한국산업정보학회논문지 제9권 제2호(2004. 6)

깊이버퍼 기반의 경계면 볼륨렌더링

(Boundary Surface Volume Rendering Based on Depth Buffer)

권 오 봉*, 송 주 환**, 최 성 희***
(Ou-Bong Gwun, Ju-Whan Song, Seong-Hee Choi)

요 약 이 논문에서는 경계면을 이용하여 볼륨데이터를 고속으로 가시화하는 한 방법을 제안한다. 일반적으로 경계면을 이용한 레이캐스팅은 두 단계로 처리되는데 첫 번째 단계에서는 경계면을 찾아 관측점에서 경계면까지의 거리를 3차원 버퍼에 저장하여 놓고, 두 번째 단계에서는 이 거리를 이용하여 볼륨 공간을 고속 탐색하여 가시화한다. 제안하는 방법은 첫 번째 단계에서 일반적인 방법과 다르다. 즉 첫 번째 단계에서 경계면에 관한 정보를 볼륨데이터 좌표계 주축에 직각인 관측평면에 투영하여 6개의 2차원 깊이버퍼에 저장하여 놓고 두 번째 단계에서 이 깊이 값을 이용하여 볼륨 공간을 탐색한다. 제안한 방법은 객체의 복잡도와 관계없이 처리 시간이 거의 일정하고 사용 메모리양도 볼륨데이터 공간 xy, yz, zx 평면 해상도의 2배로 항상 일정하다. 제안된 방법에 내재하는 문제점과 해결 방법에 대해 고찰하고 구현 예를 보인다.

핵심주제어 : 3차원 가시화, 볼륨렌더링, 레이캐스팅, 경계면, 깊이 버퍼

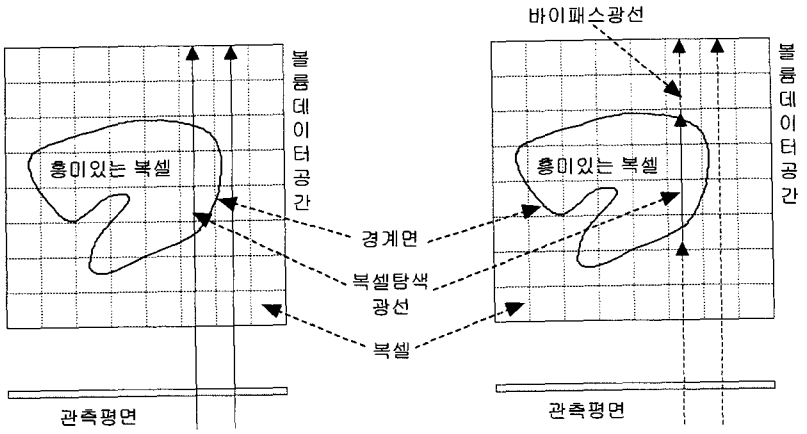
Abstract This paper focuses on a boundary surface based ray casting. In general the boundary surface based ray casting is processed in two stages. The first stage finds boundary surfaces and stores them into buffers. The second stage calculates a distance from a viewpoint to the voxels of the interested area by projecting boundary surfaces on the view plane, and then starts to traverse a volume data space with the distance. Our approach differs from the general boundary surface based ray casting in processing the first stage of it. Contrast to the typical boundary surface based ray casting where all boundary surfaces of volume data are stored into buffers, they are projected on the planes aligned to the axis of volume data coordinates and these projected data are stored into 6 buffers. Such maneuver shortens time for ray casting, and reduces memory usage because it can be carried out independently from the amount of the volume data.

Key Words : 3D visualization, volume rendering, ray-casting, boundary surface, depth buffer

1. 서 론

과학 가시화에서 자주 사용하는 데이터는 규칙적인 3차원 볼륨데이터로 전자밀도지도(electron density maps), 유체역학시뮬레이션, MRI(Magnetic Resonance Imaging), CT(Computed Tomography) 등의 의료 영상 데이터, 은하 광파장 데이터(light wavelength data for galaxies) 등이 이에 해당한다. 볼륨데이터를 3차원으로 가시화하여 모니터

* 전북대학교 전자정보공학부
** 전주대학교 교양학부
*** 전주공업대학교 유아교육학과



[그림 1] 단순레이캐스팅과 경계면을 이용한 레이캐스팅의 비교

에 표시하면 데이터 속에 내재되어 있는 현상을 쉽게 파악할 수 있어 연구, 개발, 진단 등을 용이하게 수행할 수 있다. 3차원 볼륨 데이터를 실시간으로 가시화하려는 시도는 현재 성과를 이루어 Cube-4/VolumePro 같은 볼륨렌더링 전용처리장치는 256×256해상도의 영상을 초당 30프레임 생성한다[1]. 그러나 3차원 가시화의 보급과 함께 사용자들의 요구도 증대하여 보다 높은 해상도의 3차원 영상을 소프트웨어로 더구나 실시간으로 처리 할 것을 요구하고 있다. 이러한 요구에 부응하여 이 논문에서는 범용 개인용 컴퓨터에서 소프트웨어만으로 볼륨데이터를 고속화할 수 있는 경계면을 이용한 레이캐스팅을 제안한다.

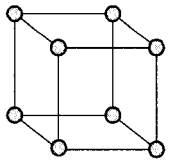
경계면을 이용한 레이캐스팅은 경계면을 이용한 레이트레이싱에서 출발하였다. 경계면을 이용한 레이트레이싱에서는 물체 공간(object space)을 계층적인 부분 공간으로 나누어 광선과 만나는 부분 공간 내에 있는 물체만을 처리하여 고속화를 도모한다[2]. 레이캐스팅 처리의 대부분은 광선의 진행 방향에 따른 복셀 탐색으로 이 처리의 고속화가 레이캐스팅 고속화의 관건이다. [그림 1]이 보이는 것과 같이 레이캐스팅을 처리하는데 있어서 경계면을 이용하면 광선이 경계면 앞에 있는 복셀을 탐색하지 않고 관측점에서 경계면까지 한꺼번에 진행하여 고속화를 이룰 수 있다.

그런데 경계면을 이용한 레이캐스팅의 전제 조건은 관측점에서 경계면까지의 정확한 거리를 아는 것이다. 이것을 알기 위해서 K. J. Zuiderveld 등과 M. Sramek 등은 관측점에서부터 흥미 있는 복셀까지 거리지도(distance map)를 만들어 이것을 이용하여 볼륨 공간을 탐색하였다[3][4]. 비록 이 방법은 볼륨 공간 탐색은 고속화하였으나 거리 지도를 만드는데 방대한 메모리를 필요로 하여 인터랙티브한 처리를 어렵게 하였다. L. M. Soberjski 등과 M. Wan 등은 경계면을 이용한 레이캐스팅을 개선하였다[5][6][7]. L. M. Soberjski 등의 PARC는 전처리 단계에서 경계면에 해당하는 폴리곤을 구하여 저장하고 본 처리에서 깊이버퍼(Z-buffer)알고리즘을 이용하여 이 폴리곤을 관측평면에 투영하여 관측점에서 경계면까지의 거리를 구하여 볼륨 공간의 탐색에 이용하였다. PARC는 여러 가지 폴리곤을 조합하여 경계면을 구할 수 있었으나 폴리곤의 수가 많은 경계면에서는 다각형의 투영시간이 증가하는 단점이 있었다.

M. Wan 등의 경계 셀 기반의 렌더링에서는 흥미 있는 복셀과 그렇지 않은 복셀의 경계를 이루는 셀을 구하여 이것을 벡터 테이블에 저장하고 본처리에서 관측평면에 투영하여 관측점과 흥미 있는 복셀과의 거리로 사용하였다. 그러나 이 방법은 벡터 테이블에 필요한 방대한 메모리 때문에 복셀수가 보다 적은 볼륨 공간에는 효과가 있었으나 보다 큰 공간에는 적용이 어려웠다. 또 M. Wan 등의 고성능가속(presence- accelerated) 레이캐스팅에서는 경계셀을 런렝스 인코딩(run length

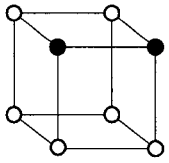
encoding)하여 테이블에 저장하고 이를 병렬화 하였다. 이 방법은 전처리 단계에서 복셀을 분류하기 때문에 인터랙티브한 등고면(iso-surface)의 처리를 어렵게 하였다. J. Choi[8] 등은 영상공간 경계면을 이용하여 복셀 탐색을 고속화하였으나 관측점이 바뀔 때마다 경계면을 다시 계산하여야 했기 때문에 인터랙티브한 애니메이션이 어려웠다.

이 논문에서는 경계면을 이용한 고속레이캐스팅을 제안하는데 다른 방법과의 차이점은 경계면을 볼륨 공간의 좌표축에 평행한 6개의 관측평면



○ 복셀

[그림 2] 셀 정의



● 흥미있는 복셀
○ 흥미없는 복셀

[그림 3] 경계 셀

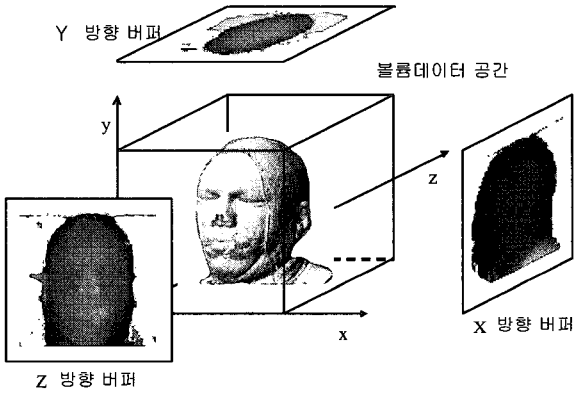
에 투영하여 이것을 깊이버퍼에 저장하고 증분 알고리즘(increment algorithm)만으로 관측면에서 흥미 있는 복셀까지의 거리를 구할 수 있는 점이다. 이 방법은 객체의 복잡도에 관계없이 레이캐스팅 시간을 일정하게 하며 비교적 작은 메모리 공간을 사용할 수 있게 한다.

2. 6면 깊이버퍼

6면 깊이버퍼는 [그림 4]와 같이 경계면을 저장하기 위하여 설치한다. 경계면은 관심 대상이 되는 복셀을 둘러 싼 면을 말한다. 셀은 [그림 2]와 같이 8개의 정점에 복셀 값을 갖는 큐브를 말하고 경계 셀은 [그림 3]과 같이 1개에서 7개까지의 흥미 있는 복셀을 갖는 셀을 말한다. 경계면은 경계셀로 이루어진다. 레이캐스팅 처리를 하는데 있어서 흥미 있는 복셀 만을 탐색할 수 있다면 처리는 고속화된다.

볼륨데이터 구조는 사각형의 3차원 배열로 표시되며 [그림 4]와 같이 볼륨데이터 공간 좌표계의 주축이 볼륨데이터 공간을 이루는 육면체의 에지가 된다. 간단히 설명하기 위하여 볼륨데이터 공간 좌표계와 관측 공간 좌표계가 [그림 4]에서와 같이 같다고 가정하자. 만일 평행 투상을 한다면 관측 방향은 양의 z 축이 되고 관측평면은 xy 면이 된다. 전형적인 레이캐스팅에서는 그림 1(a)에서와 같이 광선이 xy 면에서 출발하여 z 축을 따라 진행하며 불투명도와 복셀 값을 해당하는 픽셀에 축적할 것이다.

그러나 경계면을 이용한 레이캐스팅에서는 그림 1(b)에서와 같이 광선이 경계면에서 출발한다. 이 경계면의 깊이 값을 저장하기 위하여 x, y, z 축에 수직하게 yz(x 방향 깊이버퍼), xz(y 방향 깊이버퍼), xy(z 방향 깊이버퍼) 면에 대하여 근거리, 원거리 버퍼 총 6개의 깊이버퍼를 설치한다. 깊이버퍼의 해상도는 해당하는 방향의 볼륨데이터 공간의 해상도와 같게 한다.



[그림 4] 6면 깊이버퍼 개념 (3면만 표시하였음)

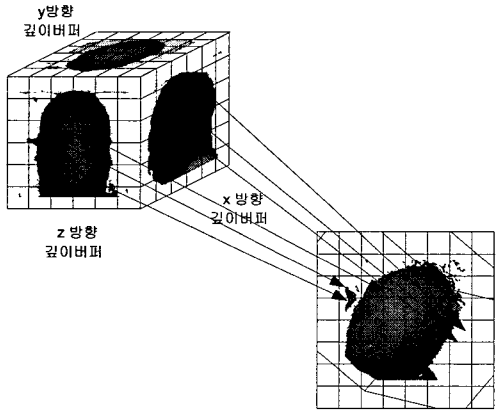
볼륨데이터 공간의 좌표계가 관측 공간의 좌표계와 같을 경우에는 깊이버퍼에 있는 값을 그대로 관측평면에서 경계면까지의 거리로 사용하면 된다. 그러나 볼륨데이터 공간의 좌표계와 관측 공간의 좌표계가 항상 같은 것이 아니기 때문에 관측점이 변하면 이것에 맞추어 깊이버퍼에 있는 값도 변환해야 한다. 깊이버퍼의 값은 관측점의 변환에 관계없이 한 번만 계산하면 되기 때문에 깊이버퍼 값을 미리 계산하여 볼륨데이터의 헤더에 보관하여 계속하여 사용한다.

다음은 z 방향 깊이버퍼에 z 값을 계산하는 알고리즘이다. x 방향, y 방향 깊이 버퍼도 같은 방법으로 계산한다.

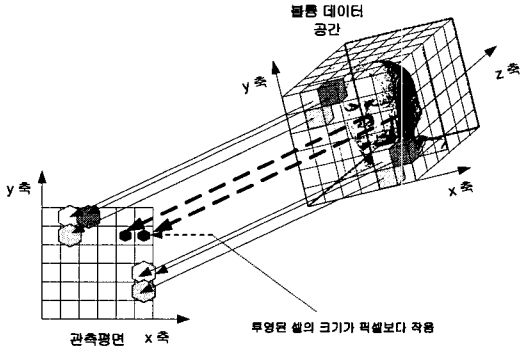
```
for all pixels(i, j) on the xy view plane
{z(i, j) = 65536;}   /* z 버퍼의 초기화 */
for all pixels(i, j) on the xy view plane
{
   spawn a ray(i, j); /* 관측평면(i, j)점에 해당하는
                         광선 생성 */
   traverse voxel of the ray(i, j)
         through the volume data space;
   if(ray(i, j) meets the voxel of interested area
         for the first time)
   {
      finds the distance of ray(i, j)
         from xy view plane to the voxel;
      z(i, j) = the distance of ray(i, j);
   }
}
```

3. 깊이버퍼의 관측 변환

관측 방향이 바뀌면 이 방향에 대하여 관측 좌표계를 만들고 이곳으로부터 근거리, 원거리 깊이 값이 구해야 한다. 새로운 관측 방향에 직교하는 관측 평면을 원점에 놓고 관측 평면의 각 픽셀에서 출발한 광선들이 볼륨데이터를 통과할 때 제일 처음 만나는 관심 복셀의 깊이 값을 관측점에서 경계면까지의 거리로 사용한다. [그림 5]는 6면 깊이버퍼 값을 이용하여 새로운 관측



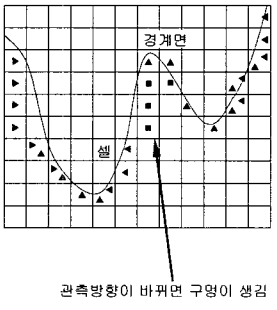
[그림 5] 새로운 관측방향의 깊이 값 계산 방법



[그림 6] 셀의 풋프린트 크기가 픽셀의 크기보다 작기 때문에 구멍이 생김

방향의 깊이 값을 구하는 과정을 보인다. 6 개의 깊이버퍼들 중 관측 방향에서 보아서 앞에 있는 3 개의 깊이버퍼 값을 새로운 관측 평면에 평행한 버퍼에 투영하면서 z 값이 제일 작은 좌표를 깊이버퍼에 저장한다. 데이터 볼륨 좌표계를 기준으로 x 방향 깊이버퍼에 있는 한 점(x_d,y_d,z_d)을 새로운 관측 방향에 대한 점으로 계산하는 알고리즘은 다음과 같다[9].

```
for(yd = 0; yd < ny - 1; yd++){
   for( zd = 0; zd< nz - 1; zd++){
      xv,yv,zv  =  view_transform(x_buffer(yd,zd),  yd,  zd);
 }
}
```



샘플링 가능성 및 방향

샘플링 가능성 및 방향	Symbol
가능, 우측	▶
가능, 좌측	◀
가능, 상측	▲
불가능	■

[그림 7] 깊이버퍼 수의 제한 때문에 발생하는 구멍

여기에서 view_transform(x_buffer(y_d, z_d), y_d, z_d)는 식 (1)로 구한다.

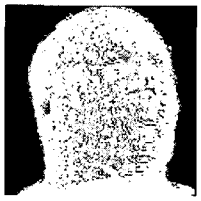
$$\begin{bmatrix} x_v \\ y_v \\ z_v \end{bmatrix} = \begin{bmatrix} \cos\alpha\cos\theta - \sin\alpha\sin\theta\sin\beta & \cos\alpha\sin\theta - \sin\alpha\sin\beta\cos\theta & -\sin\alpha\cos\beta \\ -\sin\theta\cos\beta & -\cos\theta\cos\beta & \sin\beta \\ \sin\alpha\cos\theta\sin\theta\sin\beta\cos\alpha & \sin\theta\sin\alpha + \sin\beta\cos\alpha\cos\theta & \cos\alpha\cos\beta \end{bmatrix} \begin{bmatrix} x_d \\ y_d \\ z_d \end{bmatrix} \quad (1)$$

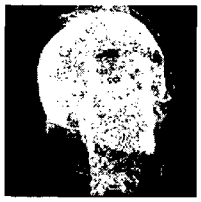
위의 식에서 ny-1와 nz-1는 각각 y, z 방향으로 볼륨데이터 해상도이고 x_d,y_d,z_d는 볼륨데이터 좌표계에서 임의의 한점의 좌표 값이고 x_v,y_v,z_v는 관측 변환 후의 그 점의 좌표이다.

위의 알고리즘은 x 방향에 대한 변환을 표시하는데 y 방향 및 z 방향에 대해서도 같은 처리를 한다. 관측 변환은 볼륨데이터 공간의 중앙에 관측 좌표계의 중심을 설정하고 y축을 중심으로 α도 회전, x축을 중심으로 β도 회전, z축을 중심으로 θ회전으로 이루어진다.

4. 내재하는 문제점 및 해결 방법

제안한 방법은 두 가지 문제점을 가지고 있다. 첫 번째 문제점은 [그림 6]에서와 같이 앞에 있는 3개의 깊이버퍼에 저장된 셀을 관측평면의 깊이버퍼에 투영하였을 경우에 생기는 풋프린트의 크기가 관측 평면의 픽셀 크기보다 작을 때 경계면에 구멍이 생겨 레이캐스팅을 수행하면 품질이 저하된 [그림 8]과 같은 3차원 영상을 생성하는 점이고 또 하나의 문제점은 6개의 버퍼만을 이용하여 3방향에서 본 경계면의 값들을 저장하였기 때문에 관측 방향이 바뀌면 [그림 8]과 같이 구멍이 생기는 점이다. [그림 7]은 관측 방향이 변하면 구멍이 생기는 이유를 보인다.


디지털맨 피부 데이터


디지털맨 두개골 데이터

[그림 8] 풋프린트 크기와 깊이버퍼수의 제한 때문에 발생하는 구멍

이러한 문제점은 영상기반렌더링에서 샘플링이 불완전해서 생기는 것과 유사한 것으로 영상기반렌더링에서는 주위에 있는 픽셀 값을 이용하여 구멍을 채우는데 이 논문에서도 같은 방법을 사용하였다. 새로운 관축 방향에 대하여 관측 변환을 하는 과정에서 구멍이 생기면 구멍의 네 이웃에 대하여 깊이 값을 찾아 이중에서 제일 작은 값을 새로운 관측 방향에 대한 깊이 값으로 하였다. 만일 4이웃에 깊이 값이 존재하지 않으면 8이웃 12이웃... 등으로 확대하여 찾았다. 이러한 방법으로 문제점을 해결한 것을 [그림 9]에 보인다.


디지털맨 피부 데이터


디지털맨 두개골 데이터

[그림 9] 셀의 크기를 확대하여 구멍을 제거한 경계면

5. 구현 및 평가

제안한 알고리즘이 속도 향상에 기여하는 정도를 알아보기 위하여 고속화 기법을 사용하지 않은 단순 레이캐스팅, PARC, 제안한 알고리즘을 Pentium III(933MHz)와 512MB 주기억을 장착한 개인용 컴퓨터에서 Visual C++로 구현하여 비교하였다. 벤치마크 데이터로는 오픈 도메인에 있는 CT 엔진을 사용하였다.

[표 1]-[표 2]와 [그림 10]-[그림 11]은 단순 레이캐스팅, PARC, 제안 알고리즘의 메모리 사용량, 처리시간, 처리시간 향상률, 생성된 영상을 보인다. 단순 레이캐스팅은 추가 메모리가 필요없으나 PARC와 제안 알고리즘은 추가 메모리가 필요하다. PARC는 경계면을 구성하는 각 복셀(다각형)의 x, y, z 좌표를 메모리에 저장하기 때문에 간단한 경우에는 메모리가 그다지 필요하지 않으나 복잡해질수록 이에 비례하여 증가한다. 만일 경계면을 구성하는 복셀 수가 n이라 하고 각 좌표를 2 Byte 정수로 저장한다고 하면 메모리 양은 $n\times3\times2$ Bytes 즉 복셀의 6배에 상당하는 메모리가 필요하다. 이에 비하여 제안 알고리즘은 경계면의 복잡도에 관계없이 볼륨데이터 공간 xy, yz, zx 평면 해상도의 2배 메모리가 필요하다. PARC 알고리즘과 제안 알고리즘은 깊이 값을 구하는 시간이 추가로 필요하지만 아주 짧다. 관측 방향이 변하면 PARC는 관측평면을 구성하는 깊이 값을 모두 처리해야하나 제안 알고리즘은 앞에 있는 3 면에 대한 깊이 값만을 이용하기 때문에 깊이 값을 구하는 시간도 제안 알고

리즘이 PARC 보다 짧다.

제안한 방법으로 생성한 영상이 단순 알고리즘으로 생성한 영상과 얼마나 차이가 나는가를 영상 평가지표인 평균절대값오차(MAE; Mean Absolute Error)와 평방자승근오차(RMSE; Root Mean Square Error)를 이용하여 평가하였다. MAE 와 RMSE는 식 (2)와 식 (3)으로 정의된다.

$$MAE = \sum_{i,j} |A(i,j) - B(i,j)|/N^2 \quad (2)$$

$$RMSE = \sqrt{\sum_{i,j} [A(i,j) - B(i,j)]^2/N^2} \quad (3)$$

여기서 $A(i,j)$, $B(i,j)$는 각각 A, B 영상의 (i,j)번째 픽셀값을 N^2는 영상의 해상도를 의미한다. 이식은 0~1의 값을 갖고 1에 가까울수록 비교 영상 간의 차이가 나는 것을 의미한다.

[표 1]과 [그림 10]은 구와 구멍이 있는 볼륨데이터를 이용한 실험 결과를 보인다. 제안 알고리즘의 메모리 사용량은 PARC의 메모리 사용량보다 구, 구멍 뚫린 구에 대하여 각각 199,080 Bytes, 415,080 Bytes가 적었다. 구의 처리 시간은 제안 알고리즘이 단순 레이캐스팅과 PARC 알고리즘에 비하여 각각 3.741배, 1.165배 빨랐고 구멍 뚫린 구의 처리 시간은 제안 알고리즘이 단순레이캐스팅과 PARC 알고리즘에 비하여 각각 5.650배, 1.387배 빨랐다. 단순레이캐스팅과 제안 알고리즘으로 생성한 볼륨 영상의 차이는 구, 구멍 뚫린 구에 대하여 MAE, RMSE 모두 0.000214 이하였다.

[표 2]와 [그림 11]은 CT Engine 볼륨데이터를 이용한 실험결과를 보인다. 제안 알고리즘의 메모리 사용량은 PARC의 메모리 사용량보다 엔진 표피, 엔진 표피와 내부의 동시 표현에 대하여 모두 319,786 Byte 적었고, 엔진 내부에 대해서는 36,680 Byte 많았다. 제안 알고리즘이 엔진 표피, 엔진 내부, 표피와 내부 동시 표현에 대하여 단순 레이캐스팅 보다 각각 3.797배, 11.678배, 2.387배 처리 속도가 빨랐고 PARC 보다는 2.516배, 2.259배, 1.755배 처리 속도가 빨랐다. 단순레이캐스팅과 제안 알고리즘으로 생성한 볼륨 영상의 차이는 엔진 표피, 엔진 내부, 표피 내부 동시 표현에 대하여 MAE, RMSE 모두 0.071824 이하였다.

결론적으로 말하면 제안된 알고리즘, 단순 레이캐스팅, PARC 알고리즘으로 생성한 볼륨 영

[표 1] 구 및 구멍 뚫린 구의 볼륨데이터의 처리시간 및 속도 향상표

벤치마크 데이터			구 (256×256×256 복셀)	구멍 뚫린 구 (256×256×256 복셀)
추가메모리 (Bytes)		(a)	0	0
		(b)	985,512	1,201,512
		(c)	786,432	786,432
탐색시간 (초)		(a)	2.323	2.785
		(b)	0.287	0.350
		(c)	0.091	0.071
렌더링 시간 (초)		(a)	0.752	0.560
		(b)	0.671	0.471
		(c)	0.731	0.521
총처리(탐색 + 렌더링) 시간 (초)		(a)	3.075	3.345
		(b)	0.958	0.821
		(c)	0.822	0.592
처리시간 향상률		(a)/(c)	3.741	5.650
		(b)/(c)	1.165	1.387
결과 영상의 오차	MAE	(a):(c)	0.000001	0.000214
		(b):(c)	0.000001	0.000125
	RMSE	(a):(c)	0.000125	0.001012
		(b):(c)	0.000147	0.000982

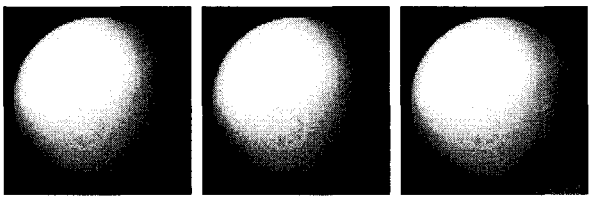
(a) 단순 레이캐스팅 (b) PARC (c) 제안 알고리즘

[표 2] CT 엔진 데이터의 처리 시간 및 속도 향상률

벤치마크 데이터			엔진 표피 (256×256×110)	엔진 내부 (256×256×110)	표피와 내부 동시표현 (256×256×110)
추가메모리 (Bytes)		(a)	0	0	0
		(b)	807,210	450,744	807,210
		(c)	487,424	487,424	487,424
탐색시간 (초)		(a)	1.654	4.561	1.660
		(b)	0.965	0.578	0.933
		(c)	0.040	0.038	0.041
렌더링 시간 (초)		(a)	0.643	0.437	1.154
		(b)	0.557	0.389	1.136
		(c)	0.565	0.390	1.138
총처리(탐색 + 렌더링) 시간 (초)		(a)	2.297	4.998	2.814
		(b)	1.522	0.967	2.069
		(c)	0.605	0.428	1.179
처리시간 향상률 (총 시간 비율)		(a)/(c)	3.797	11.678	2.387
		(b)/(c)	2.516	2.259	1.755
결과 영상의 오차	MAE	(a):(c)	0.000010	0.005900	0.012809
		(b):(c)	0.000009	0.001217	0.08721
	RMSE	(a):(c)	0.001211	0.049934	1.071824
		(b):(c)	0.000721	0.025214	0.032398

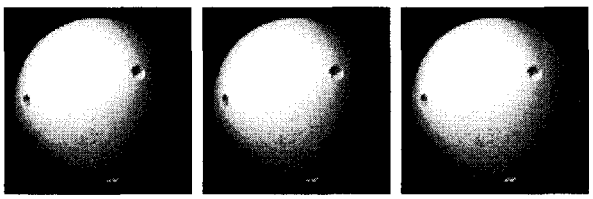
(a) 단순 레이캐스팅 (b) PARC (c) 제안 알고리즘

상의 질적 차이는 그다지 없었으나 제안된 알고리즘의 속도가 단순 레이캐스팅의 속도 보다 약 2~12배, PARC 속도 보다 약 1.1~2.5배 정도 빨랐다. 특히 제안 알고리즘은 2개의 면보다 1개의 면을 레이캐스팅 할 때 PARC 알고리즘보다 우수하였다.



(a) 단순 레이캐스팅 (b) PARC (c) 제안 알고리즘

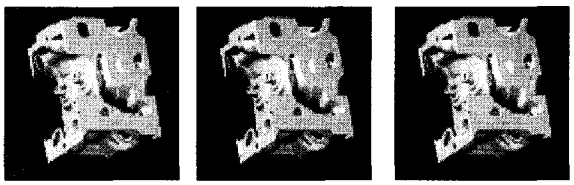
(구)



(a) 단순 레이캐스팅 (b) PARC (c) 제안 알고리즘

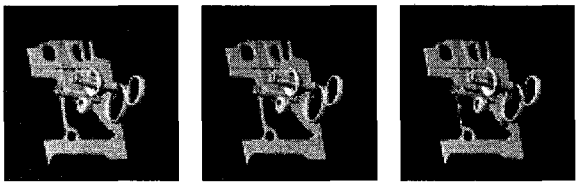
(구멍뚫린 구)

[그림 10] 인공데이터로 생성한 3차원 영상의 비교



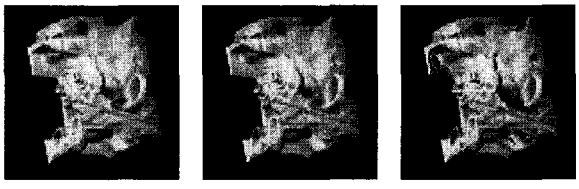
(a) 단순 레이캐스팅 (b) PARC (c) 제안 알고리즘

(엔진 표피)



(a) 단순 레이캐스팅 (b) PARC (c) 제안 알고리즘

(엔진 내부)



(a) 단순 레이캐스팅 (b) PARC (c) 제안 알고리즘

(표피와 내부)

[그림 11] CT 엔진 데이터로 생성한 3차원 영상 비교

6. 결 론

이 논문에서는 경계면을 볼륨데이터 좌표계의 주축에 직각인 6면 깊이버퍼에 저장하여 놓고 이 버퍼의 내용을 변환된 관측평면에 투영하여 깊이 값을 구하여 이것을 바탕으로 흥미 있는 복셀까지 광선을 바로 진행시키는 경계면 기반의 고속 레이캐스팅을 제안하였다. 그리고 제안한 방법으로 시스템을 구현하여 벤치마크 구, CT 엔진 볼륨데이터에 대하여 성능을 평가하였다. 제안된 방법의 레이캐스팅 시간은 0.428~1.179초로 단순 레이캐스팅에 비해서는 약 2~12배, PARC에 비해서는 약 1.1~2.5배 빨랐다. 제안 알고리즘이 사용하는 메모리양은 객체의 복잡도에 관계없이 볼륨데이터 공간 xy, yz, zx 평면 해상도의 2 배로 항상 일정하였다. 즉 제안 알고리즘은 물체의 복잡도에 관계없이 추가되는 메모리양이 일정하고 비교적 처리 시간도 빠르다. 향후 깊이버퍼 값을 구조화하는 방법을 고안하여 레이캐스팅 속도를 더욱 개선할 예정이다.

참 고 문 헌

[1] A. Kaufman, F. Dachille IX, B. Chen, I. Bitter, K. Kreeger, N. Zhang, and Q. Tang, "Real-Time Volume Rendering," International Journal of Imaging Systems and Technology, Vol. 11, No 1, pp44-52, May 2000.

[2] S. M. Rubin and T. Whitted. "A 3-dimensional representation for fast rendering of complex scenes," In SGGRAPH 80 Proceedings. ACM, New York, pp.110-116, 1980.

[3] M. Sramek and A. Kaufman, "Fast Ray-Tracing of Rectilinear Volume Data Using Distance Transforms," IEEE Transactions on Visualization and Computer Graphics, Vol.6, No. 3, pp 236-252, July-September 2000.

[4] L. M. Sobierajski and R. S. Avila, "A Hardware Acceleration Method for Volumetric Ray Tracing," IEEE Visualization 95, pp 27-34, 1995.

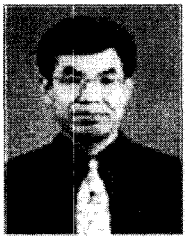
[5] M. Wan, S. Bryson and A. Kaufman. "Boundary cell-based acceleration for volume ray casting," computers & graphics, Vol 22 No 6, pp 715-721, 1999.

[6] M. Wan, A. Kauman and S. Bryson. "High performance presence accelerated ray casting," IEEE Visualization 99, pp 379-386, 1999.

[7] J.J. Choi, B.S. Shin, Y.G. Shin and K. Cleary. "Efficient volumetic ray casting for isosurface rendering," computers & graphics, Vol 24, pp 661-670, 2000.

[8] S. Yang and T. Wu, "Improved Ray- casting Algorithm for Volume visualization," SPIE Conference on Visualization and Data Analysis, San Jose Convention Center, pp. 319-326, Jan 20-25, 2002.

[9] http://www.nlm.nih.gov/research/visible/visible_human.html NLM The Visible Human Project



권 오 봉 (Ou-Bong Gwun)

1980년 고려대학교 전기공학과 학사
1983년 고려대학교 전기공학과 공학석사
1993년 일본구주대학교 총합이공학연구과 공학박사
1994년~1995년 전북대학교 컴퓨터과학과 전임강사
1996년~1999년 전북대학교 조교수
2000년~현재 전북대학교 부교수
(관심분야 : 컴퓨터 그래픽스, 사이언티픽 비주얼라이제이션)

송 주 환 (Ju-Whan Song)

1995년 전주대학교 전자계산학과 학사
1997년 전북대학교 전산통계학과 이학석사
2003년 전북대학교 전산통계학과 이학박사
2001년~현재 전주대학교 교양학부 강의전담전임강사
(관심분야 : 컴퓨터그래픽스, 사이언티픽 비주얼라이제이션, 멀티미디어)

최 성 희 (Seong-Hee Choi)

1982년 전북대학교 통계학과 학사
1984년 중앙대학교 전자계산학과 이학석사
1998년 전북대학교 전산통계학과 박사과정 수료
1989년~현재 전주공업대학교수
(관심분야 : 컴퓨터그래픽스, 멀티미디어)