정점분할을 이용한 GPU 기반 볼륨 렌더링의 가속 기법※

유성열, 이은석, 신병석
인하대학교 컴퓨터 · 정보공학과
bleularme@hotmail.com, elflee77@nate.com, bsshin@inha.ac.kr

Acceleration of GPU-based Volume Rendering Using Vertex Splitting

Seong-Yeol Yoo, Eun-Seok Lee, Byeong-Seok Shin
Dept. of Computer Science & Information Engineering, Inha University

요 약

볼륨 광선 투사법은 볼륨 데이터를 가시화하는 기법 중 고화질 영상을 만들어내는 기법이다. 하지만 일반적으로 볼륨 데이터는 매우 크기 때문에 렌더링 시간이 오래 걸리는 문제가 있다. 이를 보완하기 위하여 최근에는 GPU를 이용하여 볼륨 광선 투사법을 가속화하는 많은 기법들이 연구되고 있다. 본 논문에서는 볼륨 광선 투사법을 가속화하기 위한 GPU 기반의 옥트리 탐색을 통한 효과적인 빈 공간 도약 기법을 제안한다. 여기서는 최대-최소 옥트리를 생성하고 옥트리의 루트 노드부터 정점분할을 이용하여 빈 공간을 식별한다. 찾아낸 빈 공간을 삭제함으로써 볼륨 데이터에서 의미 있는 객체를 둘러싸는 바운딩 다면체를 최소화 시킨다. 최소화 된 바운딩 다면체에 대해서만 렌더링을 진행함으로써 기존의 볼륨 광선 투사법과 비교하여 빠른 시간에 동일한 결과물을 생성한다.

키워드 : 볼륨 렌더링, 광선 투사법, 옥트리 탐색, 빈 공간 도약, 정점분할

ABSTRACT

Visualizing a volume dataset with ray-casting which of visualization methods provides high quality image. However it spends too much time for rendering because the size of volume data are huge. Recently, various researches have been proposed to accelerate GPU-based volume rendering to solve these problems. In this paper, we propose an efficient GPU-based empty space skipping to accelerate volume ray-casting using octree traversal. This method creates min-max octree and searches empty space using vertex splitting. It minimizes the bounding polyhedron by eliminating empty space found in the octree traveral step. The rendering results of our method are identical to those of previous GPU-based volume ray-casting, with the advantage of faster run-time because of using minimized bounding polyhedron.

Keywords : volume rendering, ray-casting, octree traversal, empty space skipping, vertex splitting

접수일자 : 2012년 01월 26일 심사완료 : 2012년 02월 21일
교신저자(Corresponding Author) : 신병석(Byeong-Seok Shin) E-mail : bsshin@inha.ac.kr
※ 이 논문은 2011년도 정부(교육과학기술부)의 재원으로 한국 연구재단의 지원을 받아 수행된 연구임(No. 2011-0015779).

1. 서 론

볼륨 렌더링(volume rendering)[1,2]은 컴퓨터 그래픽스의 한 분야로, 볼륨 데이터를 가시화하여 시각적인 분석을 효과적으로 할 수 있게 하는 도구이다. 볼륨 데이터를 가시화하기 위한 직접 볼륨 렌더링(direct volume rendering) 기법들 중 볼륨 광선 투사법(volume ray-casting)[3]이 많이 사용된다. 볼륨 광선 투사법은 높은 화질의 영상을 제공하지만, 수행 속도가 느린 단점이 있다.

최근 연산 능력이 매우 높아진 GPU(graphic processing unit)는 병렬 처리에 최적화 되어 있으며, 부동 소수점 연산에 특화되어 있기 때문에 그래픽 연산에 적합하다. 영상의 각 픽셀의 색상 값을 계산하기 위해 CPU(central processing unit) 기반의 볼륨 광선 투사법은 GPU의 프래그먼트 쉐이더(fragment shader)에서 각각의 광선을 동시에 투사함으로써 CPU 기반의 볼륨 광선 투사법에 비해 매우 빠른 렌더링이 가능하다. 그러나 일반적으로 볼륨 데이터는 매우 크기 때문에 렌더링을 수행하는데 많은 시간이 필요하며, 이러한 문제를 해결하기 위해 다양한 가속화 기법들이 연구되고 있다. 볼륨 광선 투사법의 가속화 기법에는 광선이 진행하며 불투명도 값을 누적할 때 그 값에 한계를 두어 완전히 불투명해 지기 전에 광선의 진행을 멈추는 조기 광선 종료(early ray termination) 기법과 볼륨 데이터 내부의 빈 공간을 도약하여 렌더링을 가속화 시키는 빈 공간 도약 기법(empty space skipping)이 있다. 하지만 그래픽 하드웨어의 렌더링 파이프라인에서는 정점 배열이나 텍스처 등 순차적인 데이터만 처리가 가능하기 때문에 옥트리(octree)와 같은 계층적 자료구조는 사용할 수 없다. 따라서 옥트리를 이용하는 기법들은 CPU에서만 수행될 수 있고 이로 인해 속도가 저하되는 문제가 있다.

본 논문에서는 기존의 CPU에서 수행되는 옥트리 처리를 3D 텍스처 피라미드와 정점분할 기법[4]을 이용하여 GPU에서 구현하여 수행속도를 획기적으로 개선하였다. 이 기법은 그래픽 하드웨어의 렌더링 파이프라인에서 계층적 자료구조인 옥트리의 생성과 탐색을 가능하도록 함으로써 GPU만을 이용한 옥트리 기반의 볼륨 렌더링을 수행하는 기법으로 기존의 CPU에서 수행되는 옥트리 이용 기법들에 비해 매우 빠른 렌더링이 가능하다.

2장에서는 GPU 기반의 볼륨 광선 투사법에 대한 간단한 설명과 이를 가속화하는 기법들에 대하여 소개하고, 3장에서는 정점분할 기법을 이용하여 옥트리 탐색을 하는 GPU 기반의 볼륨 렌더링 기법을 기술한다. 4장에서는 기존의 기법과 비교한 결과를 보이며, 5장에서 결론을 맺는다.

2. 관련연구

볼륨 렌더링은 기하학적 모델을 기반으로 하는 기존의 렌더링 기법과는 달리 반투명한 물체의 내부를 표현하는 것이 용이하다. 볼륨 광선 투사법은 볼륨 렌더링 기법 중 가장 높은 화질의 영상을 제공하는 기법이지만, 결과 영상의 각 픽셀 색상을 결정하기 위해 뷰평면의 모든 화소에서 가상의 광선을 발사하여 광선의 경로 상에 위치한 각 샘플점으로부터 색상을 계산하고 이를 축적한다. 이는 높은 화질의 결과 영상을 생성 할 수 있지만, 광선의 방향에 따라 임의의 순서로 복셀을 참조해야 하기 때문에 수행 속도가 느리다.

Krüger는 볼륨 데이터를 2차원 텍스처로 저장하고 프록시 기하도형(proxy geometry)을 이용하여 렌더링 하던 기존의 방식과는 전혀 다른 GPU 기반의 볼륨 광선 투사법을 제안하였다[5]. 이는 렌더링 속도를 획기적으로 향상시켰을 뿐만 아니라 조기 광선 종료 기법을 GPU에서 효율적으로 사용할 수 있게 되었다.

옥트리는 구조가 간단하며 계층화된 구조를 가지고 있기 때문에 볼륨 광선 투사법을 가속하는데 많이 사용된다. 상위 노드에서 발견된 빈 공간은 하위 노드의 검색 없이 도약할 수 있기 때문이다.

Wilhelms와 Gelder는 최대-최소 옥트리(min-max octree)를 이용한 가속 기법을 제안하였다[6]. 이것은 샘플링할 때, 샘플 값을 최대-최소값과 비교하여 그 구간에 없으면 해당 블록을 투명하다고 간주하고 블록의 경계까지 도약한다. 하지만 유효하지 않은 블록의 경계까지 한번에 도약하는 것이 힘들고, 옥트리의 레벨이 증가함에 따라 블록간의 비교연산과 레벨 이동이 빈번하게 발생한다. Hong은 전체 볼륨 데이터를 브릭(brick)으로 나눈 후, 시점에 따라 최대-최소 옥트리를 생성하고 브릭을 정렬하는 기법을 제안하였다[7]. 그러나 이 기법은 시점이 변화하면 브릭들을 다시 생성해야 하는 시점 종속적인 기법이라는 단점이 있다.

빈 공간을 도약하기 위해 kd-트리를 이용하여 가속화 하는 기법[8]이 제안되었으나 전처리 과정의 오버헤드 때문에 실시간 처리가 불가능하며, 불투명도 전이 함수(opacity transfer function)가 변하는 경우에 유연하게 대응하지 못하는 문제가 있다. Sramek은 전처리 과정에서 각각의 복셀과 가장 가까운 유효 복셀까지의 거리를 저장시켜 놓은 거리맵 기반의 빈 공간 도약 기법을 제안했다[9]. 거리맵에는 가장 가까운 유효 복셀까지의 거리가 저장되기 때문에 빈 공간을 탐색하는 과정에서 유효한 복셀까지 빠르게 도약 할 수 있도록 한다. 하지만, 특정 광선은 유효 복셀에 도약할 때까지 여러 번 거리맵을 참조 해야 하는 문제가 있다.

Mensmann은 기하쉐이더(geometry shader)를 사용하여 빈 공간 도약을 가속화하는 기법을 제안하였다[10]. 광선 진입점의 깊이 값을 이용하여 기하쉐이더에서 차폐 절두체(occlusion frustum)라는 기하구조를 생성한다. 이 기법은 광선 진입점의 깊이 값만을 사용하여 효율적으로 빈 공간을 제거할 수 있지만, 불투명도 전이 함수가 변하면 광선 진입점도 변하기 때문에 이것을 다시 계산해야 하고 차폐 절두체도 재생성 해야 하는 문제가 있다. Liu는 임의의 구형 기하 구조체를 사용한 가속 기법을 제안하였다[11].

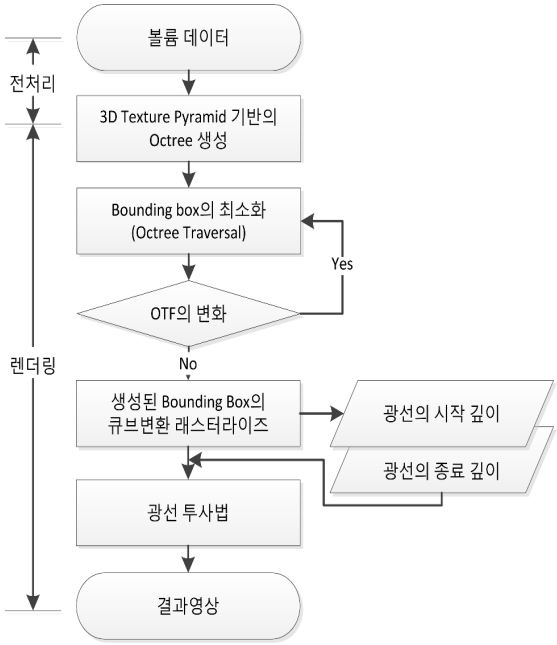
본 논문에서는 옥트리 탐색시 비교연산과 옥트리의 레벨 이동이 빈번하게 발생하여 수행시간이 큰 Wilhelms와 Gelder의 기법을 정점분할을 이용함으로써 비교연산과 레벨 이동을 단순화하여 수행시간을 단축하였다. 또한, 3D 텍스처 피라미드를 이용하여 옥트리 생성을 함으로써 불투명도 전이 함수의 변화에 따라 차폐 절두체를 재생성하여야 하는 Mensmann의 기법에서 발생하는 문제점을 해결하였다.

제안하는 기법은 3D 텍스처 피라미드(texture pyramid)를 이용하여 최대-최소 옥트리를 생성한 후, 정점분할 기법을 이용하여 생성된 옥트리를 탐색하여 가시화할 객체가 포함된 유효 공간을 식별한다. 이를 통해 유효하지 않은 빈 공간을 제거하며 객체를 둘러싸는 바운딩 다면체를 최소화 한다. 이 방법은 시점에 종속적이지 않으며, 바운딩 다면체의 최소화를 위한 옥트리의 탐색 레벨을 사용자가 임의로 조절할 수 있다. 기존의 기법들과 달리 불투명도 전이 함수가 변해도 옥트리의 재생성 과정이 필요하지 않으며, 정점분할 기법을 통해 적은 회수의 정점 비교만으로 옥트리의 탐색이 가능하여 빈 공간을 도약하고 유효 복셀을 탐색하는데 많은 옥트리 비교연산이 필요하지 않다. 또한, 볼륨 데이터의 모양이나 구조에 관계없이 최적화 된 바운딩 다면체를 생성 할 수 있는 장점이 있다.

3. 본 론

제안하는 기법의 모든 과정은 GPU에서 수행되며, [그림 1]과 같이 전처리 단계와 렌더링 단계로 구성된다. 전처리 단계에서는 3D 텍스처 피라미드를 이용하여 최대-최소 옥트리를 생성한다. 렌더링 단계에서는 정점분할 기법을 통하여 GPU에서 옥트리를 탐색하며 유효하지 않은 정점을 삭제하고, 탐색된 유효 정점들을 기하쉐이더에서 다면체로 변환하는 과정을 거치며, 객체를 둘러싸는 바운딩 다면체를 최소화 하게 된다. 볼륨 데이터는 통상 육면체 형태를 가지므로 초기 바운딩 다면체도 육면

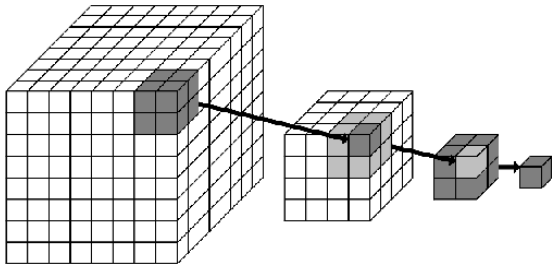
체이다. 옥트리를 이용하여 빈공간 탐색을 하면서 그 모양이 점점 객체의 형태에 가까운 복잡한 다면체가 된다. 래스터화 과정을 거치며 시점을 기준으로 최소화 된 바운딩 다면체의 시작 깊이 값과 종료 깊이 값이 저장된 텍스처가 생성된다. 이 깊이 값을 이용하여 광선 투사 단계에서는 빈 공간을 도약함으로써 유효한 영역에 대해서만 기존의 광선 투사법을 적용하여 렌더링 한다.



[그림 1] 본 논문에서 제안하는 기법의 처리절차

3.1 3D 텍스처 피라미드

옥트리는 3차원 공간을 재귀적으로 분할하여 표현하는 계층적 자료구조이다. 하지만 GPU의 렌더링 파이프 라인에서는 텍스처나 정점 배열과 같은 순차적인 데이터만 처리가 가능하며, 메모리의 특정 위치에 저장된 데이터에 직접 접근해야 하는 트리와 같은 계층적 자료구조는 사용할 수 없다. 3D 텍스처 피라미드는 이를 해결하기 위한 자료구조로서 GPU에서 볼륨 데이터를 옥트리와 같이 계층적인 구조로 표현할 수 있다. 볼륨 데이터 각각의 복셀들을 8개씩 묶어 해당 범위의 최대 밀도 값과 최소 밀도 값이 하나의 복셀로 표현될 때까지 반복한다. [그림 2]는 3D 텍스처 피라미드 생성 과정을 보여주는 예로서 83 크기를 가지는 볼륨 데이터 전체 영역이 하나의 복셀로 표현되는 과정을 보여주고 있다. 마지막의 한 복셀로 된 3D 텍스처는 옥트리의 루트 노드가 되며, 상위 레벨 텍스처의 각 복셀은 하위 레벨 3D 텍스처의 8개 복셀의 최대 밀도 값과 최소 밀도 값을 저장한다.



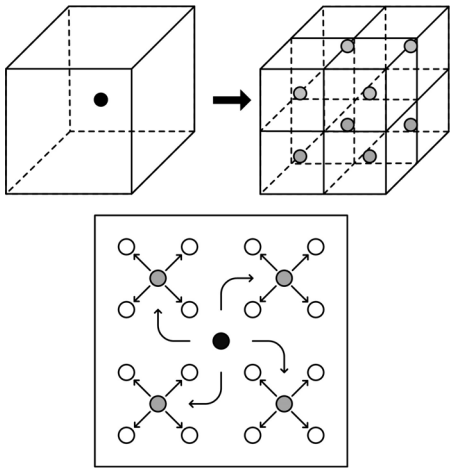
[그림 2] 3D 텍스처 피라미드 생성 과정. 각 3D 텍스처 복셀에 하위 레벨 3D 텍스처의 8개 복셀의 최대/최소 밀도 값을 저장한다.

3.2 정점분할을 이용한 옥트리 탐색

정점분할은 기하쉐이더에서 수행되며, 옥트리 탐색을 GPU에서 할 수 있게 하여 GPU 기반의 볼륨 광선 투사법을 가속 할 수 있다.

정점분할의 시작은 옥트리의 루트 노드에 해당하는 [그림 3]의 검은색 정점으로부터 시작된다. 이 정점은 [그림 2]에서 마지막으로 생성되는 볼륨 전체 영역의 최대-최소 밀도 값이 저장된 3D 텍스처에 해당하며, 3D 텍스처 볼륨의 중앙에 위치한다. 정점분할은 [그림 3]과 같이 루트 노드부터 트리를 탐색하며, 하위 노드의 탐색이 필요한 경우 루트 정점은 8개의 회색 정점으로 분할된다. 회색 정점들은 검은색 정점을 기준으로 분할된 8개 영역의 중앙에 위치한다. 이렇게 생성된 정점들은 스트림 아웃(stream out)을 통해 메모리로 저장되고 다시 렌더링 파이프라인으로 입력이 된다. 재입력된 회색 정점들은 위의 탐색 과정을 거쳐 추가 탐

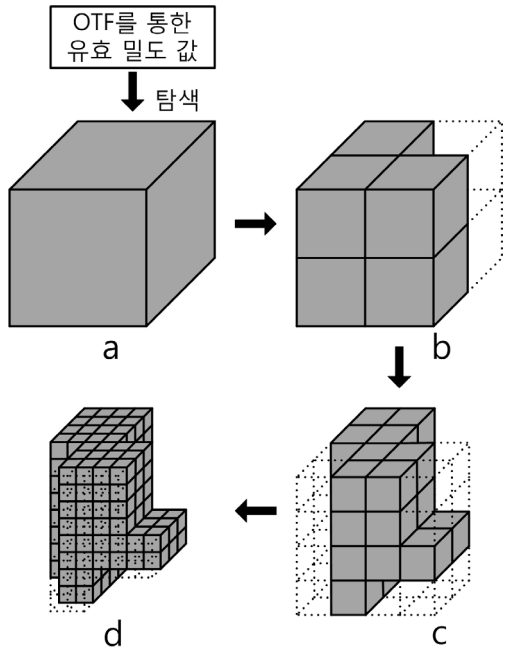
색이 필요한 경우 흰색 정점들로 분할된다. 유효하지 않은 영역으로 판별 될 경우 해당 정점을 삭제하여 하위 노드로 분할하지 않는다. 이 과정은 사용자가 임의로 정한 레벨까지 반복된다.



[그림 3] 정점분할 기법에서 하나의 정점이 8개로 분할되는 모습을 3차원 모형과 2차원 단면으로 표현한 그림. 검은색 정점이 루트 노드이고 회색은 그 자식 노드이다. 흰색 노드는 회색의 자식 노드이다.

3.3 바운딩 다면체의 최소화

불투명도 전이 함수가 정해지면 정점분할을 통해 옥트리의 루트 노드부터 복셀 값을 탐색하여 바운딩 다면체를 최소화 한다. 옥트리 탐색을 위해서 쉐이더 5.0 모델의 기하쉐이더를 이용하여 재귀적인 연산을 수행하여 병렬로 트리를 탐색한다. 이는 상위 노드에서 발견된 유효하지 않은 공간은 하위 노드에서도 그대로 적용되는 최대-최소 옥트리의 계층 구조를 이용한 것이다. 사용자가 임의로 지정한 옥트리 레벨까지 탐색을 수행하게 되며, 원본 볼륨 데이터의 바운딩 다면체(육면체)에서 유효하지 않은 공간은 삭제되어 최소화 된 바운딩 다면체를 생성한다. [그림 4]는 정점분할 기법을 이용하여 옥트리 탐색을 하였을 때 바운딩 다면체가 최소화 되는 과정을 보여주는 예이다. 각 노드가 포함하는 영역의 바운딩 다면체를 생성하고 트리를 탐색할 때마다 그림과 같이 8등분하여 탐색을 한다. 또한 바운딩 다면체를 분할하는 과정에서 빈 공간에 해당하는 노드는 삭제하여 바운딩 다면체를 최소화 하며, 그 과정에서 불필요한 하위 노드의 탐색 연산을 줄인다.

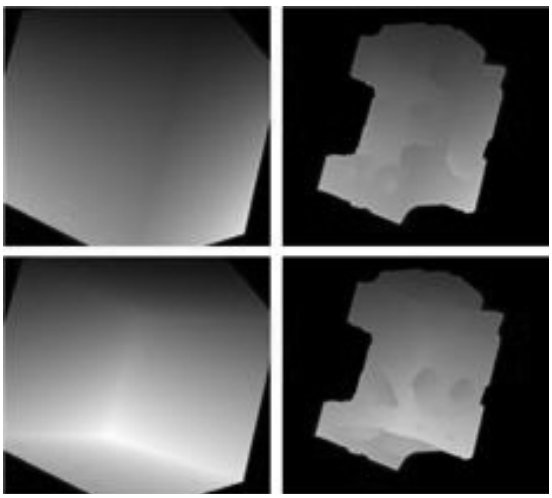


[그림 4] 옥트리 탐색을 통한 바운딩 다면체의 최소화 과정. a는 기존 볼륨 광선 투사법의 바운딩 다면체. d는 제안하는 기법의 최소화된 바운딩 다면체

3.4 바운딩 다면체의 래스터화

앞 단계에서 얻어낸 바운딩 다면체는 기하쉐이더를 거치면서 원본 볼륨 데이터에서 불투명도 전이 함수에 의해 유효한 데이터 영역만을 포함하는 최소화된 바운딩 다면체가 된다. 마지막으로 생성된 바운딩 다면체는 스트림 아웃 하지 않고 바로 깊이 테스트를 거쳐 프래그먼트 쉐이더에서 래스터화 된다. GPU 기반의 볼륨 광선 투사법을 수행하기 위해서는 광선의 시작 깊이 값과 종료 깊이 값에 대한 정보가 필요하며, 이 정보는 GPU의 래스터화 단계에서 뷰평면으로부터 바운딩 다면체의 가장 가까운 면의 깊이 값과 가장 멀리 있는 면의

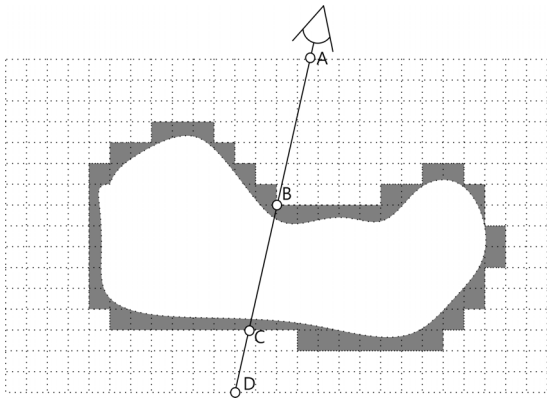
깊이 값을 통해 구할 수 있다. [그림 5]는 기존의 기법과 제안하는 기법으로 생성되는 광선 진입점의 깊이 값과 종료점의 깊이 값으로 얻은 볼륨 텍스처 좌표를 RGB 색상 값으로 나타낸 것이다. 이를 통해 바운딩 다면체가 최소화되었음을 시각적으로 확인할 수 있다.



[그림 5] 기존의 GPU 광선 투사법(왼쪽)과 제안하는 기법(오른쪽)에서 광선 진입점 및 종료점 깊이 값의 가시화 결과

3.5 광선 투사를 통한 픽셀 색상 결정

기존의 GPU 기반 광선 투사법은 [그림 5]의 좌측 이미지와 같이 볼륨 전체 영역을 포함하는 바운딩 다면체를 사용하여 광선을 투사하고 샘플링을 수행한다. 제안하는 기법에서는 [그림 5]의 우측 이미지와 같이 빈 공간이 제거된 상태의 바운딩 다각형에서 광선 진입점 깊이 값과 종료점 깊이 값을 이용하여 광선 투사를 진행한다. [그림 6]은 두 기법에서 광선이 진행하며 바운딩 다면체 내부를 샘플링 하는 구간을 보여준다. 기존 기법에서 하나의 광선은 AD의 모든 경로에서 샘플링을 수행하기 때문에 빈 공간에 대한 불필요한 샘플링도 수행하게 된다. 제안하는 기법에서 광선은 최소화된 바운딩 다면체의 광선 진입점과 종료점 사이의 경로인 BC에서만 샘플링을 수행한다. 조기 광선 종료 알고리즘을 적용하여 기존 기법에서 C와 D 사이의 샘플링이 이루어지지 않고 종료되어도 제안하는 기법에서는 빈 공간인 AB를 샘플링하지 않기 때문에 전반적인 속도는 향상된다.



[그림 6] 기존 볼륨 광선 투사법과 제안하는 기법의 광선 경로. 기존의 기법은 바운딩 다면체 전체 영역인 AD를 샘플링하며, 제안하는 기법에서는 최소화 되어진 바운딩 다면체의 영역인 BC에서 샘플링 한다.

4. 실험 결과

실험은 3.3GHz의 Intel® Core i5 2500 CPU와 DDR3 8GB의 주 메모리를 장착한 시스템에서 수행하였으며, 그래픽 장치는 DDR5 1GB의 비디오 메모리를 가지는 Geforce GTX560 Ti를 사용하였다. 결과영상의 해상도는 1024×1024 이며, DirectX 11 라이브러리를 사용하였다. 실험에 사용된 데이터 셋은 5123의 크기를 가지는 Engine과 Foot데이터이며, 제안하는 방법과 기존 광선 투사법에 모두 조기 광선 종료 알고리즘을 적용하였다.

[표 1]과 [표 2]는 두가지 실험데이터에 기존 기법과 제안한 기법을 적용하였을 때 속도차이를 비교한 것이다. 불투명도 전이 함수의 변화에 따른 렌더링 속도의 차이를 확인하기 위해서 5가지의 함수를 사용하였다. 볼륨 데이터에서 유효한 밀도 값의 범위를 0~255, 50~255, 100~255, 150~255,

200~255의 5가지 구간으로 설정하여 그 구간 밖의 밀도 값은 투명한 빈 공간으로 간주하도록 하였다. 이는 유효 범위가 줄어들었을 때, 옥트리 탐색을 통해 바운딩 다면체의 최소화가 더 많이 되었음을 확인하기 위해서이다.

옥트리는 루트 노드부터 9레벨로 나누어 실험하였다. 루트 노드에는 3D 텍스쳐가 한 복셀로 이루어져 5123의 크기의 원본 볼륨 전체를 저장하며 최하위 노드에는 한 복셀이 23의 크기의 정보를 저장한다. 실험에서는 해당 레벨까지 옥트리 탐색을 수행하였을 때의 렌더링 속도를 확인하였다.

제안한 기법에서 옥트리를 한 레벨만 탐색한 경우는 바운딩 다면체가 볼륨 데이터 전체 영역이 되므로 기존의 광선 투사법과 동일한 결과를 보였다. 옥트리의 탐색 레벨이 높아짐에 따라 바운딩 다면체는 가시화 할 객체에 밀착되기 때문에 렌더링 속도는 점점 더 향상되는 것을 확인할 수 있었다. Engine 볼륨 데이터에서 유효 밀도의 범위가 0~255인 경우, 하나의 옥트리 블럭이 163에 해당되는 레벨에서 279fps의 처리속도가 나온다. 동일한 조건에서 기존 광선 투사법의 32fps과 비교하여 속도가 최대 8.71배 향상된 것이다. Foot 볼륨 데이터는 하나의 옥트리 블록이 83에 해당되는 레벨까지 탐색하였을 때 177fps의 속도를 보여 기존 기법보다 최대 5.20배 빨라졌다.

유효 밀도 값이 변경되어도 볼륨 데이터 전체 범위를 바운딩 다면체로 사용하는 기존의 광선 투사법과는 달리, 제안하는 기법에서는 유효 밀도 값이 변경되면 옥트리 탐색을 통해 바운딩 다면체의 모양과 크기가 달라진다. 실험에서 유효 밀도 값의 범위를 200~255로 하면, 유효한 영역은 전체 데이터에 비해 매우 적어져서 속도 향상효과가 극대화 된다. 기존 방법의 속도는 23~24fps지만 제안한 방법에서는 최적의 옥트리 레벨에서 각각 494fps, 396fps의 결과를 얻어 최대 21배의 렌더링 속도 향상이 있음을 확인하였다.

기존 볼륨 광선 투사법은 볼륨 전체 영역을 바운딩 다면체로 사용하기 때문에 유효 밀도 값의 범위가 변하더라도 렌더링 속도가 크게 바뀌지 않지만, 제안한 기법은 유효 밀도 값의 범위를 줄이고 옥트리 탐색 레벨을 높이면 바운딩 다면체가 가시화할 객체에 밀착되면서 속도가 향상된다.

옥트리 탐색 레벨을 증가하면 처리속도는 점차 향상되지만 특정 레벨 이상이 되면 오히려 처리속도가 저하된다. 이것은 옥트리 탐색 레벨이 높아지면 유효한 공간을 탐색하여 만들어진 바운딩 다면체를 구성하는 다각형의 개수가 증가하여 이들을 래스터화 하는 과정에서 많은 시간이 소요되기 때문이다. 실험에 의하면 평균적으로 하나의 옥트리 블럭이 83에 해당되는 레벨까지 탐색할 때 최적의 성능을 보이는 것으로 확인되었다.

본 논문에서 제안한 기법은 기존의 볼륨 광선 투사법과 비교하여 샘플링 하는 범위만 달라질 뿐 실제 샘플링 되는 데이터는 동일하므로 결과 영상도 기존 기법과 다르지 않다. [그림 7]은 기존 기법과 제안하는 기법에서 유효 범위 값을 변경하면 렌더링 한 결과 영상을 비교한 것이다.

5. 결 론

본 논문에서는 GPU기반 볼륨 광선 투사법의 속도를 향상시키기 위해 정점분할을 이용한 효율적인 옥트리 탐색 기법을 제안하였다. 모든 과정은 GPU상에서 이루어지며, 정점분할을 이용한 옥트리의 탐색은 병렬로 처리되기 때문에 매우 빠르게바운딩 다면체를 최소화한다. 기존의 볼륨 광선 투사법과 비교하여 동일한 화질의 결과물을 생성하며 최소화된 바운딩 다면체를 렌더링하기 때문에 최적의 옥트리 탐색 레벨에서 최소 5배, 최대 21배의 가속효과가 있었다. 또한, 유효 밀도 값의 범위가 변하더라도 옥트리의 재생성 과정이 필요하지 않으며, 재탐색 과정만으로 바운딩 다면체를 최적화 할 수 있다. 제안하는 기법은 특정 밀도의 영역만을 포함하는 바운딩 다면체를 생성할 수 있기 때문에 의료 영상 분야에서 밀도가 다른 영역들을 렌더링을

하는데 활용할 수 있다.

최적의 옥트리 탐색 레벨을 지나면 속도가 오히려 느려지는 문제가 발생하였다. 향후에는 이를 개선하여 더욱 효과적인 옥트리 탐색을 수행하는 기법을 연구할 것이다.

[표 1] 제안한 기법에서 옥트리 탐색 레벨에 따른 Engine 볼륨 데이터의 렌더링 속도 (fps)

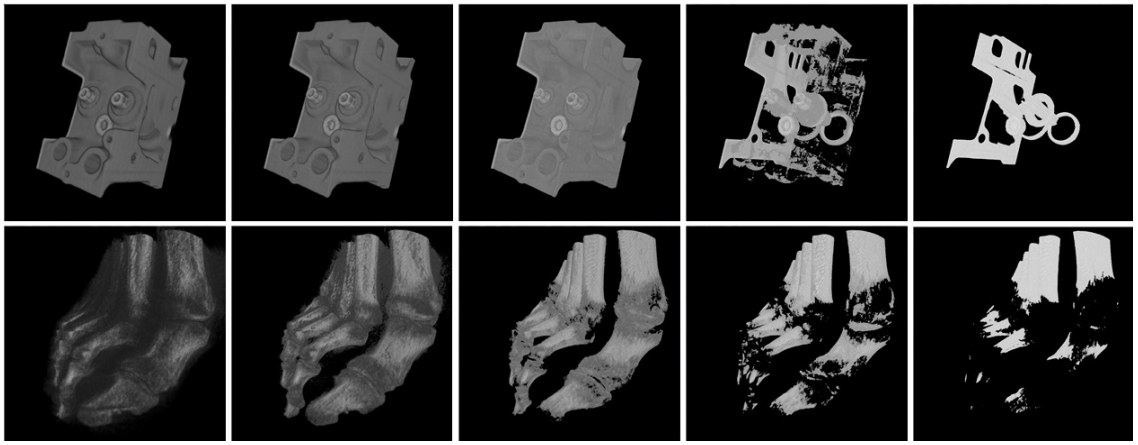
옥트리 레벨	유효 밀도 값의 범위														
	0~255			50~255			100~255			150~255			200~255		
	A	B	B/A (%)	A	B	B/A (%)	A	B	B/A (%)	A	B	B/A (%)	A	B	B/A (%)
5123	32	32	1.00	32	32	1.00	32	32	1.00	24	24	1.00	24	24	1.00
2563	-	32	1.00	-	32	1.00	-	31	0.96	-	25	1.04	-	24	1.00
1283	-	94	2.93	-	94	2.93	-	94	2.93	-	63	2.62	-	79	3.29
643	-	129	4.03	-	141	4.40	-	153	4.78	-	81	3.37	-	162	6.75
323	-	210	6.56	-	210	6.56	-	209	6.53	-	97	4.04	-	285	11.87
163	-	279	8.71	-	279	8.71	-	279	8.71	-	132	5.50	-	439	18.29
83	-	210	6.56	-	211	6.59	-	209	6.53	-	145	6.04	-	494	20.58
43	-	61	1.90	-	61	1.90	-	61	1.90	-	121	5.04	-	386	16.08
23	-	10	0.31	-	8	0.25	-	8	0.25	-	37	1.54	-	121	5.04

A=기존의 볼륨 광선 투사법, B=제안한 기법

[표 2] 제안한 기법에서 옥트리 탐색 레벨에 따른 Foot 볼륨 데이터의 렌더링 속도 (fps)

옥트리 레벨	유효 밀도 값의 범위														
	0~255			50~255			100~255			150~255			200~255		
	A	B	B/A (%)	A	B	B/A (%)	A	B	B/A (%)	A	B	B/A (%)	A	B	B/A (%)
5123	34	34	1.00	28	28	1.00	26	26	1.00	24	24	1.00	23	23	1.00
2563	-	33	0.97	-	28	1.00	-	26	1.00	-	24	1.00	-	23	1.00
1283	-	40	1.17	-	42	1.50	-	41	1.57	-	37	1.54	-	39	1.69
643	-	55	1.61	-	62	2.21	-	76	2.92	-	79	3.29	-	83	3.60
323	-	116	3.41	-	116	4.14	-	116	4.46	-	126	5.25	-	172	7.47
163	-	160	4.70	-	160	5.71	-	160	6.15	-	209	8.70	-	296	12.86
83	-	177	5.20	-	179	6.39	-	181	6.96	-	245	10.20	-	396	17.21
43	-	101	2.97	-	101	3.60	-	101	3.88	-	187	7.79	-	308	13.39
23	-	26	0.76	-	26	0.92	-	26	1.00	-	52	2.16	-	109	4.73

A=기존의 볼륨 광선 투사법, B=제안한 기법



[그림 7] 제안한 기법에서의 각 유효 밀도 값의 범위에 따른 Engine과 Foot 볼륨 데이터의 결과 영상
(왼쪽부터 0~255, 0~255, 50~255, 100~255, 150 ~255, 200~255)

참고문헌

[1] Kurt Akeley, "Reality Engine Graphics", Proceeding of SIGGRAPH 93, 1993.

[2] Timothy J. Cullip, Ulrich Neumann, "Accelerating Volume Reconstruction with 3D Texture Hardware", University of North Carolina at Chapel Hill. NC, TR93-027, 1994.

[3] Klaus Engel, Markus Hadwiger, Joe M. Kniss, Christof Rezk Salama, "Real-time volume graphics", Eurographics 2006, 2006.

[4] Eun-Seok Lee, Byeong-Seok Shin, "Geometry Splitting: An Acce-leration Technique of Quadtree-Based Terrain Rendering Using GPU", IEICE Transactions on Information and Systems, Vol. E94-D, No.1, pp137-145, 2011.

[5] J.Krüger, R.Westermann, "Acceleration Techniques for GPU-based Volume Rendering", Proceedings of the 14th IEEE Visualization 03, 2003.

[6] Jane Wilhelms, Allen Van Gelder, "Octrees for faster isosurface generation", ACM Transactions on Graphics, Volume 11, Issue3, 1992.

[7] Wei Hong, Feng Qiu, Arie Kaufman, "GPU-based object-order ray-casting for large datasets", Volume Graphics, pp.177-185, 2005.

[8] Vincent Vidal, Xing Mei, Philippe Decaudin, "Simple Empty-Space Removal for Interactive Volume Rendering", Journal of Graphics, GPU and Game Tools, Volume 13, Issue 2, pp.21-37, 2008.

[9] Milos Sramek, Arie Kaufman, "Fast Ray-tracing of Rectilinear Volume Data Using Distance Transforms", IEEE Transactions on Visualization and Computer Graphics, Volume 6, Issue 3, pp.236-252, 2000.

[10] Jorg Mensmann, Timo Ropinski, Klaus Hinrichs, "Accelerating Volume Raycasting using Occlusion Frustums", In IEEE/ Eurographics International Symposium on Volume and Point-Based Graphics, pp.147-154, 2008.

[11] Baoquan Liu, Gordon Clapworthy, Feng Dong, "Aceelerating volume raycasting using proxy spheres". Computer Graphics Forum, Volume 28, Issue 3, pp.839-846, 2009.

유 성 열 (Yoo, Seong-Yeol)

2009년 2월 인하대학교 컴퓨터정보공학과 (학사)
2009년 2월-현재 인하대학교 컴퓨터정보공학과 (석사)

관심분야 : 볼륨 렌더링

이 은 석 (Lee, Eun-Seok)

2008년 2월 인하대학교 컴퓨터공학부 (학사)
2010년 8월 인하대학교 컴퓨터정보공학과 (석사)
2011년 2월-현재 인하대학교 컴퓨터정보공학과 (박사)

관심분야 : 지형 렌더링, 상세단계선별, 차세대 컴퓨팅

신 병 석 (Shin, Byeong-Seok)

1990년 2월 서울대학교 컴퓨터공학과 (학사)
1992년 2월 서울대학교 컴퓨터공학과 (석사)
1997년 2월 서울대학교 컴퓨터공학과 (박사)
2000년-현재 인하대학교 컴퓨터정보공학부 교수

관심분야 : 볼륨 그래픽스, 실시간 렌더링, 차세대 컴퓨팅, 의료영상