깊이 버퍼를 이용한 레이캐스팅의 고속화

김승완[O†] 송주환[††] 권오봉[†]
† 전북대학교 전자정보공학부
†† 전주대학교 교양학부
kswsamson@mail.chonbuk.ac.kr[O], jwsong@jeonju.ac.kr, obgwun@moak.chonbuk.ac.kr

Acceleration Method for Ray Casting using Depth Buffer

Seungwan Kim[O†] Juwhan Song[††] Oubong Gwun[†]
† School of Electronics & Information Engineering Chonbuk National University
†† School of Liberal Arts Jeonju University

요 약

이 논문에서는 레이캐스팅을 고속화하는 단순하고 효율적인 알고리즘을 제안한다. 범용 PC에서 볼륨데이터를 이용하여 애니메이션을 하기 위해서는 초당 30 프레임의 영상을 생성하여야하나 아직 이에 도달하지 못하여 고속화가 필요하다. 지금까지의 바운딩서피스 기반의 레이캐스팅의 고속화에서는 임의의 시점에서 객체(object)의 깊이(depth)값을 그 객체의 바운딩서피스를 깊이 버퍼에 투영하여 구하였다. 이와는 다르게 이 논문에서 제안하는 방법은 시점과 무관하게 x, y, z 세 방향의 깊이 버퍼를 설치하고 이 것을 이용하여 임의의 방향에서 시점에 대한 물체의 깊이 값을 구한다. 이렇게 함으로서 임의의 시점에서 객체의 깊이 값을 구하는 시간을 N^3에서 $8N^2$으로 줄일 수 있다. 여기서 N은 차원당 복셀의 개수이다.

1. 서 론

레이캐스팅(ray casting)은 볼륨 데이터를 3차원으로 가시화하는 알고리즘이다. 볼륨데이터를 가시화하는 알고리즘은 이 이외에 플레인 컴포지팅(plane compositing), 텍스처 매핑(texture mapping)등의 방법이 있으나 레이트레이싱 방법이 다른 방법보다 견고하고 융통성이 있다. 예를 들어 레이트레이싱은 등고면(iso-surface)의 위치, 불투명도 및 칼라 값의 축적, 최대값 투영 등의 처리를 할 수 있다. 현재 볼륨프로(VolumePro) 보드는 253^3 해상도의 볼륨 데이터를 이용하여 253^2픽셀의 프레임을 30개 생성할 수 있는 능력이 있어 볼륨 렌더링의 속도 개선에 관한 연구의 중요성이 떨어진다고 생각할 수 있다. 그러나 1280×1024 해상도를 갖는 21인치 모니터 위에 객체를 가시화하기 위해서는 2400^3의 볼륨 데이터를 실시간으로 가시화할 수 있어야한다. 현재의 기술은 이러한 해상도로 볼륨데이터를 가시화할 수 있는 단계에 이르지 못하여 속도 개선이 아직도 필요하다. 그리고 범용 PC를 이용하여 볼륨데이터를 3 차원으로 가시화하기 위해서는 전용 보드를 이용하는 것보다는 소프트웨어를 이용하는 것이 더욱 바람직하다.

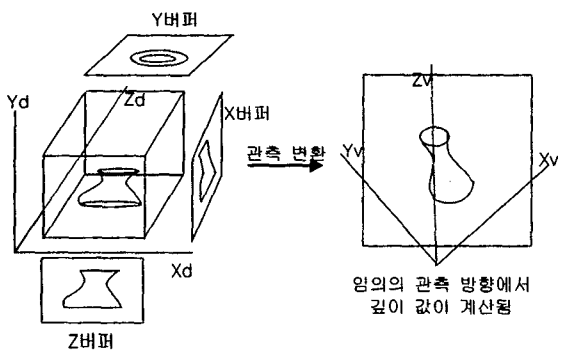
레이캐스팅을 고속화하는데 흔히 사용하는 방법은 연산레벨의 고속화[1], 복셀탐색의 고속화, 바운딩 서피스를 이용한 고속화로 나눌 수 있다. 연산레벨의 고속화는 레이캐스팅 프로그램을 분석하여 많이 발생하는 연산의 회수를 줄이거나 연산의 속도를 향상시키는 방법이다. 복셀탐색의 고속화는 광선의 진행 방향에 따라서 복셀값을 샘플링 할 때 이를 빠르게 탐색하는 방법으로 3DDDA(3 Dimensional Digital Differential Analyser), 템플릿기반볼륨렌더링(template based volume rendering), 팔진 트리(octree), 거리변환(distance transform) 등의 방법이 있다. 바운딩 서피스를 이용한 고속화는 물체를 바운딩 서피스(경계면)로 감싸고 이 경계면의 내부에 있는 복셀만을 처리함으로서 레이캐스팅의 속도를 향상시키는 방법이다.

바운딩 서피스를 이용하는 방법은 다시 영상 공간(image space)을 이용하는 방법과 물체 공간(object space)을 이용하는 방법 두 가지로 나눌 수 있는데 영상 공간을 이용하는 방법은 영상판(image plane)에 객체를 투영하여 투영된 객체와 만나는 픽셀에 대해서만 광선을 생성하여 처리하고 물체 공간을 이용하는 방법은 바운딩 서피스를 이용하여 볼륨 데이터를 계층적으로 분할한 다음에 계층적으로 바운딩 서피스와 복셀을 탐색하여 처리하는 방법이다. 바운딩 서피스를 이용한 레이캐스팅의 고속화 방법으로서는 PARC(Polygon Assisted Ray Casting)알고리즘[2], 경계셀기반고속화(boundary cell based acceleration)[3], 고성능프레젠스 레이캐스팅)[4] 등이 있다. PARC 알고리즘은 볼륨 공간 내부에 있는 객체를 육면체를 이용하여 둘러싸고 이 면들을 영상판에 투영하여 근거리 및 원거리 깊이 값을 구하여 이를 이용하여 레이캐스팅 하였다. 경계셀기반고속화는 전처리에서 객체의 바운딩 서피스를 구하고 이 서피스를

구성하는 셀의 테이블을 만든 다음에 본 처리에서 셀 테이블을 이용하여 바운딩 서피스만을 관측면에 투영하여 원거리 깊이 버퍼값과 근거리 깊이 버퍼값을 구하여 레이캐스팅 하였다. 고성능프레젠스 레이캐스팅에서는 바운딩 서피스를 구성하는 셀을 부호화하여 고속화하였다. 그러나 위의 바운딩 서피스를 이용한 모든 방법은 시점과 관측 방향이 변하지 않을 때에는 원거리 근거리 버퍼값을 다시 구할 필요가 없으나, 변할 때에는 다시 구하여야 하기 때문에 애니메이션 등의 처리에서는 고속화가 쉽지 않다. 이 논문에서는 볼륨 데이터의 6면의 방향에 대한 각각의 깊이를 미리 구해서 볼륨 데이터에 같이 저장한 후, 레이캐스팅 할 때 저장된 값을 이용하여 관측 방향에 대한 깊이를 구하여 렌더링 한다. 이때 6면에 해당하는 깊이에서 관측방향에서의 깊이를 구하는데는 아주 짧은 시간이 걸리게 되어, 관측방향이 바뀔 때에도 고속 처리가 가능한 알고리즘을 제안한다.

2. 깊이 버퍼를 이용한 레이캐스팅의 고속화

일반적으로 볼륨 데이터 내부는 대부분이 비어 있고 일부의 공간만이 데이터를 가지고 있다. 따라서 레이캐스팅을 할 때 데이터가 존재하는 복셀만을 탐색한다면 처리 시간을 많이 줄일 수 있다. 여기에서는 볼륨의 6면에 직교하는 방향을 관측 방향으로 가정하고 각각 방향에 대해 깊이를 미리 구해 버퍼에 저장하여, 그것을 이용하여 렌더링 하고자 하는 데이터를 가지고 있는 복셀을 효율적으로 탐색하는 방법에 대하여 고찰한다.



[그림 1] 깊이 버퍼의 개념

[그림 1]은 깊이 버퍼를 이용하여 임의의 방향에서 깊이 값을 구하는 개념도 이다. 데이터 볼륨 공간에서 X, Y, Z 축 각각에 대하여 2개씩, 전체 6개의 버퍼를 사용한다([그림 1]에서는 3개만 보임). X 버퍼, Y 버퍼, Z 버퍼는 각각 YZ 평면, XZ 평면, XY 평면에 2개씩 설치되며 X, Y, Z축의 원점으로부터 각각 객체 바운딩 서피스의 근거리 및 원거리 깊이 값을 갖는다. 각 버퍼의 해상도는 해당하는 면의 볼륨 데이터 해상도의 크기와 같다. 각축의 원점에서부터 해당하는 바운딩 서피스의 근거리 및 원거리 값은 전처리 단계에서 각 버퍼의 해상도에 맞추어 평행 광선을 쏘아 이 광선이 물체와 만나는 교점을 구하여 계산한다. 이때 주목할 점은 바운딩 서피스의 근거리 및 원거리 값은 시점에 따라 변하지 않기 때문에 한번만 구하면 된다. 광선이 처음 물체와 만나는 곳이 근거리 깊이 값이 되고 떠나는 곳이 원거리 깊이 값이 된다.

위의 X, Y, Z의 세 방향의 깊이 버퍼를 이용하여 다음과 같이 고속 레이트레이싱 처리 계산을 한다.

① 데이터 볼륨 데이터 좌표계에 세 방향 깊이 버퍼를 설치하여 바운딩 서피스의 근거리 및 원거리 깊이 값을 구한다.
② 시점과 관측 방향이 변하면 ①에서 구한 세 방향 버퍼 값을 관측 변환하여 임의의 관측 방향에 대한 깊이 버퍼 값을 구한다.
③ 임의의 관측 방향에 대하여 ②에서 구한 근거리, 원거리 깊이 버퍼 값을 이용하여 근거리 깊이부터 시작하여 원거리 깊이까지만 복셀을 탐색하여 레이캐스팅 한다.

데이터 볼륨 좌표계를 기준으로 한 X, Y, Z 버퍼의 원거리, 근거리 값을 새로운 관측 방향에 대하여 계산하는 식은 다음과 같다. 여기서 $n_{x-1}, n_{y-1}, n_{z-1}$는 각각 x, y, z 방향으로 볼륨 데이터의 해상도이고 x_d, y_d, z_d는 데이터 볼륨 공간에서 한 점의 좌표이고 x_v, y_v, z_v는 새로운 관측 방향 즉 관측 변환 후의 좌표이다. X 방향에 대한 변환은 다음과 같다.

$$\text{for}(y_d=0;\ y_d<n_y-1;\ y_d++)$$
$$\quad \text{for}(z_d=0;\ z_d<n_z-1;\ z_d++)$$
$$\quad\quad x_v, y_v, z_v = \text{view\_transform}(\text{x\_buffer}(y_d, z_d), y_d, z_d);$$

Y 방향 및 Z 방향에 대해서도 동일한 처리를 한다. 관측 변환(view_transform)은 데이터 볼륨 공간 중앙에 관측 좌표계의 중심을 설정하고 y 축을 중심으로 α도 회전, x 축을 중심으로 β도 회전, z 축을 중심으로 θ도 회전으로 이루어진다. 볼륨 데이터 공간에서 한 점의 좌표값을 관측 좌표계의 한 점으로 변환하는 식은 다음과 같다[5].

관측 방향이 바뀔 때마다 볼륨 데이터의 6개의 깊이를 위 식에 적용하여 근거리 깊이와 원거리 깊이를 새로 구하여 레이캐스팅 함으로써 임의의 방향에 대해서 렌더링이 가능하다.

3. 구현 및 평가

이 논문에서 제안한 기법의 타당성을 Pentium III 933MHz CPU와 512MB를 설치한 PC 환경에서 Visual C++를 이용하여 구현하여 검증하였다. 제안된 기법이 속도 향상에 어느 정도 효과가 있는지를 알아보기 위하여 고속화 기법을 사용하지 않은 레이캐스팅, PARC 알고리즘을 이용한 레이캐스팅을 구현하였다. 벤치마크 데이터는 볼륨 렌더링 기법을 평가하는데 많이 사용하는 Visible Human의 머리부분 CT 영상과 노스캐롤라이너 대학에서 구한 엔진의 CT 영상을 사용하였다.

[표 1]은 각 방법의 처리 시간 및 속도 향상율을 보인다. 제안한 알고리즘을 이용한 렌더링이 렌더링 시간은 PARC 알고리즘에 비해서 약간 떨어지나, 방향을 바꾸었을 때도 제안한 방법은 똑같은 시간이 걸리게 되어 임의의 방향에 대해서도 실시간으로 렌더링 할 수 있음을 보인다. [그림 2]에서는 머리의 CT 영상에서 표피부분과 두개골 부분을 각각 렌더링 한 결과와 엔진을 렌더링한 결과를 나타내었다. 세 가지 방법 모두의 결과의 영상이 차이가 거의 없음을 알 수 있다.

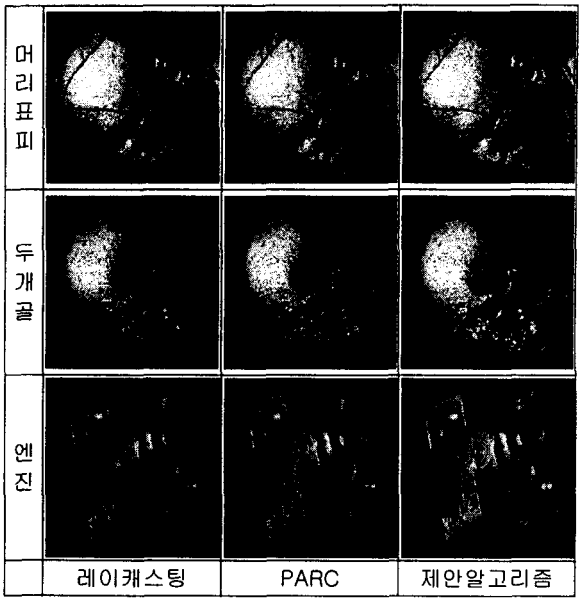
벤치마크 데이터		레이캐스팅		PARC		제안알고리즘	
		시간	향상율	시간	향상율	시간	향상율
머리 표피	전처리	없음	-	1.803	-	0.090	-
	렌더링	2.613	1	0.671	3.89	0.871	3.00
	전체	2.613	1	2.474	1.05	0.961	2.71
두개골	전처리	없음	-	2.854	-	0.061	-
	렌더링	3.535	1	0.471	7.50	0.601	5.88
	전체	3.535	1	3.325	1.06	0.662	5.34
엔진	전처리	없음	-	2.944	-	0.040	-
	렌더링	3.365	1	0.341	9.86	0.390	8.62
	전체	3.365	1	3.285	1.02	0.430	7.82

[표 2] 처리 시간 및 속도 향상율

4. 결론

PARC 알고리즘은 관측 방향에 대해서 볼륨 내부에서 경계에 해당하는 셀의 깊이를 전처리 단계에서 계산한 후 렌더링하기 때문에 처리 시간이 매우 빠른 반면, 관측방향이 바뀌면 새로 깊이를 구해야 하기 때문에 많은 시간이 소요되는 전처리 과정을 다시 하게 되는 문제점이 있다. 본 논문에서 제안한 방법은 육면체 형태로 주어지는 볼륨 데이터에서 6면의 직교 방향이 관측방향이라 가정하고 각각의 깊이를 미리 구해서 볼륨 데이터에 저장해 둔다. 렌더링 시에는 6면의 깊이를 이용해서 관측방향에 대한 깊이를 구하여 사용하는데, 이 시간이 0.1초 이하의 시간이 걸리게 되어 실시간으로 방향을 바꾸면서 렌더링 할 수 있다.

6면의 깊이를 이용해서 깊이를 구하는데 일부에서는 깊이를 구하지 못하여 구멍이 생기는 문제점이 있어 향후 이 문제점을 없애는 연구를 할 계획이다.



[그림 2] 제안한 방법으로 생성한 영상

5. 참고 문헌

[1] J. Song, O. Gwun and H. Jeong, "Operation Level Acceleration for Volume Rendering," Visualization and Data Analysi 2002, 154-164.

[2] L. M. Sobierajski and R. S. Avila, "A Hardware Acceleration Method for Volume Ray Tracing," IEEE Visualization 1995; 95: 27-34.

[3] M. Wan, S. Bryson and A. Kaufman, "Boundary Cell-based Acceleration for Volume Ray Casting," Comput. & Graphics, Vol. 22, No. 6, pp. 715-721, 1998.

[4] M. Wan, A. Kaufman and S. Bryson, "High Performance Presence-Accelerated Ray Casting," IEEE visualization 1999; 99: 379-86.

[5] S. Yang and T. Wu, "Improved Ray-casting Algorithm for volume visualization," Visualization and Data Analysi 2002, 319-326.