SBS에 기반한 이진 볼륨 데이타의 렌더링 알고리즘

(A Rendering Algorithm for Binary Volume Data based on Slice-based Binary Shell)

김 보 형 † 서 진 욱 †† 신 병 석 ††† 신 영 길 †††† 강 흥 식 †††††

(Bo Hyoung Kim) (Jinwook Seo) (Byeong-Seok Shin) (Yeong Gil Shin) (Heung Sik Kang)

요 약 이진 볼륨 데이타(binary volume data)는 외과 수술 시뮬레이션(surgical simulation)이나 컬러 볼륨 렌더링과 같이 그레이-스케일 볼륨(gray-scale volume)을 이용하기에는 부적절한 분야에서 유용하게 사용된다. 본 논문에서는 이진 볼륨을 효과적으로 표현하기 위해서 새로운 자료구조인 슬라이스 기반 이진 쉘(SBS : Slice-based Binary Shell)을 제안하고, 이 자료구조를 이용한 렌더링 알고리즘도 함께 제시한다. 슬라이스 기반 이진 쉘은 렌더링을 위해 필요한 최소한의 표면 복셀(surface voxel)들을 슬라이스 순으로 저장하고 복셀 좌표의 직접 계산을 가능하게 하기 때문에, 다중 개체(multiple objects)를 포함하고 있는 볼륨을 렌더링할 때 매우 효율적이다. 본 논문에서 제시하고 있는 슬라이스 기반 이진 쉘의 렌더링 알고리즘은 특별한 렌더링 가속 하드웨어가 없는 PC에서 100개 이상의 이진 개체들을 1초 내에 렌더링할 수 있다.

Abstract Binary volume data has its widespread use in the application of color volume rendering and surgical simulation system where gray-scale volume is inappropriate. For the efficient representation of binary volume, this paper proposes a new data structure - the Slice-based Binary Shell (SBS) - along with its rendering algorithm. Since SBS stores the minimal set of surface voxels in slice order and supports the direct computation of voxel coordinates, it shows high efficiency for rendering multiple objects. The rendering algorithm of SBS running on a PC with no specialized hardware renders more than one hundred binary objects in a second.

1. 서 론

볼륨 그래픽스(volume graphics)에서 개체는 복셀이라고 불리는 단위 볼륨 셀들의 3차원 배열로 표현된다. 이러한 볼륨 데이타는 일반적으로는 그레이-스케일 이미지(gray-scale image)들로부터 만들어지지만, 이진 이미지 볼륨(binary image volume)이 더 적합한 경우도 많이 있다. 예를 들어 인체 해부도 시스템(human anatomy atlas system)[1,2]과 같이 많은 부분에서, 비슷한 컬러 분포(color gamut)를 갖는 생체 조직들이 서로 인접해 있는 컬러 볼륨 데이타를 렌더링할 때, 조직들의 컬러값 자체를 사용하여 제대로 된 법선 벡터를 구하기는 거의 불가능하다. 이러한 경우, 각각의 조직들을 조직의 내부는 1로 조직의 외부는 0으로 이진 분할한 볼륨(binary segmented volume)과 이진 볼륨 렌더링(binary volume rendering)을 사용하는 것이 효과적이다. 이진 볼륨이 필요한 또 다른 예는 대화식 조작(interactive manipulation)이 필수적인 외과 수술 시뮬레이션 시스템이다. 외과 수술 시뮬레이션 시스템[3]에

· 이 연구는 보건복지부에서 주관한 선도기술 의료공학 기술개발사업의 지원(HMP-98-G-1-002-A)에 의하여 이루어진 것임

† 비 회 원 : 서울대학교 전산과학과
lemon@cglab.snu.ac.kr
†† 정 회 원 : 공군사관학교 전산과학과 교수
basic@cglab.snu.ac.kr
††† 비 회 원 : 인하대학교 전자전기컴퓨터공학부 교수
bsshin@cglab.snu.ac.kr
†††† 종신회원 : 서울대학교 전산과학과 교수
yshin@cglab.snu.ac.kr
††††† 비 회 원 : 서울대학교 의과대학 교수
kanghs@radcom.snu.ac.kr

논문접수 : 1999년 11월 1일
심사완료 : 2000년 3월 29일

서는 시점의 변경뿐만 아니라 개체의 위치 변경과 모양 변형(deformation)도 가능하여야 한다. 만약 이러한 시스템에서 그레이-스케일 볼륨을 이용한다면, 위치 변경이나 모양 변형 연산의 경우 볼륨의 재분류(re-classification)가 필요하기 때문에 대화식 조작이 어렵게 된다. 반면, 이진 볼륨 데이타를 이용하는 경우 일단 볼륨이 한번 만들어지고 나면 더 이상의 재분류 작업은 필요 없기 때문에 위치 변경이나 모양 변형 연산들의 대화식 수행이 보다 용이하게 이루어질 수 있다.

이러한 이진 볼륨 데이타의 효율적인 저장 공간 활용과 대화식 렌더링을 위해서는, 어떠한 볼륨 표현 방법을 사용할 것인지가 매우 중요하다. 잘 정의된 데이타 표현 방법을 사용하면 데이타 압축도 잘 할 수 있을 뿐만 아니라 렌더링 효율도 높일 수 있다. 그레이-스케일 볼륨을 이용하는 분야에서는 볼륨내의 데이타들 간의 응집성(coherence)을 이용함으로써 렌더링 비용을 줄이려는 여러 가지 방법들[4-9]이 제안되어 왔다. 이러한 방법들은 유효 복셀(valid voxel)들의 유무를 인코딩하는 공간적인 자료구조(spatial data structure)를 사용하여, 볼륨에서 투명한 복셀 영역은 렌더링하지 않고 건너뜀으로써 렌더링 비용을 줄이고 있다. 하지만 표면 복셀(surface voxel)들만이 렌더링 과정에 영향을 주는 이진 볼륨에서는 이러한 표면 복셀들만을 효율적으로 표현할 수 있는 방법이 필요하다.

Udupa는 볼륨에서의 개체 표면 표현을 위해서 반경계(semiboundary)[10]와 쉘(shell)[11] 자료구조를 제안하였다. 여기에서는 볼륨 공간을 서로 수직이 되는 3개의 평행한 평면 집합으로 분할함으로써 얻어지는 입방체들을 복셀이라고 하였다. 반경계와 쉘은 개체의 경계 주변에 있는 복셀들의 집합과 그 집합의 원소인 복셀들의 몇 가지 특성값들로 이루어진다. 반경계와 쉘은 두 개의 자료구조로 표현되는데, 하나는 그 크기가 복셀 스켄라인의 개수와 같은 포인터 배열이고 나머지 하나는 표면 복셀들의 특성값 리스트이다. 포인터 배열은 각 복셀 스켄라인의 최초 표면 복셀을 위한 특성값 리스트 상의 인덱스값을 저장한다. 이 두 가지 자료구조는 단일 개체를 표현하는데는 매우 효율적이지만, 개체의 개수가 많아지면 포인터 배열을 참조하는데 드는 비용이 상당히 증가하기 때문에 다중 개체를 표현하는데는 적합하지가 않다.

옥트리(octree)[4-6], k-d 트리[7], 그리고 런-길이 인코딩(run-length encoding)[8,9]과 같은 응집성 의존 자료구조(coherence data structure)들도 표면 표현에 사용될 수 있다. 이러한 자료구조들은 볼륨에서 유효하지 않은 영역들에 대한 쓸데없는 계산을 피하기 위해 유효 복셀들의 존재 유무를 인코딩한다. 이것들 역시 단일 개체를 표현하는데는 효율적이지만 다중 개체를 표현하는데 있어서는, 나중에 설명되어 있듯이, 본 논문에서 제시하는 방법보다 더 많은 계산 시간을 필요로 한다.

본 논문에서는 이진 볼륨의 효율적인 표현을 위해 슬라이스 기반 이진 쉘(Slice-based Binary Shell, 이하 SBS)이라는 새로운 자료구조를 제안한다. 슬라이스 기반 이진 쉘은 렌더링하는데 필요한 표면 복셀들의 최소 집합만을 슬라이스 순으로 저장하고, 렌더링 과정에서 복셀들의 좌표를 직접 계산할 수 있도록 한다. 단일 개체를 표현하는 경우에는 슬라이스 기반 이진 쉘을 이용하는 것이 다른 자료구조를 이용하는 것 보다 그다지 크게 효율적이지는 않지만, 다중 개체들을 표현하는 경우에는 상당한 성능 향상을 보인다.

본 논문의 구성은 다음과 같다. 2장에서는 새로운 자료구조인 슬라이스 기반 이진 쉘에 대하여 설명하고, 3장에서는 슬라이스 기반 이진 쉘의 렌더링 알고리즘을 알고리즘 복잡도 분석과 함께 제시한다. 4장에서는 몇 가지 실험 결과들을 보인 다음, 5장에서 결론을 맺는다.

2. 효율적인 이진 볼륨 표현법: 슬라이스 기반 이진 쉘(SBS)

2.1 슬라이스 기반 이진 쉘

이 장에서는 이진 볼륨의 효율적인 표현을 위해 본 논문에서 제안하고 있는 새로운 자료구조에 대하여 설명한다. 일반적인 영상장비(imaging device)를 이용해 얻은 데이타를 $D=(V,f,g)$ 라고 하자. 여기서 V는 정수 좌표를 갖는 복셀들의 3차원 배열이다.

$$V=\{(x,y,z)\mid 1\le x\le X_{\max},1\le y\le Y_{\max},1\le z\le Z_{\max}\} \quad (1)$$

f와 g는 볼륨 공간 V로부터 특정값으로의 함수로, $f(p)$는 복셀 p의 밀도값을 나타내고 $g(p)$는 복셀 p의 시각적 또는 물질적 성질에 대한 추가적인 정보를 나타낸다. 일례로 컬러볼륨의 경우 함수 g의 공변역은 $\{(r,g,b)\mid 0\le r\le 25,0\le g\le 25,0\le b\le 25\}$이 되고 $g(p)$는 복셀 p에서의 컬러 값을 나타낸다.

이진 볼륨은 0 또는 1의 두 가지 밀도값만으로 구성된다. 주어진 어떤 볼륨 데이타 D가 있을 때, 그것의 이진 볼륨데이타는 임계치 연산(thresholding)이나 경계 지정 연산자(boundary specification operator) 같은 이진화(binarization) 함수를 적용함으로써 얻을 수 있다. 이진화 함수 b를 적용하여 얻어진 이진 볼륨을 $D_b=(V,f_b,g_b)$라고 표기할 때, 이진화 함수 b가 임계치

연산이면 f_b는 다음과 같이 정의된다. (T_1은 하한 임계치, T_2는 상한 임계치)

$$f_b(p)=\begin{cases}1 & T_1\le f(p)\le T_2\\ 0 & otherwise\end{cases} \quad (2)$$

경계 지정 연산자가 함수 b인 경우, f_b는 다음과 같이 정의된다.

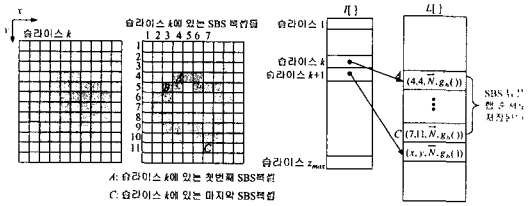
$$f_b(p)=\begin{cases}1 & b(p)\le 0\\ 0 & otherwise\end{cases} \quad (3)$$

$g_b()$는 이진 볼륨 D_b의 추가적인 고유한 특성에 대한 정보를 위한 함수이다. 예를 들어, 만약 g가 V로부터 컬러 값의 집합으로의 함수라면, g_b는 다음과 같이 정의된다.

$$g_b(p)=\begin{cases}g(p) & \text{if } f_b(p)=1\\ background\ color & \text{if } f_b(p)=0\end{cases} \quad (4)$$

위와 같은 이진화 과정을 통하여 볼륨 공간 V는 두 개의 복셀 집합으로 나누어지게 된다. 하나는 밀도값 1을 갖는 복셀들의 집합이고, 나머지 하나는 밀도값 0을 갖는 복셀들의 집합이다. 전자는 실제로 개체를 정의하는 복셀들을 포함하고 있고, 후자는 빈 공간에 해당하는 배경 복셀들을 포함한다. 자신의 26-이웃 복셀(26-neighbors)들 가운데 밀도값이 0인 복셀을 적어도 하나 이상 포함하고 있는 밀도값 1의 복셀들의 집합을 다음과 같이 정의하자.

$$SBS=\{p\in V \mid f_b(p)=1\wedge(\exists q,\ |x_p-x_q|\le 1\wedge|y_p-y_q|\le 1\wedge|z_p-z_q|\le 1\wedge f_b(q)=0)\} \quad (5)$$



(a) (b) (c)

그림 1 슬라이스 기반 이진 셀(SBS) (a) 슬라이스 k에서의 이진 화상 (b) 슬라이스 k에 있는 SBS 복셀 (c) SBS를 저장하기 위한 두 개의 자료구조

SBS를 저장하기 위해서는 두 개의 자료구조가 사용된다. 하나는 크기가 Z_{max}인 인덱스 배열 I이고, 다른 하나는 $(x,y,\vec{N},g_b())$ 튜플들의 리스트 L이다(그림 1 참조). 각각의 슬라이스 k에 대하여 배열 I는 그 슬라이스에 존재하는 SBS 튜플들의 리스트 L에서의 시작 위치를 가리키는 인덱스값을 저장하고 있다. 각 슬라이스에 있는 SBS 튜플은 행 순서(raw-major order)로 순차적으로 리스트 L에 저장된다. 그림1(c)에서 슬라이스 k에 있는 SBS 튜플들은 복셀 A에서 시작해서 복셀 C에서 끝나는 순서대로 저장되고, 배열 I의 k번째 원소는 슬라이스 k의 첫 번째 SBS 복셀인 A에 해당하는 튜플 위치를 가리키게 된다. 어떤 슬라이스가 SBS 복셀을 하나도 갖고 있지 않은 경우에는 그 슬라이스에 해당하는 인덱스 값을 -1로 하여 렌더링이 필요한 튜플이 하나도 없음을 나타내도록 하였다.

리스트 L은 투영(projection)과 음영처리(shading)를 위한 정보 및 추가적인 특성값들을 포함한다. $(x,y,\vec{N},g_b())$ 튜플에서 x와 y는 해당 SBS 복셀의 x와 y 좌표이다. $\vec{N}$은 그 복셀에서의 표면 법선 벡터로 2-바이트 인코딩 방법[12]으로 인코딩되어 저장된다. $g_b()$는 그 SBS 복셀의 추가적인 특성값들을 나타낸다.

2.2 SBS의 장점

옥트리[4-6], k-d 트리[7], 그리고 런-길이 인코딩[8,9]과 같은 응집성 의존 자료구조들과는 달리, SBS는 유효한 표면 복셀들의 좌표를 인코딩 없이 직접 저장한다. 전자는 유효한 복셀들의 존재 유무를 인코딩하고 있기 때문에 복셀의 좌표를 계산하는데 있어서 약간의 더 많은 계산을 필요로 한다. 예를 들어 런-길이 인코딩을 사용하는 경우, 유효한 표면 복셀의 좌표는 그 복셀보다 앞에 있는 모든 런들을 순회한 다음에야 결정이 된다. 그림 2는 런-길이 인코딩 방법과 SBS를 사용하여 그림 1(b)에 있는 표면 복셀들의 좌표를 계산하는 과정을 통하여 비교하고 있다. 그림 2(a)에서 보는 바와 같이 런-길이 인코딩 방법은 스캔라인 4에 있는 복셀 A의 좌표를 계산하기 위해, 4개의 투명(transparent)한 런들과 그 사이사이에 있는 모조의 비투명(nontransparent) 런 3개를 순회해야만 한다. 복셀 B의 좌표를 계산하기 위해서도, 복셀 A를 참조한 후에 4개의 런을 더 선회해야 한다. 반면, SBS는 표면 복셀들의 x와 y좌표를 저장하고 있기 때문에, 리스트 L로부터 표면 복셀들의 좌표를 곧바로 얻을 수 있다. 슬라이스 k에 있는 표면 복셀들을 참조하기 위해서는 $I[k]$가 가리키는 튜플에서부터 $I[k+1]$가 가리키는 튜플의 바로 전 튜플까지를 읽으면

된다.

응집성 의존 자료구조 사용시 좌표 계산을 위해 드는 추가적인 비용은, 소수의 개체만을 렌더링할 때에는 크게 문제가 되지 않는다. 그러나 렌더링 하는 개체의 수가 많은 경우에는 그 비용이 누적되어 상당한 성능 저하를 가져오게 된다. 반면에 SBS는 렌더링하는 개체가 늘어나더라도 개체의 수에 의해 크게 영향을 받지 않는다. 3.2절에서 자세히 설명하겠지만, SBS로 인코딩된 다중 개체들을 렌더링하는데 드는 추가적인 비용이 전체 렌더링 비용에서 차지하는 비중은 매우 작다.

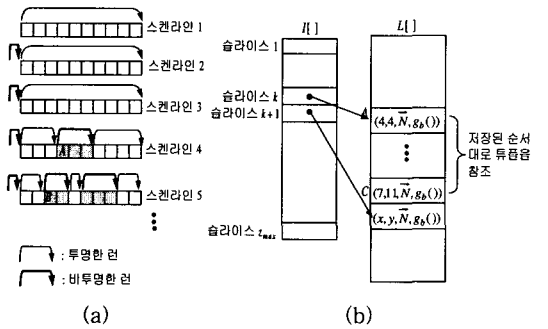


그림 2 표면 복셀 좌표 계산의 비교 (a) 런-길이 인코딩 사용 (b) SBS 사용

많은 개체들이 모여 하나의 볼륨을 구성할 때, 각각의 개체에 의해 차지되는 영역은 전체 볼륨에 비해 일반적으로 매우 작다. 이러한 사실은 유효한 표면 복셀을 하나도 갖고 있지 않은 무효 슬라이스(empty slice)의 개수가 증가한다는 것을 의미한다. SBS는 그림 3에서 볼 수 있듯이, 그러한 무효 슬라이스를 배열 I에 -1 값을 갖는 원소를 사용하여 표현한다. 또 이러한 무효 슬라이스는 렌더링할 때 -1값을 갖는 인덱스를 건너뛰으로써, 렌더링 과정에서 쉽게 제외시킬 수 있다. 이렇듯

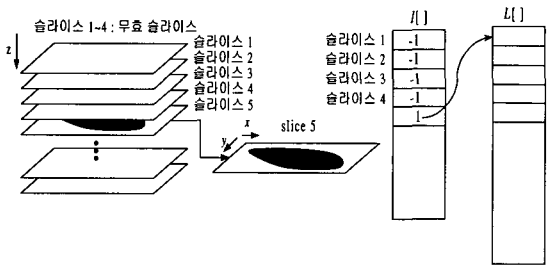


그림 3 무효 슬라이스의 표현. 무효 슬라이스는 인덱스 값 -1로 표현된다

쉽게 무효 슬라이스를 생략할 수 있기 때문에, 무효 슬라이스에 있는 모든 스켄라인들을 개별적으로 각각 제외시켜야 하는 반경계[10]에 비해 다중 개체를 렌더링할 때 성능이 상당히 향상된다.

3. SBS의 렌더링 알고리즘 및 렌더링 시간 복잡도 분석

3.1 SBS의 렌더링 알고리즘

이 장에서는 쉬어-왑 분해(shear-warp factorization)[8,9,13]를 사용하여 SBS를 렌더링하는 알고리즘을 제시한다. 렌더링 알고리즘은 크게 2 단계로 구성된다. 첫 번째 단계에서는 볼륨을 쉬어드 물체 공간(sheared object space)상의 2차원 중간 화상(intermediate image)으로 투영시키고, 두 번째 단계에서는 중간 화상을 최종 화상(final image)으로 와핑(warping)시킨다. 쉬어-왑 분해는 중간 화상에 있어서의 복셀의 변위가 동일 슬라이스 k 내에서는 변하지 않는다는 중요한 성질을 가지고 있다. 복셀 (x_p, y_p, k)이 중간 화상으로 투영된 좌표값을 정수값으로 근사시킨 것을 (x_p', y_p')라고 할 때 x_p'과 y_p'는 다음 식과 같이 나타낼 수 있다.

$$x_p' = x_p + S_x \cdot k \qquad y_p' = y_p + S_y \cdot k \tag{6}$$

여기서 S_x와 S_y는 각각 x축과 y축 방향으로의 쉬어 요소(shear component)값들이다. 가시 복셀(visible voxel) 결정을 위한 깊이 비교(depth comparison)를 피하기 위해, 가까운 슬라이스에서 먼 슬라이스 순서로 투영

```
ProjectToIntermediateImage( )
{
   //가까이에서 먼 슬라이스 순서로 투영
(1) for(k=1 ; k≤Z_max ; k++) {
(2)      u←S_x · k;      //(u,v)는 슬라이스 k에 있는 원점 복셀의 중간
          화상으로의
(3)      v←S_y · k;      //투영좌표

(4)      startInx←I[k];
(5)      if(startInx ≥ 0 ){      //startInx가 -1이면 무효 슬라이스를
          의미
(6)         endInx←I[k+1]-1; //endInx는 슬라이스 k에 있는 마지막
             SBS 튜플의 인덱스

(7)         for(i=startInx ; i≤endInx ; i++) {
               //중간 화상 화소가 배경색인 경우에만 음영처리
(8                  if( IntermediateImage[u+L[i].x][v+L[i].y] ==
                     background )

(9)                ColorShading(L[i]. N⃗, L[i].g_b());
}
```

그림 4 SBS를 중간 화상으로 투영하는 알고리즘

을 수행하였다. 이러한 순서로 투영을 수행하면 배경색을 가진 픽셀(pixel)로 투영되는 복셀들에서만 음영처리를 하면 되기 때문에 음영처리 연산도 최소화할 수 있다.

그림 4는 SBS로 인코딩된 이진 볼륨을 중간 화상으로 투영시키는 알고리즘을 보여준다. 이 알고리즘은 (4)~(9)의 명령어 라인을 렌더링하는 개체 수만큼 반복 수행함으로써, 다중 개체 렌더링을 위한 알고리즘으로 쉽게 확장될 수 있다. 중간 화상이 만들어지고 나면 2차원 와핑에 의해 최종 화상을 얻게 된다.

전처리 과정에서 x축, y축, z축을 위한 3개의 SBS 데이타 집합을 만들고, 렌더링 과정시 시각 방향 벡터(viewing direction vector)의 성분값을 살펴서 3개의 SBS 중 하나를 선택한다. 비록 3개의 데이타 집합을 가지고 있지만, 표면 복셀들만을 저장하기 때문에 SBS로 인코딩된 볼륨의 크기는 원래 볼륨보다는 훨씬 작다.

3.2 렌더링 시간 복잡도 분석

이 장에서는 SBS의 쉬어-왑 렌더링의 성능을 반경계의 쉬어-왑 렌더링과 비교한다. 볼륨 데이타는 n^3개의 복셀(각 면 당 n개의 복셀)로 구성되고, 2차원 화상은 n^2개의 픽셀로 구성되어 있다고 가정하자. 어떠한 공간적 자료구조도 사용하지 않고 수행되는 쉬어-왑 렌더링은 볼륨에 포함된 모든 복셀들에 대해서 투영과 음영처리를 수행하기 때문에 $O(n^3)$의 연산이 소요된다. 그러나 이진 볼륨은 대부분의 경우가 이진 분할에 의해 얻어진 것으로, 표면 복셀들의 수가 전체 볼륨에 비해 상당히 작다. SBS나 반경계는 그러한 표면 복셀들만을 저장하기 때문에, SBS나 반경계를 사용하면 쉬어-왑 렌더링의 복잡도를 최악의 경우인 $O(n^3)$보다 작게 줄일 수 있다.

SBS의 쉬어-왑 렌더링은 가시 복셀들만을 투영하고 음영처리한다. 가시 복셀들의 수는 가시 표면이 투영된 화상에서의 픽셀의 개수(기껏해야 n^2개)와 같으므로 볼륨의 투영과 음영처리 연산의 복잡도는 $O(n^2)$이 된다. SBS의 렌더링을 위해서는 인덱스 배열 I와 튜플 리스트 L의 참조가 추가적으로 필요하다. 인덱스 배열의 크기는 전체 슬라이스 개수와 같기 때문에 인덱스 배열 참조를 위한 복잡도는 $O(n)$이 된다. 튜플 리스트에 있는 표면 복셀들을 참조하는 데에는 $O(C_B \times n^2)$의 연산이 필요한데, 여기서 n^2은 볼륨 내의 복셀 스켄라인의 개수이고 C_B는 스켄라인 당 평균 SBS 복셀의 개수이다. 이외에도 복잡도 $O(n^2)$의 2D 와핑 연산이 필요하지만, 이것은 사용되는 자료구조와는 무관하기 때문에 렌더링 복잡도 계산에서 고려하지 않도록 한다.

반경계의 쉬어-왑 렌더링 비용은 볼륨의 투영 및 음영처리 그리고 튜플 리스트의 참조에 있어서는 SBS와 같다. 단지 인덱스 배열 참조에서만 차이를 보이는데, 반경계는 복셀 스켄라인 개수만큼의 인덱스를 저장하는 스켄라인-기반 인덱스 리스트이므로 인덱스 배열 참조를 위해서 $O(n^2)$의 연산이 필요하다.

위에서 알 수 있듯이, SBS는 인덱스 배열을 참조하는데 있어서 반경계보다 효율적인데, 이러한 효율성은 다중 개체를 렌더링할 때 더욱더 현저해진다. 볼륨이 하나의 큰 개체로 이루어져 있건 여러 개의 작은 개체들로 이루어져 있건 간에, 작은 개체들이 자치하는 영역은 그 만큼 작아지기 때문에 가시 복셀의 수와 필요한 전체 튜플 리스트의 크기는 거의 동일하게 된다. 그러나 인덱스 배열을 참조하는데 드는 비용은 개체의 수에 비례하여 증가한다. 개별적으로 SBS 인코딩된 다중 개체들의 렌더링에서는 개체 수만큼의 인덱스 배열의 참조가 필요하기 때문에 $O(S \times n)$의 연산이 소요된다. 여기서 S는 렌더링되는 개체들의 개수이다. 결과적으로 SBS의 총 렌더링 복잡도는 $O(C_B \times n^2)$이 된다. 비슷한 방법으로, 반경계로 인코딩된 다중 개체들의 인덱스 배열 참조 비용은 $O(S \times n^2)$이 되고, 총 렌더링 복잡도는 $O((C_B + S) \times n^2)$이 된다.

표 1 SBS와 반경계의 쉬어-왑 렌더링 복잡도

비용	하나의 개체		여러 개의 개체	
	SBS	반경계	SBS	반경계
볼륨의 투영 및 음영처리 비용	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n^2)$
인덱스 배열 참조 비용	$O(n)$	$O(n^2)$	$O(S \times n)$	$O(S \times n^2)$
튜플 리스트 참조 비용	$O(C_B \times n^2)$	$O(C_B \times n^2)$	$O(C_B \times n^2)$	$O(C_B \times n^2)$
총 렌더링 비용	$O(C_B \times n^2)$	$O(C_B \times n^2)$	$O(C_B \times n^2)$	$O((C_B + S) \times n^2)$

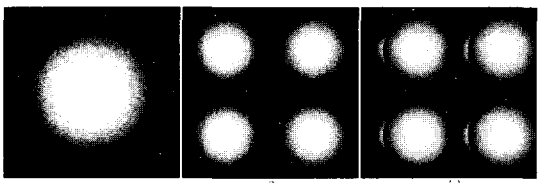
SBS와 반경계의 쉬어-왑 렌더링의 복잡도는 표 1에 요약되어 있다. 개체의 수가 많아질수록 SBS의 인덱스 배열 참조 연산의 복잡도는 증가율 n의 선형 증가를 보인다. 따라서 인덱스 배열을 참조하는데 드는 비용은 전체 렌더링 비용 계산에 있어서 지배적인 요소가 되지

못한다. 그러나 반경계 방법에서는 인덱스 배열을 참조하는데 드는 비용이 n^2의 증가율을 갖기 때문에 렌더링되는 개체들의 수가 조금만 증가하더라도 전체 렌더링 복잡도는 크게 증가하게 된다.

4. 실험 결과

이 장에서는 본 논문에서 제안하고 있는 이진 볼륨 표현 방법의 효율성을 증명하는 몇 가지 실험 결과들을 제시한다. SBS의 저장공간 및 SBS 렌더링 알고리즘의 성능은 표 2에 요약되어 있다. 렌더링 시간은 특별한 그래픽 하드웨어가 장착되어 있지 않은 Pentium III 450MHz의 CPU, 메인 메모리 128MB인 PC 상에서 측정하였다. 와핑 시간은 볼륨 표현을 위해 사용되는 자료구조와는 상관없이 최종 화상의 크기에만 영향을 받기 때문에 측정 시간에서 제외하였다. 표 2에서 "시간" 필드는 볼륨을 중간 화상으로 투영하는데 드는 시간을 천분의 일초(ms) 단위로 표시한 것이고, "저장공간" 필드는 3개의 SBS 데이타 집합의 크기를 메가바이트(Mb) 단위로 표시한 것이다. 렌더링 성능은 처음에 3개의 SBS 데이타 집합을 메인 메모리로 모두 올린 이후에 평가하였다.

성능 평가를 위해, 반지름이 100인 구 1개, 반지름이 50인 구 4개, 그리고 반지름이 50인 구와 30인 구가 각각 4개씩 포함된 8개의 구를 사용하였다. 모든 경우에서 볼륨의 크기는 256^3이며 이미지의 크기는 256×256이다. 여러 개의 구를 포함하는 볼륨에서 각각의 구는 따로따로 SBS로 인코딩 되었다. 그림 5는 위에서의 구들의 렌더링 결과를 보여주고 있다.



(a) (b) (c)

그림 5 다양한 볼륨 가시화된 구 영상. (a) 반지름이 100인 구 1개 (b) 반지름이 50인 구 4개 (c) 반지름이 50인 구와 30인 구가 각각 4개씩 포함된 8개의 구

먼저 반지름이 100인 구 1개와 반지름이 50인 구 4개의 렌더링 시간을 살펴보자. 구의 반지름이 2배 증가하면 표면적은 4배 증가하기 때문에 반지름이 100인 구의 표면 복셀의 수는 반지름이 50인 구 4개의 총 표면 복셀의 수와 거의 같게 된다. 그리고 이들 두 볼륨 데이타 집합의 가시 복셀 영역의 면적도 서로 같기 때문에, 4개의 구를 렌더링하기 위해 드는 추가적인 비용은 4개의 인덱스 배열을 참조하는데 드는 비용뿐이다. 그런데, 표에서 알 수 있듯이 반지름 100인 구 1개를 렌더링하는데 걸리는 시간은 반지름이 50인 구 4개를 렌더링하는데 걸리는 시간과 거의 같다. 따라서 4개의 인덱스 배열을 참조하는데 드는 추가적인 비용은 전체 렌더링 비용에 큰 영향을 끼치지는 못한다는 사실을 알 수 있다.

표 2 일반 PC에서의 렌더링 시간 및 저장공간 사용량

데이타 집합	볼륨 크기	시간(ms)	저장공간(Mb)
반지름이 100인 구 1개	256^3	120	3.35
반지름이 50인 구 4개	256^3	119	3.31
반지름이 50인 구 4개와 30인 구 4개	256^3	128	4.49

표 2를 보면 반지름이 100인 구 1개를 저장할 때 좀 더 많은 저장공간이 필요하고, 렌더링 시간도 조금 더 소요됨을 알 수 있다. 이는 SBS가 대각선 상에 밀도값이 0인 복셀을 가진 밀도값 1인 표면 복셀(대각 표면 복셀)들도 SBS 튜플로 저장하기 때문이다. 실제 대각 표면 복셀의 개수는 하나의 큰 물체를 표현할 때, 그와 표면적의 합이 같은 여러 개의 작은 물체를 표현할 때보다 더 많다. 따라서 반지름이 100인 구 하나를 표현할 때는 반지름이 50인 구 4개를 표현할 때 보다 렌더링 시간과 저장공간이 약간 더 필요하다.

반지름이 30인 구 4개를 반지름이 50인 구 4개로 이루어진 구 집합에 추가하면 전체 표면 복셀의 개수 및 인덱스 배열의 개수가 증가한다. 그러나 전체 렌더링 시간은 가시 복셀 영역의 크기가 비슷한 경우에는 그다지 크게 증가하지는 않는다.

위에서 설명한 렌더링 시간들은 그레이-스케일 렌더링을 위한 것이다. 컬러 렌더링의 경우에는 2개의 컬러 채널을 더 처리해야 하기 때문에 대략 2배정도 시간이 더 든다. 표 3은 SBS와 런-길이 인코딩으로 각각 인코딩된 컬러 볼륨을 쉬어-왑 렌더링 알고리즘을 이용하여 렌더링할 때의 성능을 비교하여 보여준다. "시간" 필드에 있는 부필드 "중간 화상으로의 투영" 은 SBS나 런-

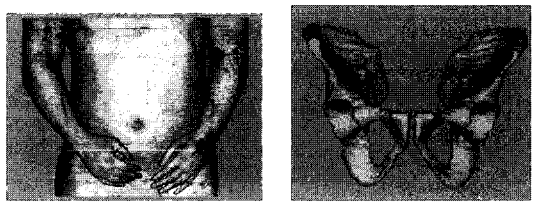
길이 인코딩 방법을 사용하여 인코딩된 볼륨을 중간 화상으로 투영하는데 걸리는 시간의 평균을 나타내고, "와핑" 부필드는 평균 와핑 시간이다. "RLE" 부필드는 런-길이 인코딩된 경우를 나타내고, "SBS/RLE (%)" 부필드는 런-길이 인코딩에 대한 SBS의 저장공간 사용량과 렌더링 시간의 비율을 백분율(percent)로 보여주고 있다.

표 3 일반 PC에서의 렌더링 시간 및 저장공간 사용량. "시간"필드는 평균 렌더링 시간을, "메모리" 필드는 SBS 혹은 런-길이 인코딩 3벌의 크기를 나타낸다.

데이타 집합	볼륨 크기	시간(ms)				저장공간(Mb)		
		중간 화상으로의 투영			와핑			
		SBS	RLE	SBS/RLE (%)		SBS	RLE	SBS /RLE (%)
몸통 피부	300×200×221	238	243	97.9	112	10.93	7.49	145
골반 뼈	300×150×218	148	147	100	94	5.62	4.17	134
신체 하지에 있는 33개의 뼈	290×180×933	150	1268	12.1	387	5.37	4.28	125
132개의 근골격계 구조물	290×180×933	567	5596	10.1	392	46.53	38.38	121

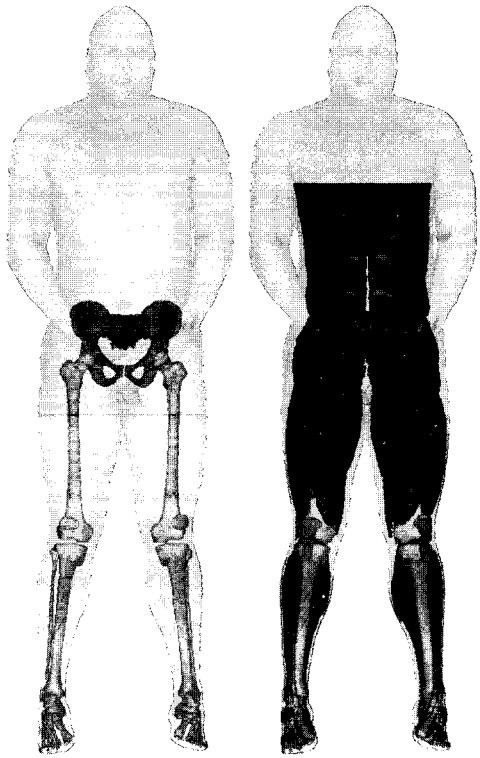
SBS는 표면 복셀의 x 좌표와 y 좌표를 함께 저장하기 때문에 런-길이 인코딩보다 대략 1.3배 정도의 더 많은 저장공간을 필요로 한다. 하지만 SBS는 좌표 계산의 부담이 없기 때문에 중간 화상으로의 투영에 있어서 런-길이 인코딩보다 좋은 성능을 보인다. 단지 1개의 개체만을 렌더링할 때에는 SBS와 런-길이 인코딩의 렌더링 속도가 비슷하지만, 개체의 수가 증가할수록 SBS의 좌표 계산에서의 효율성이 런-길이 인코딩의 효율성을 크게 능가하게 된다. SBS는 33개의 뼈로 이루어진 데이타 집합을 런-길이 인코딩을 이용한 렌더링 시간의 12%만에 렌더링할 수 있으며, 132개의 근골격계 구조물도 10%시간에 렌더링할 수 있다. 와핑 시간은 볼륨 표현을 위해 사용되는 자료구조와는 무관하기 때문에 SBS와 런-길이 인코딩 방법이 차이를 보이지 않는다.

그림 6(a)는 해상도가 $300\times200\times221$인 인간 몸통 피부의 렌더링 결과이고, 그림 6(b)는 해상도가 $300\times150\times218$인 골반 뼈의 렌더링 결과이다. 그림 6에서 사용된 데이타는 Visible Human Data 중 단면 사진촬영 영상(cryosectional photographic image)에서 해당하는 영역을 이진 분할하여 획득한 것이다. 그림 7(a)는 신체 하지에 있는 33개의 뼈들을 렌더링한 것이고, 그림 7(b)는 132개의 뼈와 근육을 포함하는 근골격계 구조물들을 렌더링한 것이다. 각각의 구조물들은 Visible Human Data로부터 개별적으로 분할, SBS로 인코딩되어 렌더링에 사용되었다. 보다 현실감 있는 렌더링을 위해 피부 전체를 반투명하게 렌더링하고 있는



(a) (b)

그림 6 (a) 인간 몸통 피부 렌더링 영상 (b) 골반 뼈 렌더링 영상



(a) (b)

그림 7 (a) 인체 하지에 있는 33개의 뼈 렌더링 영상 (b) 132개의 근골격계 구조물의 렌더링 영상

데, 표 3에 있는 시간에는 피부를 렌더링 하는데 드는 시간은 포함되어 있지 않다.

5. 결 론

이진 볼륨 데이타는 컬러 볼륨 렌더링이나 외과 수술 시뮬레이션 같이 그레이-스케일 볼륨의 사용이 부적합한 분야에서 유용하게 사용된다. 본 논문에서는 이진 볼륨의 효율적인 표현과 대화식(interactive) 렌더링을 위하여 새로운 자료 구조인 SBS를 제안하였다. SBS는 렌더링을 위해 필요한 최소한의 표면 복셀들을, 슬라이스-기반 인덱스 값을 저장하기 위한 인덱스 배열과 복셀의 좌표 및 추가적인 특성값들을 저장하기 위한 튜플 리스트라는 두 개의 자료 구조를 사용하여 저장한다. 이러한 두 개의 자료 구조를 사용함으로써 SBS는 효율적인 슬라이스-기반 볼륨 처리 및 복셀 좌표의 직접 계산을 가능하게 하여 다중 개체 렌더링에서 높은 효율성을 보인다.

여러 가지 실험 결과를 통하여 SBS 저장공간 및 SBS 렌더링 알고리즘의 성능을 다른 방법들과 비교해 보았다. SBS가 다른 방법들에 비해 약간 더 많은 저장 공간을 필요로 하기는 하지만, 다중 개체들을 렌더링할 때에는 다른 방법들보다 매우 우수함을 알 수 있었다. SBS는 렌더링 가속을 돕는 특별한 하드웨어가 부착되어 있지 않은 PC 상에서 백 개 이상의 이진 개체들을 1초안에 렌더링할 수 있다.

본 논문에서 제안하는 자료구조는 대화식 수준의 이진 볼륨 렌더링을 위한 것이다. 향후 과제로는 이진 볼륨 데이타의 보다 현실감있는 렌더링을 위한 것으로, 개체의 경계(boundary)에서 나타나는 밀도값의 급격한 차이로 인한 잡영(aliasing)을 빠른 시간안에 줄이는 방법에 대하여 연구할 계획이다.

참 고 문 헌

[1] H.S. Kang, B.H. Kim, J.W. Ryu, S.H. Hong, and H.W. Chung, S.Y. Cho, Y.H. Kim ,S.I. Hwang, D.K. Jeong, and Y.G. Shin, "The Visible Man: 3D Interactive Musculoskeletal Anatomic Atlas of the Lower Extremity," RadioGraphics Vol. 20, pp. 279-286, 2000.

[2] J.P. Kerr, M. Sellberg, P. Ratiu, D. Knapp, and C. Caon, "Photorealistic volume rendered anatomical atlases and interactive virtual dissections of The Dissectable Human," http://www.nlm.nih.gov/research/visible/vhp_conf/kerr/nlmpaper.htm

[3] R. Yagel, D. Stredney, G.J. Wiet, P. Schmalbrock, L. Rosenberg, D.J. Sessanna, and Y. Kurzion, "Building a virtual environment for endoscopic sinus surgery simulation," Computer Graphics Vol. 20, No. 6, pp. 27-41, 1996.

[4] M. Levoy, "Efficient ray tracing of volume data," ACM Transactions on Graphics, Vol. 9, No. 3, pp. 245-261, 1990.

[5] D. Laur, and P. Hanrahan, "Hierarchical splatting:A progressive refinement algorithm for volume rendering," Proceedings of SIGGRAPH '91, pp. 285-288, 1991.

[6] J. Danskin and P. Hanrahan, "Fast algorithms for volume ray tracing," 1992 Workshop on Volume Visualization, pp. 91-98, 1992.

[7] K.R. Submanian, and D.S. Fussel, "Applying space subdivision techniques to volume rendering," Proceedings of Visualization '90, pp. 150-159, 1990.

[8] P. Lacroute, and M. Levoy, "Fast volume rendering using a shear-warp factorization of the viewing transformation," Proceeding of SIGGRAPH '94, pp. 451-458, 1994.

[9] T. Kim, and Y.G. Shin, "An Efficient Wavelet-Based Compression Method for Volume Rendering," Proceedings of Pacific Graphics '99, pp. 147-156, 1999.

[10] J.K. Udupa, and D. Odhner, "Fast Visualization, manipulation, and analysis of binary volumetric objects," IEEE Computer Graphics and Applications, Vol. 11, No. 6, pp. 53-62, 1991.

[11] J.K. Udupa, and D. Odhner, "Shell rendering," IEEE Computer Graphics and Applications, Vol. 13, No. 6, pp. 58-67, 1993.

[12] A.S. Glassner, "Normal coding," In Graphics Gems, Academic Press, NewYork, pp 257-264, 1990.

[13] J. Yla-Jaaski, F. Klein, and O. Kübler O, "Fast direct display of volume data for medical diagnosis" CVGIP: Graph Models Image Processing, Vol. 53, No. 1, pp. 7-18, 1991.

김 보 형

1995년 서울대학교 전산과학과 학사. 1997년 서울대학교 전산과학과 석사. 1997년 ~ 현재 서울대학교 전산과학과 박사과정. 관심분야는 영상처리, 볼륨 그래픽스, 정보 가시화 등임.

서 진 욱

1995년 서울대학교 전산과학과 학사. 1997년 서울대학교 전산과학과 석사. 1998년 ~ 현재 공군사관학교 교수부 전산과학과 전임강사. 관심분야는 인공지능, 유전알고리즘, 볼륨 그래픽스 등임.

신 병 석

1990년 서울대학교 컴퓨터공학과 학사. 1992년 서울대학교 컴퓨터공학과 석사. 1997년 서울대학교 컴퓨터공학과 박사. 2000년 ~ 현재 인하대학교 전자전기컴퓨터공학부 전임강사. 관심분야는 볼륨 렌더링, 실시간 애니메이션, 가상현실

신 영 길

정보과학회논문지 : 시스템 및 이론
제 27 권 제 1 호 참조

강 홍 식

1977년 서울대학교 의과대학 의학사. 1980년 서울대학교 대학원 방사선과학 석사. 1985년 서울대학교 대학원 방사선과학 박사. 1985년 ~ 1987년 서울대학교 의과대학 전임강사. 1987년 ~ 1992년 서울대학교 의과대학 조교수. 1992년 ~ 1997년 서울대학교 의과대학 부교수. 1997년 ~ 현재 서울대학교 의과대학 교수. 관심분야는 방사선과학, 인체해부도 개발, 3차원 의학영상 등임.