축 해제를 위하여 텍스쳐 하드웨어를 이용하였다. 데이터가 서로 다른 메모리 시스템 간에 압축된 형식으로 전송되므로 입출력 시간이 매우 절약된다. 그러나

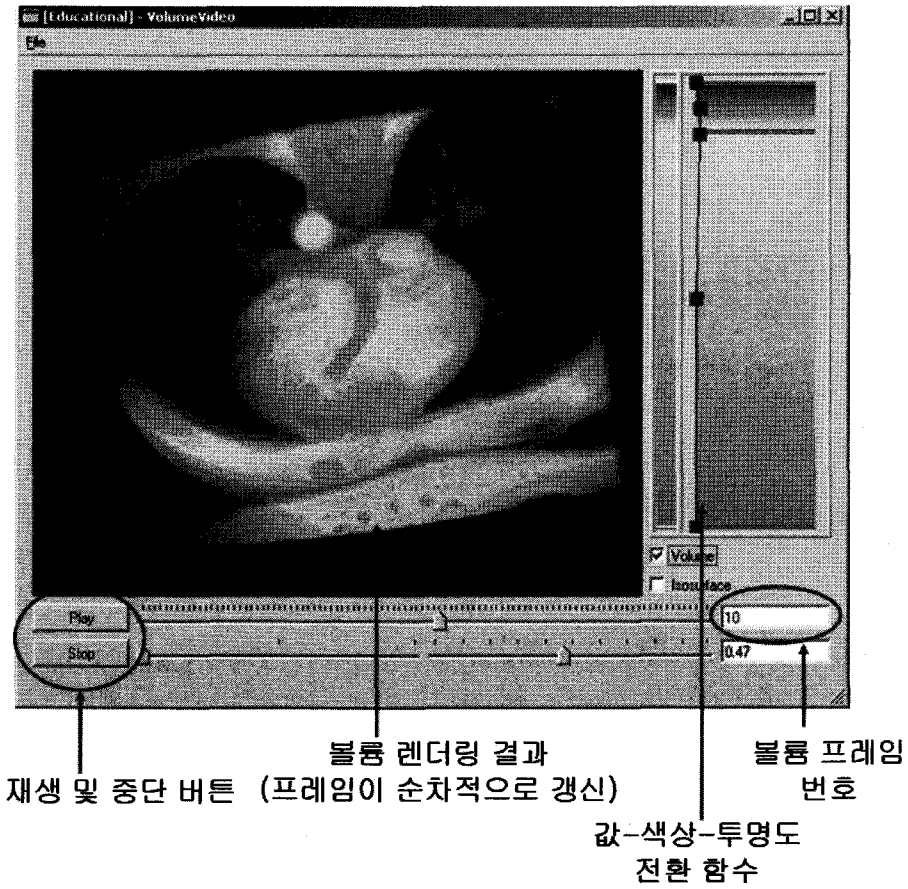

그림 2. 시간의 변화에 따라 심장이 뛰는 모습을 스캔하여 얻어진 시변 볼륨 데이터를 대화형 볼륨 비디오 플레이어로 실행한 예제.
Fig 2. Example of interactive volumetric video playback for time-varying data.

단순한 압축 방식을 써야 하는 제약성 때문에 압축률이 높지 않다.

III. 삼차원 볼륨 비디오

이차원 비디오와 같이 삼차원 볼륨 비디오는 삼차원 이미지들을 매 프레임 순서대로 가시화 한다. 이차원 비디오에서는 사용자가 연속적으로 변하는 이차원 영상들을 수동적으로 볼 수 밖에 없는데, 삼차원 비디오 시스템은 사용자로 하여금 삼차원 볼륨 이미지를 시간적으로 그리고 공간적으로 탐구할 수 있게 해준다. 대부분의 과학적 시뮬레이션이 시간이 변하는 데이터들을 생성한다는 점을 고려하면 삼차원 볼륨 비디오 시스템은 과학적 데이터의 분석에 특히 유용하다. 그림 2는 우리가 개발한 삼차원 볼륨 비디오 플레이어의 작동 화면을 캡쳐한 것이다. 이 예제에서는, 시간이 변함에 따라 심장이 뛰는 모양을 가지는 시변 볼륨 데이터를 입력으로 받아 가시화 한 것이다. 수동적으로 보는 이차원 비디오와 달리 보는 시점과 시간 프레임 그리고 데이터 값-색상-투명도 전환 함수를 대화형으로 변화시킬 수 있는 장점이 있다.

시간 변화 삼차원 볼륨 데이터를 가시화 하는 단순한 방법은 주어진 가시화 변수를 사용하여 디스크로부터 순서대로 각 프레임을 읽고 렌더링 하는 것을 반복하는 것이다. 대부분의 시간 변화 볼륨 데이터들은 그 크기가 매우 크고 시간적, 공간적 일관성을 가지고 있기 때문에 저장 오버헤드를 감소시키기 위하여 압축 기법을

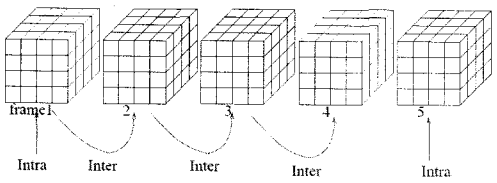

그림 3. 자체코드화 프레임(intra-coded frame)과 예언 프레임(predictive frame)의 분류. 4개의 프레임이 하나의 시간 그룹을 구성하는 예를 보여준다.
Fig 3. Classification of intra-coded frame and predictive frame

적용하는 것을 자연스럽게 생각할 수 있다. 게다가, 하나의 프레임 사이즈가 디스크로부터 실시간으로 읽기에 너무 클 수 있기 때문에 입출력 대역폭 한계를 감소시키기 위하여 효율적인 압축 기법 적용을 요구한다. 그러나 정적인 또는 시간이 변하는 데이터가 표준적인 압축 기법에 의해서 코딩된 데이터는 실행 중 압축 해제할 때 실시간 가시화를 하는 데 있어 병목 현상을 일으키는 주된 요인이 될 수 있다. 이것은 대부분의 표준 압축 기법들이 압축 해제시 이미지 데이터를 구성하는 모든 블록들을 디코딩 하는 것을 요구하기 때문이다. 실행 중의 압축 해제 시간을 낮추기 위하여 우리는 블록들을 중요성에 따라 정렬하고 많이 변하는 블록들만 인코딩함에 의해서 디코드 할 블록 수를 제한하였다.

IV. 볼륨 비디오의 압축

(1) 압축 기법

압축될 입력 데이터는 시변 볼륨 데이터 V이다, $V=\{V_1, V_2, ..., V_N\}$. 이 시간 변화 데이터의 각 프레임은 그림 3에서 보듯이 자체코드화 프레임(intra-coded frame)과 예언 프레임(predictive frame)으로 분류된다. 전체 데이터 V는 다시 다음과 같이 나타내어진다.

$V=\{\{I_1, P_{11}, P_{12}, ..., P_{1k_1}\}, ...$

$, \{I_n, P_{n1}, P_{n2}, ..., P_{nk_n}\}\}$. 이때 I_i는 i번째 시간 그룹에서의 자체코드화 프레임이며 P_{ij}는 i번째 시간 그룹의 예언 프레임이다.

대부분의 경우 연속적인 프레임 간에 작은 변화만 있다고 가정할 때, 전체 프레임 대신 프레임 간 변화에 대해서만 웨이블릿 변환을 수행하는 것은 높은 압축률과 빠른 압축해제 속도를 이룰 수 있게 해준다. 그러므로 예언 프레임의 압축은 같은 시간 그룹 내의 이전 프레임에 종속적인 반면, 자체코드화 프레임의 압축은 다른 프레임의 압축과 독립적으로 이루어진다. 전반적인 압축 알고리즘은 다음과 같다. 압축은 각 볼륨 프레임마다 수행이 됨을 주목하여야 한다. 모든 삼차원 볼륨 프레임은 4x4x4 블록으로 분해되며 웨이블릿 변환은 각각의 블록에 적용된다.

1. 볼륨 차이 계산 : $\Delta V_k = V_k - V'_{k-1}$, V_k는 k번째 프레임 원본 이미지이며, V'_{k-1}는 압축된 V_{k-1}으로부터 복원된 볼륨 이미지이다. 만약 V_k가 자체코드화 프레임이라면 $V'_{k-1}=0$으로 가정한다.

2. 웨이블릿 변환 : $\Delta WV_k = (\Delta V_k)$를 정규화된 Harr 웨이블릿 기저를 이용하여 변환한 것. ΔV_k를 나타내는 웨이블릿 계수 $c_1, ..., c_m$ 를 계산한다.

3. 웨이블릿 계수의 절단과 정렬 : 정규화된 웨이블릿 계수들을 내림차순으로 정렬하여 정렬된 계수들 $c_{\pi(1)}, c_{\pi(2)}, ..., c_{\pi(m)}$을 만든다. 사용자 입력으로 받은 영이 아닌 경계 비율값 τ에 따라 $c_{\pi(\lfloor m \cdot \tau \rfloor)}$보다 작은 계수들을 모두 0으로 정한다.

4. 블록의 절단과 정렬 : 블록 절단은 예언 프레임에만 적용한다. 각 블록은 64개의 웨이블릿 계수를 가지고 있다. 우리는 블록 내 각 계수들 제곱의 합계(square sum)를 비교하여 블록들을 내림차순으로 정렬한다. 그 결과로서 정렬된 블록들의 배열 $b_{\pi'(1)}, b_{\pi'(2)}, ..., b_{\pi'(n)}$을 얻을 수 있다. 사용자로부터 입력받는 경계 비율값 λ에 따라 $b_{\pi'(\lfloor n \cdot \lambda \rfloor)}$보다 작은 블록들의 모든 웨이블릿 계수들을 모두 0으로 정한다.

5. 인코딩(encoding) : 전반적인 인코딩 기법은 그림 4에서 나타나 있다. 인코딩은 각 블록에 대하여 수행되어 인코딩된 블록들의 배열로 생성된다. 우리는 블록 내 64개의 계수들을 하나의 레벨 0 계수, 7개의 레벨 1 계수들, 그리고 8×7 개의 레벨 2 계수들로 분류하여 블록의 계층적인 구조를 이용한다.

한 프레임의 헤더에는 각 블록의 유의미성을 나타내는

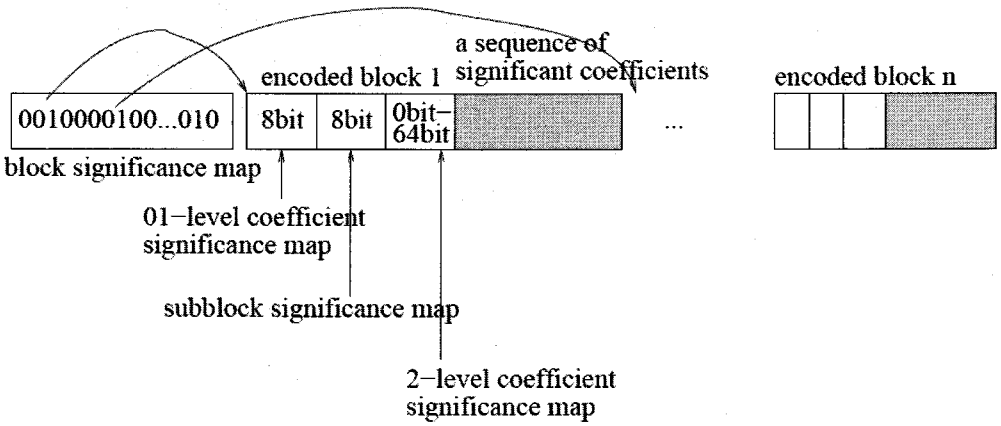

그림 4. 빠른 압축 해제 속도와 높은 압축률을 지원하는 인코딩 방법
Fig 4. Encoding scheme for fast decompression and high compression ratio

비트스트림(bit stream)이 저장되어 각 비트에 해당되는 블록이 모두 0으로 구성된 블록인지 아닌지를 나타낸다. 이것은 중요성이 적은 블록을 저장하는 추가적인 오버헤드를 피하게 해 준다. 각 블록에 대하여 하나의 비트가 할당된다.

그 다음으로, 중요성을 가지는 각 블록들에 대하여 우리는 레벨 0와 레벨 1의 계수들이 0인지 아닌지를 나타내는 8비트의 맵(map)을 저장한다. 다음으로, 각 블록 내 여덟 개의 2x2x2의 부분 블록이 영이 아닌 웨이블릿 계수를 가지고 있는지를 나타내는 8비트의 맵(map)이 저장되고, 그 다음엔 영이 아닌 웨이블릿 계수를 가지는 각각의 부분 블록내의 레벨 2의 계수들이 영인지 아닌지를 나타내는 유의미성 맵을 저장한다. 마지막으로, 영이 아닌 웨이블릿 계수들이 2바이트로 정량화되어 순서대로 저장된다.

(2) 실행 중 압축 해제

우리는 웨이블릿으로 인코드화 된 볼륨 ΔWV_k의 배열을 가지고 있기 때문에, 우리는 디코딩과 웨이블릿 역변환을 수행함에 의해서 근사화된 볼륨이미지 V_k를 얻을 수 있다. ΔV_k = inverse transformation(ΔWV_k) 이며, $V_k = V'_{k-1} + \Delta V_k$ 이다. 자체코드화 프레임인 V_i에 대하여는, $V'_{i-1} = 0$ 이며 우리는 ΔV_i로부터 V_i 를 바로 얻을 수 있다. 일단 V_i가 복원되면, 나머지 예언 프레임들은 그다음 자체코드화 프레임에 도달할 때까지 각 프레임 순서대로 디코딩 된다.

압축 해제는 블록마다 디코딩 함에 의해서 이루어 진다. 자체코드화 프레임에서는 모든 블록이 완전한 웨이블릿 역변환을 수행하여 압축 해제 된다. 반면에, 예언 프레임은 중요한 변화를 보이는 블록들만 갱신되어 그것이 최소의 압축 해제 시간으로 실제 이미지와 가능한 비슷하도록 근사한다.

인코딩된 각 블록에 대하여 다음 과정을 수행한다. 8비트인 b_1^1, ..., b_8^1을 읽고 하나의 레벨 0 계수와 일곱 개의 레벨 1 계수들이 영인지 아닌지를 결정한다. 다음으로 8비트인 b_1^2, ..., b_8^2를 읽고 각각의 여덟 부분블록들이 영의 값을 갖는지 아닌지를 결정한다. 만약 b_k^2가 1로 정해져 있다면 8비트인 c_1^k, ..., c_8^k를 읽어서 k번째 부분블록을 이루는 영이 아닌 계수들을 결정한다. k는 1에서 8까지의 정수 값을 가질 수 있다. 위에서 읽은 유의미성 맵으로부터 우리는 실제 영이 아닌 웨이블릿 계수들을 순서대로 읽을 수 있다. 모든 계수 값이 결정되면 웨이블릿 역변환을 수행하여 블록을 이루는 실제 값을 복원하며 갱신한다. 예언 프레임의 경우 역변환 하는 블록의 개수를 제한하였으므로, 압축 해제 속도 및 압축률이 높아진다.

표 4. 실험 데이터 집합 (자료 제공 : C. Bajaj)
Table 1. Information on Dataset

데이터	해상도	형식	프레임수	프레임 크기
가스	256x256x256	float	144	64MB
바다	2160x960x30	float	115	240MB

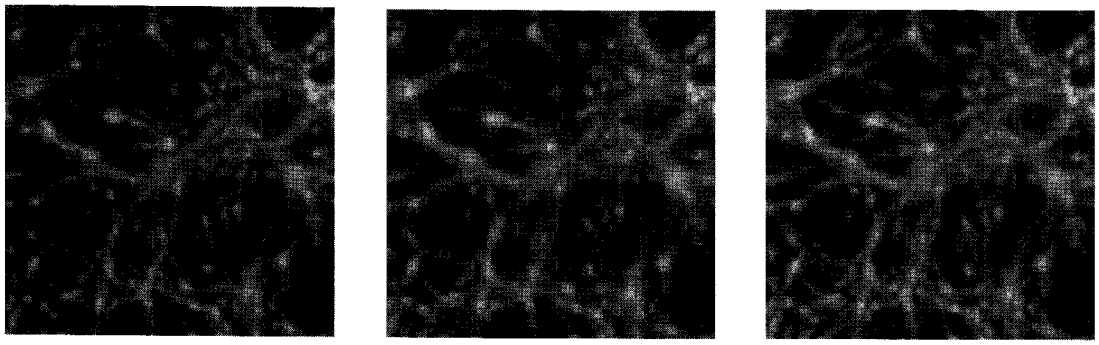

그림 5. 서로 다른 압축 변수를 이용하여 압축한 우주의 가스 밀도에 대한 시변 데이터를 볼륨 렌더링으로 가시화 한 결과. (왼쪽) 원본 데이터를 렌더링 한 것. (중간) $\tau = 1\%, \lambda = 5\%$, 압축률 145 : 1, (오른쪽) $\tau = 5\%, \lambda = 5\%$, 압축률 96 : 1
Fig 5. Volume rendering of gas density data. (left) original data, (middle) comp. ratio 145 : 1, (right) compression ratio 96 : 1

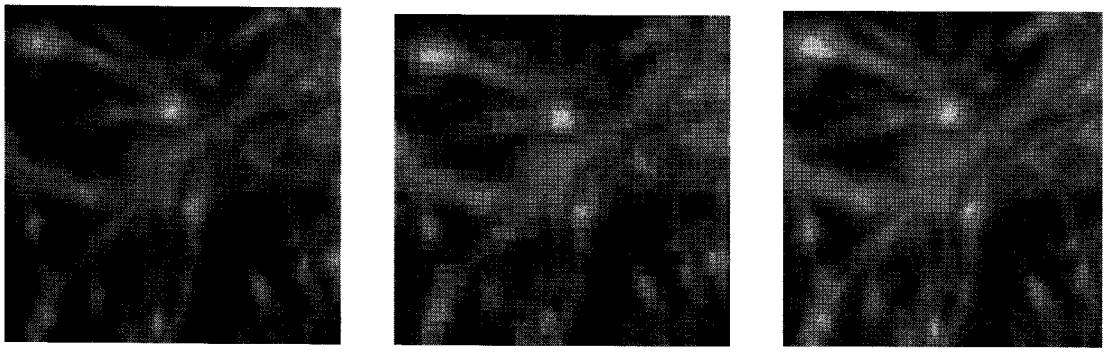

그림 6. 그림 5에서의 렌더링 결과를 확대한 그림. (중간) 압축으로 인한 이미지 왜곡을 인식하기 쉽다. (오른쪽) 압축으로 인한 이미지 왜곡을 미세하게 인식할 수 있다.
Fig 6. Zoomed images of Fig 5.

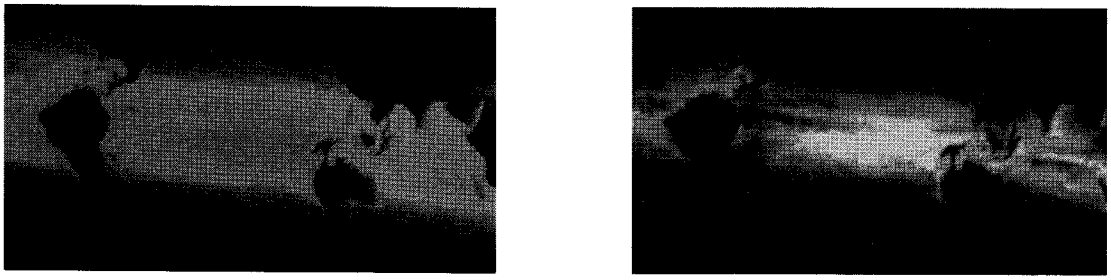

그림 7. 시간에 따른 바다 온도 분포의 변화를 나타내는 데이터를 압축후 볼륨 렌더링 한 결과. 압축 변수 $\tau = 5\%, \lambda = 5\%$ 를 사용하였고 압축률은 91.6 : 1이다.. 위의 두 그림은 서로 다른 값-색상-투명도 전환 함수를 이용하여 (왼쪽) 고온과 저온의 바다 온도 분포를 모두 가시화하였고, (오른쪽) 상대적으로 고온의 바다 온도 분포만 가시화 하였다.
Fig 7. Volume rendering of temperature changes in ocean. The time-varying data set was compressed with comp. ratio 91.6 : 1.

표 5 바다 온도 데이터에 대한 실험결과
Table 2. Results on oceanography dataset

τ, λ	압축률	압축해제시간(초) I, P
$\tau = 10\%, \lambda = 10\%$	35 : 1	8.31, 0.84
$\tau = 10\%, \lambda = 1\%$	96 : 1	8.18, 0.13
$\tau = 5\%, \lambda = 10\%$	58 : 1	7.89, 0.81
$\tau = 5\%, \lambda = 1\%$	188 : 1	7.67, 0.12

표 6 가스 밀도 데이터에 대한 실험결과
Table 3. Results on gas density dataset

τ, λ	압축률	압축해제시간(초) I, P
$\tau = 10\%, \lambda = 10\%$	40 : 1	2.4, 0.27
$\tau = 10\%, \lambda = 1\%$	93 : 1	2.3, 0.04
$\tau = 5\%, \lambda = 10\%$	66 : 1	2.16, 0.24
$\tau = 5\%, \lambda = 1\%$	148 : 1	2.17, 0.032

V. 실험 결과

압축 결과와 압축 해제 결과는 24개의 R12000 프로세서와 25GByte 메모리를 가진 실리콘 그래픽스 ONYX 2 InfiniteReality2 시스템에서 수행되었다. 삼차원 텍스쳐 매핑을 이용한 실시간 볼륨 렌더링의 구현을 위하여 OpenGL과 OpenGL Volumizer를 사용하였다.표 1은 예제 데이터들에 관한 정보를 제공한다.

우리는 다양한 조합의 입력 변수들에 대하여 자체코드화 프레임과 예언 프레임의 압축률과 압축 해제 속도를 측정하였다. 입력 변수들은 영이 아닌 웨이블릿 계수의 비율을 나타내는 τ = 1%, 5%, 10%와 영이 아닌 블록의 비율을 나타내는 λ = 1%, 5%, 10%를 사용하였다.

표 2와 표 3에서 볼 수 있듯이, 압축률은 τ와 λ에 큰 영향을 받는다. 자체코드화 프레임의 압축 해제하는 시간은 매우 크고 τ와 λ의 값에 관계없이 일정하다. 예언 프레임의 압축 해제 시간은 매우 작으며 λ값의 영향을 크게 받는다. 이 실험 결과는 우리가 설정한 압축 기법의 목표, 즉 갱신하는 블록의 개수를 줄여 압축률을 높이고 압축 해제 시간을 최소화 하는 것과 잘 맞음을 볼 수 있다. 그림 6은 두 개의 서로 다른 압축 변수값을 적용했을 때 얻게 된 이미지와 원본 이미지를 비교하여 보여준다. 본 논문에서 제안된 압축 및 압축 해제 기법을 사용하면, 비록 $\lambda = 1\%$ 와 $\lambda = 10\%$일 때 약간의 이미지 왜곡을 인식할 수 있지만, 저장 공간을 아끼고 압축된 시간 변화 볼륨 데이터를 실시간으로 재생하는 것이 가능하게 한다.

그림 8은 서로 다른 λ값을 가지고 재생된 볼륨에 대하여 시간점의 변화에 따른 각 프레임의 PSNR값을 를 보여준다. 프레임 1은 자체코드화 프레임이기 때문에 블록 절단이 이루어지지 않았다. 그래서 자체코드화 프레임에서는 서로 다른 λ값에 대하여도 PSNR이 같다. 그림 8의 그래프는 바다 온도 데이터의 경우 영이 아닌 블록들이 5%또는 10% 있더라도 PSNR이 증가하므로 자체코드화 프레임의 이미지 질을 유지하기에 충분하다는 것을 보여준다.

그림 9는 서로 다른 λ에 대하여 같은 시간 그룹 내의 각 프레임을 압축 해제하는 시간을 보여준다. 자체코드화 프레임 내 대부분의 블록들이 압축 해제 되어야 하므로 속도가 느리다. 반면, 예언 프레임의 복원은 갱신하여야 할 블록 수가 적으므로 압축 해제 시간이 훨씬 적게 든다.

그림 7은 데이터 값을 색상과 투명도 값으로 매핑하는 두 개의 다른 함수를 이용하여 바다 온도 분포를 나타내는 데이터를 가시화한 것이다. 두 번째 그림이 따뜻한 온도를 가지는 바다 부분을 보여주는 반면, 첫 번째 그림은 넓의 분포의 바다 온도를 보여준다. 두 이미지 모두 같은 압축 변수($\tau = 5\%$, $\lambda = 5\%$)를 이용하여 같은 프레임으로부터 그려졌으며 이미지의 왜곡을 찾기 힘들다.

VI. 결론

대용량의 시변 볼륨 데이터는 시간적 그리고 공간적 일관성을 가지고 있으므로 압축 기법이 저장 공간을 절약하고 데이터 가시화의 성능을 높이는데 필요하다. 그러나 모든 블록들을 인코딩하는 표준적인 비디오 압축 기법은 실행 중 압축 해제 시간이 너무 길어 적합하지 못하다. 이러한 동기로 본 논문에서는 중요하지 않은 웨이블릿 계수들과 블록들

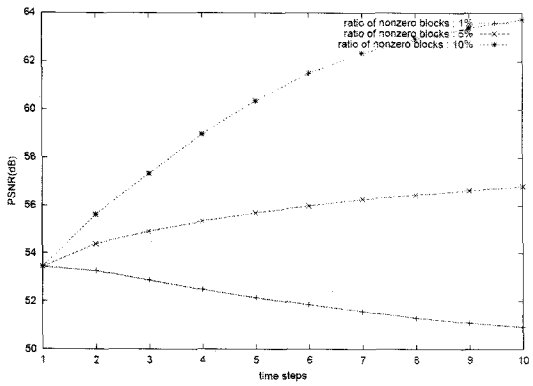

그림 8 압축된 바다 온도 데이터 각 프레임의 PSNR. 압축 변수값 $\tau = 5\%$, $\lambda = 1\%, 5\%, 10\%$.
Fig 8. PSNR of compressed oceanography dataset

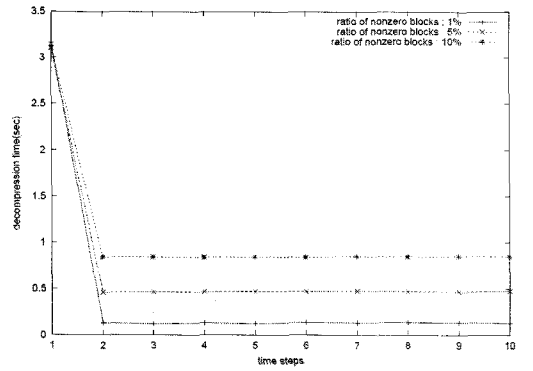

그림 9 압축된 바다 온도 데이터 각 프레임의 압축 해제 시간. 압축 변수값 $\tau = 5\%$, $\lambda = 1\%, 5\%, 10\%$.
Fig 9. Decompression time of oceanography dataset

을 절단함에 의해서 다음과 같은 목적을 가지는 압축 기법을 제안하였다. (i) 높은 압축률, (ii) 최소한의 이미지 왜곡, 그리고 (iii) 실시간 압축 해제. 실험 결과는 적은 개수의 중요한 블록들을 갱신하는 것은 높은 압축률과 이미지의 질을 유지하면서 빠른 압축 해제 속도를 달성할 수 있음을 보여준다. 끝으로 C. Bajaj, V. Siddavanahalli, S. Park, A. Thane는 본 논문을 위해 유용한 토론을 하고 실험하는데 있어 도움을 주었음을 밝힌다.

참고문헌

[1] R.A. Drebin, L. Carpenter, P. Hanrahan, "Volume Rendering", ACM SIGGRAPH, Vol. 22, No. 4, pp. 65-74, 1988

[2] R. Westermann, T. Ertl, "Efficiently Using Graphics Hardware in Volume Rendering Applications", ACM SIGGRAPH, pp. 169-177, 1998

[3] D.L. Gall, "MPEG: A Video Compression Standard for Multimedia Applications", Communications of the ACM, Vol. 34, No. 4, 1991

[4] C. Bajaj, I. Ihm, S. Park, "3D RGB Image Compression for Interactive Applications", ACM Transactions on Graphics, Vol 20, No. 1, pp. 10-38, 2001

[5] D. Ellsworth, L.-J. Chiang, H.-W. Shen, "Accelerated Time-Varying Hardware Volume Rendering Using TSP Trees And Color-Based Error Metrics", Volume Visualization and Graphics Symposium, pp. 119-128, 2000

[6] H.-W. Shen, L.-J. Chiang, K.-L. Ma, "A fast volume rendering algorithm for time-varying fields using a time-space partitioning(TSP) tree", proc. of IEEE Visualization, pp. 371-378, 1999

[7] K.-L. Ma, D. Camp, "High Performance Visualization of Time-Varying Volume Data over a Wide Area Network", Supercomputing Confernece, 2000

[8] W.B. Pennebaker, J.L. Mitchel, "JPEG Still Image Data Compression Standard", V. N. Rheinhold, 1993

[9] Sengupta, M. Hilton, B. Jawerth, "A computationally fast wavelet based video coding scheme", Digital Video Compression on Personal Computers: Algorithms and Technologies", vol. 2187 of Proc. of the SPIE, pp. 152-157, 1994

[10] K.-L. Ma, D. Smith, M.-Y. Shih, H.-W. Shen, "Efficient Encoding and Rendering of Time-Varying Volume Data", ICASE Report No. 98-22, 1998

[11] S. Guthe, W. Strasser, "Real-Time Decompression and Visualization of Animated Volume Data", Proc. of IEEE Visualization, pp. 349-356, 2001

[12] E.B. Lum, K.-L. Ma, J. Clyne, "Texture Hardware Assisted Rendering of Time-Varying Volume Data", Proc. of IEEE Visualization, pp.263-270, 2001

저 자 소 개

손 봉 수

1999년 2월 : 서울대학교 학사(전산과학)

2005년 8월 : 텍사스오스틴대학교 박사(전산과학)

2006~ 현재 : 경북대학교 컴퓨터공학과 전임강사