Journal of Korea Multimedia Society Vol. 14, No. 10, October 2011(pp. 1229-1237)

포인트 프리미티브를 이용한 실시간 볼륨 렌더링 기법

강동수†, 신병석††

요 약

직접 볼륨 렌더링은 반투명한 물체에 대한 고화질 영상 생성이 가능한 기법으로 광선 투사법이 대표적이다. 이것은 각 화소별로 오브젝트 공간상의 관심 영역을 샘플링하기 때문에 높은 해상도의 영상을 생성할 수 있지만, 각 샘플점마다 반복적으로 수행하는 텍스처 참조와 누적연산 때문에 렌더링 성능이 저하되는 문제가 있다. 최근에는 연산 능력이 매우 커진 GPU를 이용해 광선 투사법을 가속화하는 기법들이 많이 연구되고 있지만 이들 역시 전처리 단계 및 추가적인 메모리 사용이 불가피하다. 본 논문에서는 반투명 물체의 표현이 가능하고, 전처리 과정 및 추가적인 텍스처 메모리를 사용하지 않으면서 기존의 방법들보다 고속으로 볼륨 데이터를 가시화할 수 있는 포인트 프리미티브 기반의 새로운 볼륨 렌더링기법을 제안한다. 이 방법은 볼륨 데이터를 샘플링하여 포인트 프리미티브를 생성하고 이를 이미지 평면상에 투영하는 방식으로 수행속도가 매우 빠르다. 또한, 생성된 포인트 프리미티브를 실행시간에 추가 및 삭제할 수 있기 때문에 OTF를 변경해도 실시간 대응이 가능하다.

Real-time Volume Rendering using Point-Primitive

Dong-Soo Kang†, Byeong-Seok Shin††

ABSTRACT

The volume ray-casting method is one of the direct volume rendering methods that produces high-quality images as well as manipulates semi-transparent object. Although the volume ray-casting method produces high-quality image by sampling in the region of interest, its rendering speed is slow since the color acquisition process is complicated for repetitive memory reference and accumulation of sample values. Recently, the GPU-based acceleration techniques are introduced. However, they require pre-processing or additional memory. In this paper, we propose efficient point-primitive based method to overcome complicated computation of GPU ray-casting. It presents semi-transparent objects, however it does not require preprocessing and additional memory. Our method is fast since it generates point-primitives from volume dataset during sampling process and it projects the primitives onto the image plane. Also, our method can easily cope with OTF change because we can add or delete point-primitive in real-time.

Key words: GPU Volume Rendering(GPU 볼륨 렌더링), Point-Primitive(포인트 프리미티브), Geometry Shader(기하 쉐이더)

1. 서 론

볼륨 렌더링은 컴퓨터 그래픽스 분야중 하나로서, 3차원 데이터에 대한 효과적인 분석과 통찰력을 갖게 해주는 도구다[1,2]. 볼륨 렌더링은 크게 표면 렌더링과 직접 볼륨 렌더링으로 나뉜다[3]. 표면 렌더

※ 교신저자(Corresponding Author) : 신병석, 주소 : 인천시 남구 용현동 253 인하대학교 하이테크센터 1403호(402-751), 전화 : 032-860-7452, FAX : 032-872-7452, E-mail : bsshin@inha.ac.kr
접수일 : 2011년 5월 9일, 수정일 : 2011년 9월 1일
완료일 : 2011년 9월 23일
† 준회원, 인하대학교 컴퓨터 · 정보공학과
(E-mail : gagalchi@msn.com)
†† 정회원, 인하대학교 컴퓨터 · 정보공학과

링은 3차원 볼륨 데이터를 기하정보로 재구성한 후, 해당 기하정보를 범용 그래픽 가속기를 이용하여 렌더링하는 기법이다. 반면에 직접 볼륨 렌더링은 3차원 볼륨 데이터를 다른 형태로 바꾸지 않고 직접 볼륨 데이터로부터 영상을 얻는 기법이다. 표면 렌더링 기법의 경우, 반투명한 물체를 표현하는 것이 어려울 뿐만 아니라, 불투명도 전이함수(OTF : Opacity Transfer Function)로부터 특정 밀도 값을 갖는 복셀들을 추출하여 메쉬(mesh)로 재구성하는 비용이 크기 때문에 OTF의 변화에 실시간으로 대처할 수 없다. 직접 볼륨 렌더링은 크게 화상순서 기반 기법과 객체순서 기반 기법으로 나뉜다[4]. 화상순서 볼륨 렌더링 기법은 화상의 스캔라인 순서대로 각 화소의 값을 차례로 결정해 나가는 방식이다. 화상순서 알고리즘의 대표적인 예로 볼륨 광선 투사법이 있다[5]. 광선 투사법은 영상에서 가상의 광선이 발사된다고 가정하여 광선들이 볼륨 내부를 지나면서, 이와 교차하는 복셀들로부터 샘플링한 값을 합성하여 각 화소의 색상 값을 결정한다. 하지만 광선을 따라가며 일정 간격으로 재샘플링하는 연산은 3차원 텍스처를 반복해서 참조해야하고 각 샘플점에서 얻은 색상정보를 최종색상에 반영하기 위해서는 누적연산을 반복적으로 수행해야 한다. 이것은 고속의 렌더링을 어렵게 하며, GPU(Graphics Processing Unit) 기반의 가속화 알고리즘을 이용하여도 광선 투사법이 갖는 과도한 연산량 때문에 속도향상에는 한계가 있다. 객체순서 볼륨 렌더링은 전향매핑(forward mapping) 방식으로 볼륨 데이터를 직접 영상에 투영하는 기법이다. 이것은 볼륨 데이터의 저장순서에 따라 탐색하며 각 샘플점들을 일정 모양의 커널로 만들고 이를 이미지 평면에 투영하여 각 화소들의 색상을 결정하는 기법이다. 객체순서 알고리즘의 대표적인 기법에는 스플래팅(splatting)이 있다[6,7]. 이것은 볼륨 데이터를 일정 간격으로 샘플링한 후, 전처리 과정에서 OTF를 이용하여 투영할 샘플점들을 선별한다. 그리고 선별된 샘플점들을 육면체, 구 또는 타원체(ellipsoid) 형태의 풋프린트(footprint)로 만들어 컨볼루션(convolution) 연산으로 누적한다. 이것은 메모리에 저장된 순서대로 처리할 수 있다는 장점을 갖지만, 선-분류(pre-classification)방식을 사용하기 때문에 분류한 볼륨의 해상도가 낮은 경우 뿌옇게(blurry)보이는 단점이 있다.

본 논문에서는 기존 기법들의 단점들을 보완하기 위해 포인트 프리미티브(point-primitive)를 이용한 새로운 볼륨 렌더링기법을 제안한다. 이것은 객체순서 기반의 렌더링기법으로 기존의 볼륨 스플래팅과는 다르게 특정한 모양의 커널을 대신해 포인트 프리미티브를 사용하여 볼륨 데이터를 가시화한다. 포인트 프리미티브는 특정 형태의 커널을 이용할 때 발생하는 추가적인 정점의 개수를 최소화할 수 있으며, 적은 정점들로 이루어진 정점버퍼(vertex buffer)를 이용해 보다 빠르게 영상을 생성할 수 있다. 또한, 제안하는 기법은 기존의 스플래팅에서 사용하는 선-분류 방식과는 다르게 렌더링 단계에서 복셀당 매핑되는 유효 포인트 집합을 GPU상에서 분류한다. 따라서 OTF의 변화에 따른 복셀의 분류를 실시간에 처리 할 수 있다. 또한, 객체순서 기반의 렌더링기법들이 그런 것처럼 그래픽 가속기에 최적화된 투영방식의 단순한 렌더링 파이프 라인을 사용하기 때문에 참조와 누적을 반복하는 화상순서 기반의 기법들보다 렌더링 속도가 빠르다. 특히, 이런 기법은 시간에 따라 변하는 4D 볼륨 데이터의 렌더링 분야에 유용하게 사용될 수 있다. 기존에 제안된 기법들은 볼륨 데이터를 전처리 시간에 가공하거나, 선-분류 또는 볼륨데이터 전체를 모두 샘플링해야 하므로 4D 볼륨 데이터의 고속 렌더링에는 적합하지 않다. 하지만, 본 논문에서 제안하는 기법은 시간에 따라 변하는 영역만을 실행시간에 포인트 프리미티브로 변환할 수 있고, 이를 GPU의 렌더링 파이프라인에서 제공하는 단순한 연산을 통해 가시화할 수 있기 때문에 대화식 수준의 4D 볼륨 렌더링이 가능하다.

2장에서는 볼륨 렌더링과 관련된 가속화기법들과 스플래팅 및 GPU기반의 볼륨 렌더링 가속화 기법에 관하여 기술하고, 3장에서는 포인트 프리미티브의 생성 및 렌더링 과정에 관하여 자세히 서술한다. 4장에서는 실험한 결과를 정리하고, 기존 볼륨 렌더링기법과 성능 및 화질 측면에서 비교한다.

2. 관련 연구

일반적으로 볼륨 데이터는 크기가 방대하기 때문에 렌더링 시간이 오래 걸린다. 따라서 대화식 수준의 볼륨 렌더링을 위해서는 하드웨어적인 지원이 요구되며, 최근에는 GPU 또는 GPU에 일반 연산을 적

용하는 GPGPU(General-purpose computing on Graphic Process Unit)기술을 이용한 볼륨 렌더링 가속화기법에 관한 연구들이 활발히 진행되고 있다[8]. 특히, GPU의 정점 쉐이더(vertex shader)와 기하 쉐이더(geometry shader) 그리고 프래그먼트 쉐이더(fragment shader)를 사용하면 렌더링 속도가 매우 빨라진다[9].

볼륨 렌더링을 가속화하는 대표적인 방법으로 GPU기반 광선 투사법[10] 있다. 광선 투사법의 가속화기법 중 빈 공간 도약기법으로는 Hong이 발표한 팔진트리(octree)기반의 볼륨 렌더링기법이 있다[11]. 이것은 볼륨을 일정 크기의 브릭(brick)들로 나누고 시점변화에 따라 팔진트리를 생성하여 실행시간에 빈 공간 탐색을 효율적으로 한다. 하지만 이 방법은 시점 종속적인 단점이 있다. Vincent는 kd-트리 기반의 광선 투사법을 제안하였다[12]. 이것은 전처리 단계에서 OTF를 기반으로 사전에 kd-트리를 생성하여 유효 노드들을 미리 필터링하는 기법이다. 전처리 단계에서 생성한 트리구조를 통해 렌더링 속도를 향상할 수 있지만, 트리구조를 저장하기위한 추가적인 메모리가 필요하고, OTF가 변할 때마다 트리를 재구성해야하는 문제가 있다. Liu는 임의의 구형 기하를 사용한 가속화기법을 제안하였다[13]. 이 방법은 전처리 시간에 생성한 밉맵(mip-map) 피라미드를 이용하여 관심 있는 복셀들을 찾고, 기하 쉐이더를 이용하여 해당 복셀에 일정한 크기의 구를 생성한다. 구는 깊이 맵(depth map)을 통해 각 화소에서 발사된 광선의 시작 위치와 종료 위치를 계산하는데 사용된다. 이것은 광선 투사법에서 관심 영역에 최대한 밀착된 빈 공간 탐색과 광선을 조기종료를 수행할 수 있기 때문에 고속 렌더링이 가능하다.

고속 볼륨 렌더링기법들은 4D 볼륨 렌더링 분야에서 시간에 따라 변하는 시변(time-varing) 볼륨 데이터를 가시화하는데 많이 사용된다. 하지만 GPU의 성능향상에도 불구하고, 현재까지 일반 PC상에서 실시간으로 입력되는 시변 데이터를 대화식의 속도로 가시화하는 것은 어렵다. Shen과 Johnson은 시변 볼륨 데이터를 가시화하기위해 차등 볼륨 렌더링(differential volume rendering)기법을 제안하였다[14]. 이것은 이전 시간에 입력된 볼륨 데이터와 현재 입력된 볼륨 데이터를 비교하여 변하지 않는 영역의 중복연산을 피하기 위해 시간 일관성(temporal coherence)을 사용한다. 시간 일관성기법은 시간 축에 따라 변화된 복셀들을 참조하는 화소들에 대해서만 광선 투사를 실시하고, 변하지 않은 복셀을 참조하는 화소에 대해서는 이전 장면에서 계산된 화소의 색상을 재사용한다. Shen과 Ma는 팔진트리를 사용한 공간 일관성(spatial coherence)기법을 제안하였다[15]. 이것은 복셀 단위로 화소의 갱신여부를 계산하지 않고, 팔진트리구조로 검색하여 경신여부에 대한 탐색시간을 줄인다. 이후, Shen은 시간과 공간 일관성을 모두 사용하는 시공간 분할 트리(TSP: Time-space Partitioning Tree)기법을 제안하여 4D 볼륨 데이터 렌더링의 성능과 화질을 개선하였다[16]. 최근에는 GPU 기반의 압축 방법 또는 시공간 일관성을 이용한 4D 볼륨 데이터의 가속화기법들이 많이 연구되고 있다[17-19]. 하지만 이들은 기본적으로 광선 투사법을 사용하기 때문에 빈번한 텍스처 참조와 샘플링이 반복되는 누적연산을 볼륨의 시작부터 끝까지 모두 수행해야만 한다. 이것은 높은 해상도의 영상을 제공할 수 있지만 아직까지 대화식 속도에는 미치지 못한다. Klajnsek은 광선 투사를 기반으로 하는 방법들에서 나타나는 과도한 연산을 줄이기 위해 객체순서 기반의 차등 스플래팅(differential splatting)기법을 제안하였다[20]. 이것은 2차원의 풋프린트 커널을 생성하여 변형된 복셀에 대해서만 커널을 이미지 평면에 투영시키는 방식이다. 이 방법은 광선 투사법 기반의 접근방식들 보다는 빠르게 영상을 생성할 수 있지만, CPU 기반의 알고리즘을 사용하기 때문에 128^3의 데이터에서 약 3*fps*의 속도를 보인다. 또한, 직교투영에만 중점을 두었기 때문에 투시투영의 경우 결과영상의 화질이 저하된다.

3. 포인트 프리미티브를 이용한 볼륨 렌더링

본 논문에서는 추가적인 메모리 사용 및 전처리 과정을 사용하는 기존의 볼륨 렌더링 기법과는 다르게 그래픽 장치에서 사용하는 그리기의 최소 단위인 포인트 프리미티브 집합의 투영만으로 볼륨 데이터를 빠르게 가시화하는 새로운 방법을 제안한다. 제안하는 기법은 고속의 렌더링을 위해 기존의 고정 그래픽스 파이프라인을 이용하지 않고, GPU내의 파이프라인을 직접 프로그래밍할 수 있는 쉐이더 아키텍처를 사용하였다. 우선, 입력 정점들의 행렬 연산을 주

로 담당하는 정점 쉐이더를 통해 이미지 평면에 최종적으로 투영시킬 유효 포인트 프리미티브의 위치를 계산하여 생성하고, 입력된 정점의 삭제와 생성 및 변형을 담당하는 기하 쉐이더를 이용하여 유효 포인트 프리미티브를 실시간에 선별하는 방법을 고안하였다. 이것은 실시간으로 변화되는 OTF에 빠르게 대응하는 것을 가능하게 한다. 또한, 3차원 모델의 레스터화(rasterization)를 수행하는 프레그먼트 쉐이더 단계에서 최종적으로 선별된 포인트 프리미티브의 밀도값을 이미지 평면상에 투영하여 각 화소의 색상 값을 알파 블렌딩을 통해 계산하도록 하였다. 이것은 기존의 방법들과는 다르게 그래픽 하드웨어에서 제공하는 알파 블랜딩과 3차원 정점의 투영 연산만으로 최종 영상을 생성할 수 있기 때문에 연산속도가 매우 빠르다. 제안하는 기법의 전체적인 수행과정은 다음과 같다(그림 1). 우선, 포인트 프리미티브 생성 단계에서 입력된 볼륨 데이터의 복셀 개수만큼 임의의 포인트 리스트를 생성한다. 해당과정은 정점

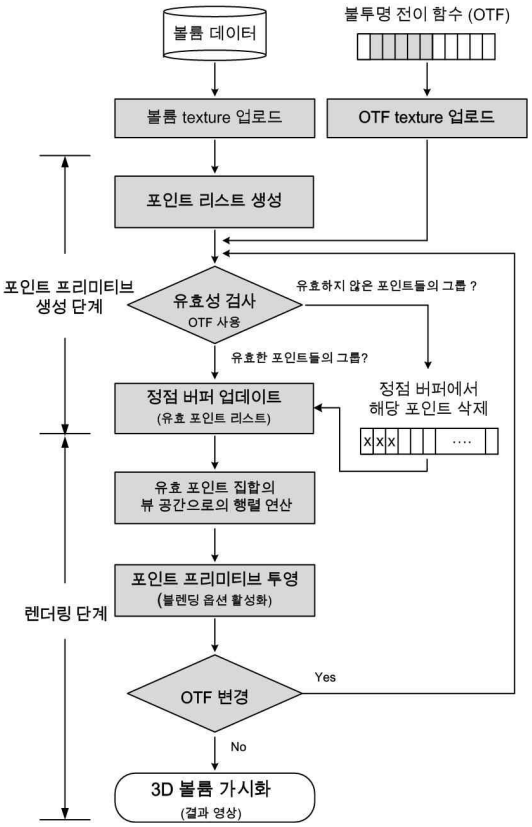

그림 1. 제안한 기법의 전체 수행과정

쉐이더에서 수행되며, 반복 없이 한 번의 연산으로 빠르게 수행된다. 두 번째 단계에서는 기하 쉐이더를 통해 OTF에 의해 비투명한 것으로 정해진 포인트 리스트들만을 선별하는 유효성 검사를 수행한다. 유효한 포인트들의 집합은 실제로 GPU 메모리상의 정점버퍼로 업데이트된다. 마지막 단계에서는 유효 포인트 리스트를 행렬연산을 통해 뷰(view)공간으로 이동시켜 포인트 프리미티브를 생성하고, 프래그먼트 쉐이더에서 객체순서 기반의 투영연산을 수행하여 입력된 볼륨 데이터를 가시화한다.

3.1 포인트 프리미티브 생성

기존의 GPU에서 사용하는 쉐이더는 렌더링 과정에서 정점 쉐이더로부터 마지막 단계인 프레그먼트 쉐이더까지 순차적으로 수행해야만 했다. 하지만 DirectX 10 이후의 쉐이더 아키텍처는 그림 2처럼 중간 단계인 기하 쉐이더까지만 수행하는 것이 가능하며, 파이프라인 내에서 원하는 쉐이더 단계를 반복 수행하는 것도 가능하다[21]. 이것은 렌더링 단계에서 입력된 정점 데이터의 생성, 삭제, 이동을 가능하게 한다.

제안하는 기법은 렌더링단계에서 정점 쉐이더와 기하 쉐이더 단계의 반복을 이용하여 정점버퍼에 저장된 포인트 프리미티브 리스트를 생성하거나 삭제한다. 포인트 리스트의 생성과정은 다음과 같다. 우선, 볼륨 데이터를 일정 간격으로 샘플링하여 포인트 프리미티브를 생성할 후보 샘플점들을 추출한다. 실제로 후보 샘플점들은 GPU 메모리상의 정점버퍼에 저장되고, 다음 단계인 기하 쉐이더로 보내진다. 기하 쉐이더에서는 미리 입력된 OTF를 이용하여 비투명도를 갖는 유효 포인트 집합을 선별한다. 이것은 기하 쉐이더의 스트림 아웃(SO : stream out)기능을 이용하여 정점버퍼에서 유효하지 않은 포인트들은 삭제하고 최종적으로 유효 포인트만을 남긴다. 이 과정은 각 포인트가 하나의 쓰레드(thread)가되어 GPU상에서 병렬로 수행된다.

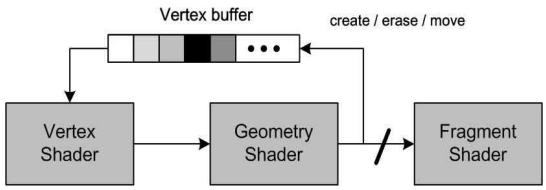

그림 2. GPU 상에서의 정점버퍼의 피드백(feedback)

3.2 포인트 프리미티브를 이용한 볼륨 렌더링

정점버퍼에 저장된 유효 포인트들은 모델 좌표계에서 뷰 공간으로의 행렬연산을 통해 포인트 프리미티브의 좌표로 변환된다. 각각의 포인트 프리미티브는 4개의 부동소수점으로 구성된 벡터에 저장되는데 처음 세 개의 필드에는 변환된 3차원 공간상의 (x, y, z)의 좌표 값을 그리고 마지막 필드(w)에는 볼륨 데이터로부터 샘플링한 밀도(density) 값을 저장한다. 포인트 프리미티브는 프래그먼트 쉐이더로 이동하여 w값을 투명도로 간주하여 알파 블랜딩을 수행하며, 그림 3과 같이 각 화소별 투영 공간에 최종적으로 그려진다. 각 화소는 수식 1과 같이 프래그먼트 쉐이더에서 제공하는 알파 블랜딩 연산을 통해 프레임 버퍼로 투영된다.

$$C_i' = C_i + (1 - A_i)\, C_{i-1}' \quad (1)$$

수식 1에서 Ci는 화소에 현재까지 누적된 포인트 프리미티브 집합들의 색상 값이며, Ci는 현재 입력된 포인트 프리미티브의 색상 값, Ai는 현재 샘플링된 포인트 프리미티브의 밀도 값, Ci-1는 이전까지 누적된 색상 값을 나타낸다.

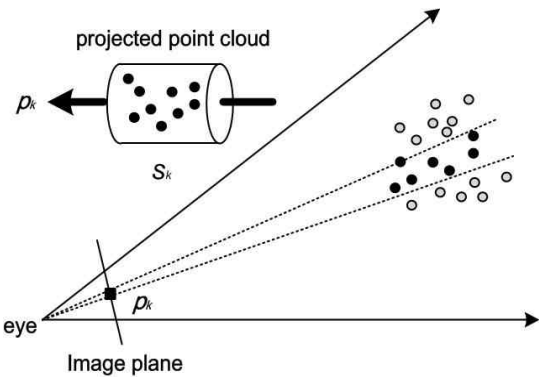

그림 3. 이미지 평면상에 각 화소별로 투영되는 포인트 프리미티브의 집합. Pk는 이미지 평면상의 한 화소이며, Sk는 Pk에 투영될 뷰 공간상의 포인트 프리미티브의 집합

3.3 OTF의 실시간 변경

제안하는 포인트 프리미티브 기반의 볼륨 렌더링은 실행시간에 OTF가 변경될 경우, 기하 쉐이더의 스트림 아웃기능을 이용하여 변경된 샘플점들을 빠르게 찾아 생성하고 삭제할 수 있다. 하지만 기존 스플래팅의 경우에는 OTF가 변경될 때마다 CPU상에서 순차적으로 볼륨 데이터 전체를 탐색하여 유효 커널들을 다시 선별하여 생성해야만 한다. 일반적으로 의료분야 응용에서는 사용자인 의사의 요구에 따라 수시로 OTF가 변경된다. 기존의 스플래팅 기법은 이런 요구에 취약할 뿐만 아니라, 볼륨 데이터의 용량이 클 경우 전처리 시간이 선형적으로 증가하기 때문에 대화식 속도의 렌더링은 불가능하다. 하지만 제안하는 기법은 그림 1에서 보는 것과 같이 사용자에 의해 OTF가 변경될 경우에 실시간으로 기하 쉐이더를 이용해 손쉽게 필요한 포인트 프리미티브 집합들을 선별하여 삭제하거나 재생성하는 것이 가능하기 때문에 실시간 렌더링이 가능하다.

4. 실험 결과

본 논문에서는 NVIDIA GeForce™ GTX260 그래픽 카드를 장착한 Intel Core2Duo 6400 2.13GHz PC에서 실험하였다. 뷰 포트(viewport)의 크기는 512×512을 사용하였고, DirectX 10 라이브러리를 사용하였다. 실험은 표 1과 같이 4가지의 볼륨 데이터에 적용하였다. 제안하는 기법의 성능평가는 Krüger가 제안한 광선 조기종료기법이 적용된 GPU 광선 투사법[10]과 동일한 데이터에 대해 수행속도(fps)을 비교하여 평가하였다. 아래의 그림 5는 주어진 4가지의 데이터에 대해 제안된 기법으로 가시화한 결과영상이다.

제안하는 기법의 성능은 표 2에서 보는 것과 같이 광선 투사법에 비해 약 두 배 빠르다. 객관적인 비교를 위해 제안하는 기법의 결과 영상과 유사하도록 광선 투사법의 샘플링 간격을 늘려서 렌더링하고 이때의 속도를 비교하였다. 일반적으로 광선 투사법은 특별한 자료구조를 사용하지 않는 한 입력 데이터의 크기에 비례하여 렌더링 시간이 증가한다. 하지만 제안된 기법의 성능은 데이터의 크기 보다는 유효 포인

표 1. 실험에 사용된 데이터의 종류

종 류	비트 수 (Bit)	크기	데이터 용량
Phantom	8	256×128×110	4.45 MB
Big Head	8	256×256×256	14 MB
Engine	8	512×512×512	128 MB
Foot	8	512×512×512	128 MB

표 2. GPU 광선 투사법과 제안된 기법의 수행시간 비교

종 류	광선 투사법(A)	포인트 프리미티브 볼륨 렌더링(B)	유효 포인트 프리미티브 개수	속도향상 (B/A)
Phantom	183 fps	320 fps	3,012,295	1.75
Big Head	143 fps	162 fps	7,146,549	1.13
Engine	106 fps	186 fps	9,988,646	1.75
Foot	108 fps	235 fps	6,777,711	2.18

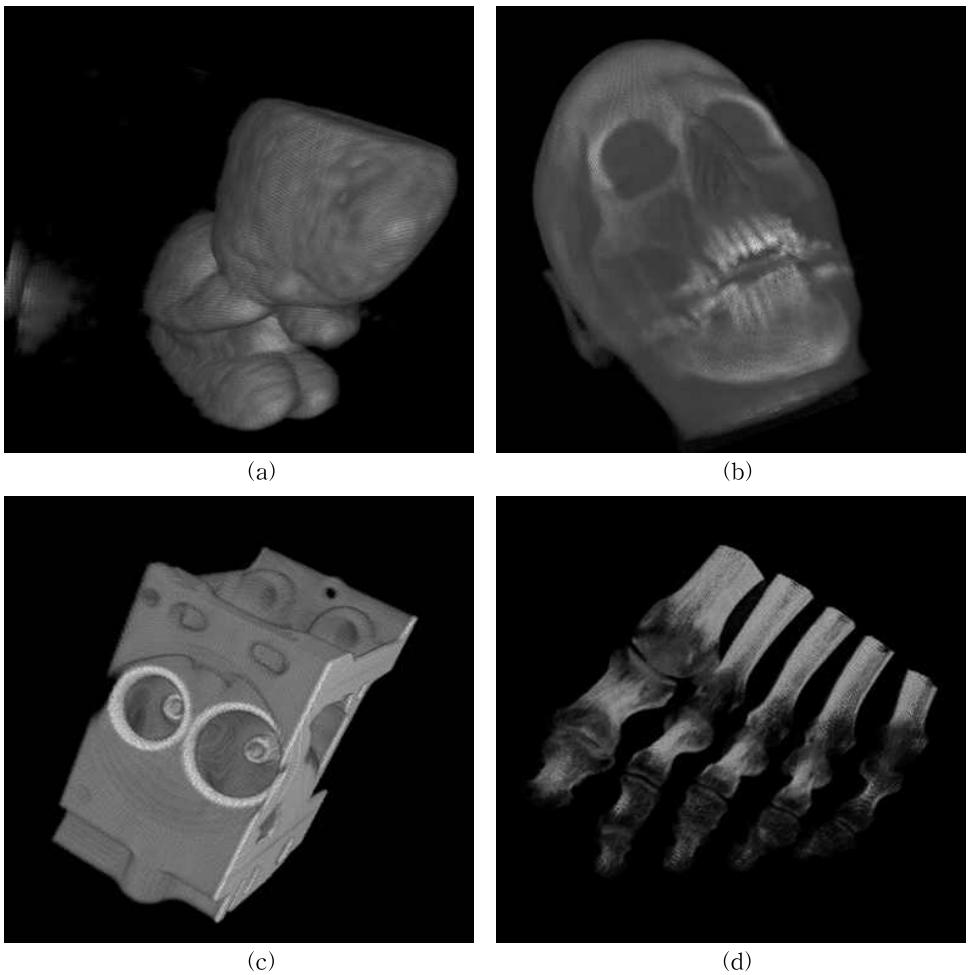

그림 4. 포인트 프리미티브를 이용한 볼륨 렌더링기법의 결과 영상. (a) Phantom, (b) Big head, (c) Engine, (d) Foot

트 프리미티브의 개수와 밀접한 관계가 있다. 즉, 입력된 데이터의 크기보다는 OTF에 의해 분류된 포인트 프리미티브의 개수가 렌더링 속도를 결정한다.

그림 5는 제안하는 기법과 GPU 광선 투사법의 결과영상을 비교한 것이다. 그림에서 보는 것과 같이 포인트 프리미티브 기반의 볼륨 렌더링기법이 광선 투사법에 비해 색상이 어두운 것을 알 수 있다. 결과 영상의 밝기의 차이는 광선 투사법의 경우 후향 맵핑으로 색상을 혼합(compositing)하는 방식을 사용하고, 제안하는 기법의 경우는 전향 맵핑인 스플래팅 방식을 사용하기 때문이다. 일반적으로 광선 투사법에서 샘플링 간격을 작게 하여 생성한 영상과 스플래

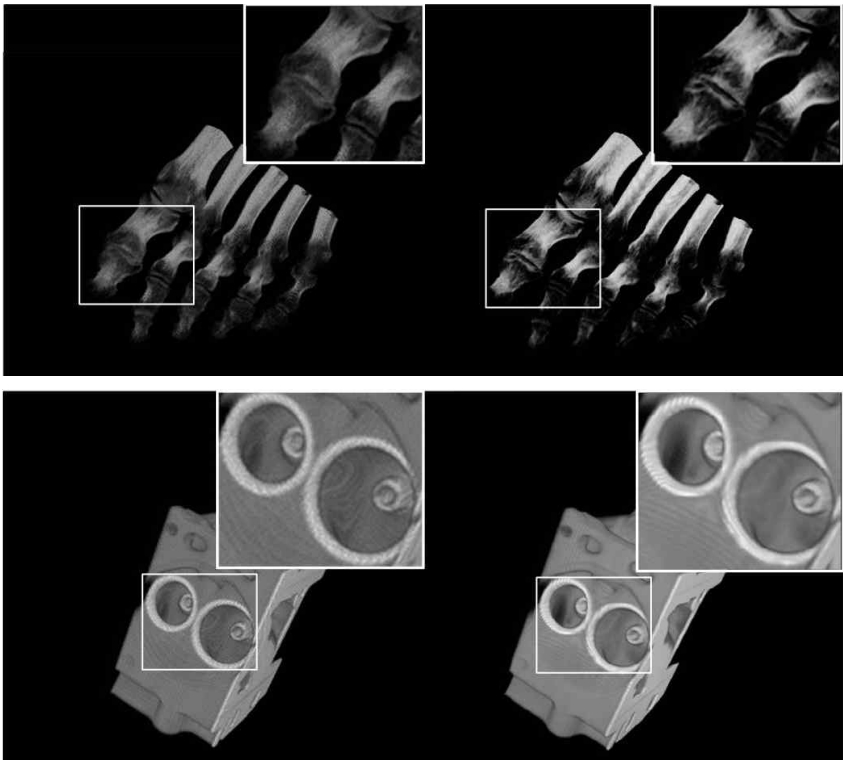

그림 5. 제안 기법(왼쪽)과 광선 투사법(오른쪽)을 이용한 렌더링 결과(Engine 데이터(위) - 512^3, Foot 데이터 (아래) - 512^3)

팅 기반 볼륨 렌더링 기법의 화질을 비교하면 상대적으로 후자의 방법이 보다 거칠게 표현된다. 이것은 이미지 평면으로 투영되는 포인트 프리미티브의 크기가 이미지 평면상의 한 화소의 크기보다 작기 때문에 생기는 구멍(hole) 때문이다. 이것을 화소 오차(pixel error)라고 부른다. 결과영상에 구멍이 나타나지 않도록 하기 위해서는 화소 오차가 발생하지 않도록 실시간으로 유효 프리미티브의 개수를 조절해야 한다 하지만 유효 프리미티브의 증가는 렌더링 속도를 저하시키기 때문에 화질과 성능사이의 적절한 균형 유지가 중요하다.

5. 결 론

GPU 광선 투사법은 프레그먼트 쉐이더에서 반복적인 텍스쳐의 참조와 누적연산을 수행하는 반면에 제안하는 기법은 포인트 프리미티브에 대하여 한 번의 텍스쳐 참조만을 하기 때문에 수행속도가 매우 빠르다. 또한, 포인트 프리미티브를 실행시간에 추가 및 삭제할 수 있기 때문에 OTF가 바뀌어도 즉각적으로 영상을 생성할 수 있다. 이런 특징들은 4D 볼륨 렌더링 분야에서 시간차를 두고 입력되는 시변 데이터에 대하여 변화된 부분만을 갱신하고 나머지 부분은 재사용할 수 있도록 한다. 기존의 방법들은 과도한 연산량과 추가적인 전처리 단계 그리고 선-분류 기법등이 필요하기 때문에 시변 볼륨 데이터의 실시간 렌더링이 어렵다. 하지만 제안하는 기법은 렌더링 단계에서 실시간으로 유효 프리미티브 집합을 변환하고 갱신할 수 있기 때문에 시변 볼륨 데이터의 고속 렌더링이 가능하다. 이것은 최근 큰 이슈가 되고 있는 4D 초음파 렌더링이나 입자 시뮬레이션 분야에 매우 유용하다. 향후, 이런 문제점들은 점 기반의 상세단계(LOD : Level-Of-Detail)선택 기법을 이용하여 해결할 수 있으며, 이것은 포인트 프리미티브의 개수를 실시간으로 조절할 수 있기 때문에 효과적으로 렌더링 성능과 결과영상의 화질을 관리할 수 있을 것이다.

참 고 문 헌

[1] M. Levoy, "Display Of Surface From Volume Data," *IEEE Computer Graphics and Applications*, Vol.8, No.3, pp. 29-37, 1988.

[2] A. Kaufman, *Volume Visualization*, IEEE Computer Society Press, 1991.

[3] T. Elvins, "A Survey Of Algorithms For Volume Visualization," *Computer Graphics*, Vol.26, No.3, pp. 194-201, 1992.

[4] B. Lichtenbelt, R. Crane, and S. Naqvi, *Introduction To Volume Rendering*, Hewlett-Packard Professional Books, 1998.

[5] A. Sabella, "A Rendering Algorithm OR Visualizing 3D Scalar Fields," *Computer Graphics*, Vol.22, No.4, pp. 51-58, 1988.

[6] G. Herman and J. Udupa, "Display Of Three-Dimensional Discrete Surfaces," *Proceeding SPIE*, Vol.283, pp. 90-97, 1981.

[7] D. Gordon and R. Reynolds, "Image-Space Shading Of 3-Dimensional Objects," *Computer Vision, Graphics, and Image Processing*, Vol.29, pp. 361-376, 1985.

[8] 계희원, 김준호,"GPGPU 환경에서 최대휘소투영 렌더링의 고속화 방법," 한국멀티미디어학회 논문지, 제 14권 8호, pp. 981-991, 2011.

[9] R. Fernando, *GPU Gems, Programming Technique, Tips, And Tricks For Real-time Graphics*, Addison-Wesley, second printing, 2004.

[10] J. Krüger and R. Westermann, "Acceleration Techniques For GPU-based Volume Rendering," *In Proceedings of IEEE Visualization*, pp. 287-292, 2003.

[11] W. Hong, F. Qui, and A. Kaufman, "GPU-Based Object-Order Ray-Casting For Large Datasets," *In Volume Graphics*, pp. 177-185, 2005.

[12] V. vidal, X. Mei, and P. Decaudin, "Simple Empty-Space Removal For Interactive Volume Rendering," *Journal of Graphics, GPU, and Game Tools*, Vol.13, No.2, pp. 21-36, 2008.

[13] B. Liu, G. Clapworthy, and F. Dong, "Accelerating Volume Raycasting Using Proxy Sphere,"*IEEE VGTC Symposium on Visualization*, 2009.

[14] H. Shen and C. Johnson, "Differential Volume Rendering: A Fast Volume Visualization Technique For Flow Animation," *IEEE Visualization*, pp. 180-187, 1994.

[15] K. Ma and H. Shen, "Compressing And Accelerated Rendering Of Time-Varying Volume Datasets," *In Workshop on Computer Graphics and Virtual Reality, International Computer Symposium*, 2000.

[16] H. Shen, L. Chiang, and K. Ma, "Fast Volume Rendering Algorithm For Time-Varying Fields Using A Time-Space Partitioning (TSP) Tree," *IEEE Visualization*, pp. 371-377, 1999.

[17] E. Lum, K. Ma, and J. Clyne, "A Hardware-Assisted Scalable Solution For Interactive Volume Rendering OF Time-Varying Data," *IEEE Transactions on Visualization and Computer Graphics*, Vol.8, No.3, pp. 286-301, 2002.

[18] J. Woodring and H. Shen, "Chronovolumes: A Direct Rendering Technique For Visualizing Time-Varying Data," *Proceedings of the 2003 Eurographics/IEEE TVCG Workshop on Volume Graphics*, pp. 27-34, 2003.

[19] D. Tost and S. Grau, "Ray-Casting Time-Varying Volume Data Sets With Frame-to-Frame Coherence," *In Proceedings of SPIE-IS Electronic Imaging*, Vol.606006, pp. 1-10, 2006.

[20] G. Klajnsek and B. Csebfalvi, "4D Volume Rendering By Differential Splatting," *Information Technology Interfaces*, pp. 651-656, 2003.

[21] S. Patidar, S. Bhattacharjee, J. Singh, and P. Narayanan, "Exploiting the Shader Model 4.0 Architecture," Technical Report IIIT Hyderabad, 2006.

강 동 수

2005년 2월 인하대학교 컴퓨터·정보공학과(학사)
2007년 2월 인하대학교 컴퓨터·정보공학과(석사)
2008년 3월～현재 인하대학교 컴퓨터·정보공학과 박사과정

관심분야 : 볼륨 그래픽스, 의료 영상, 병렬 처리

신 병 석

1990년 2월 서울대학교 컴퓨터공학과(학사)
1992년 2월 서울대학교 컴퓨터공학과(석사)
1997년 2월 서울대학교 컴퓨터공학과(박사)

2000년～현재 인하대학교 컴퓨터공학부 교수
관심분야 : 실시간 렌더링, 볼륨 그래픽스, 의료 영상