GPU 클러스터를 이용한 VKH 데이터의 빠른 볼륨 렌더링

이 중 연
한국과학기술정보연구원 슈퍼컴퓨팅센터
jylee@kisti.re.kr

Fast Volume Rendering of VKH dataset using GPU Cluster

Joong-Youn Lee
Supercomputing Center, Korea Institute of Science and Technology Information

요 약

볼륨 렌더링은 3차원이나 그 이상의 차원의 볼륨 데이터에서 의미있는 정보를 추출해 내어 직관적으로 표출하는 가시화 기법을 말하며 의료영상, 기상학, 유체역학 등 다양한 분야에서 널리 사용되고 있다. 한편, 최근 PC 하드웨어의 급격한 발전으로 과거에는 슈퍼컴퓨터에서나 가능했던 대용량 볼륨 데이터의 가시화가 일반 PC 환경에서도 가능하게 되었다. PC 그래픽스 하드웨어의 꼭지점 및 픽셀 세이더의 수치 계산에 최적화된 벡터 연산으로 빠른 볼륨 가시화를 가능하게 한 것이다. 그러나 그래픽스 하드웨어의 메모리 용량의 한계로 대용량의 볼륨 데이터를 빠르게 가시화하는 것은 지금까지 어려운 문제로 남아있다. 본 논문에서는 한국과학기술정보연구원에서 제작한 대용량의 인체영상 데이터인 Visible Korean Human 데이터를 여러 개의 그래픽스 하드웨어 메모리에 분산시키고 이를 꼭지점 및 픽셀 쉐이더를 이용하여 빠르게 가시화하여 고해상도의 이미지를 얻고자 하였다.

1. 서 론

여러 가지 과학적 가시화 기법 중 가장 중요한 위치를 차지하는 볼륨 가시화는 3차원이나 그 이상의 차원의 볼륨 데이터에서 의미있는 정보를 추출해 내어 직관적으로 표출하는 가시화 방법으로 의료영상, 기상학, 유체역학 등 다양한 분야에서 활용되고 있다. 최근 대용량의 볼륨 데이터에 대한 고품질의 실시간 가시화에 대한 필요성이 급격히 증가하고 있으며 이를 원활하게 처리하기 위한 시도가 꾸준히 있어왔다. 특히, 한국과학기술정보연구원에서 제작한 Visible Korean Human(VKH) 데이터는 한국인 최초의 고해상도의 인체영상 데이터로써, MRI, CT 등 다양한 영상들의 집합이다[1]. 미국의 경우 NLM(National Library of Medicine)에서 이미 지난 86년부터 Visible Human 프로젝트를 기획하여, 지난 94년과 95년에 각각 백인 남성 및 여성에 대한 정밀 영상 데이터를 배포하였다. 그러나, 이 데이터는 한국인의 신체와는 구조가 달라 국내에서는 교육과 연구에 어려움이 많았다. 한편, 최근 수년간 PC 그래픽스 하드웨어가 급속도로 발전하면서 전문적인 3D 그래픽스 워크스테이션에서나 가능하였던 고품질, 고성능의 그래픽스 작업이 일반 PC에서도 가능하게 되었다. 특히 고정되어 있던 그래픽스 파이프라인을 사용자가 임의로 수정하여 사용할 수 있도록 한 꼭지점 및 픽셀 쉐이딩(vertex & pixel shading) 기능은 가히 혁명적이라고 할 수 있을 정도로 PC 그래픽스 하드웨어의 성능을 한단계 끌어올렸다[2]. 그러나 PC 그래픽스 하드웨어를 이용한 렌더링 방법들은 대부분 가시화할 볼륨 데이터를 3차원 텍스쳐로 간주하여 텍스쳐 메모리에 읽어 들이고 렌더링하기 때문에 가시화가 가능한 데이터의 크기가 텍스쳐 메모리의 크기에 제약받는다는 한계가 있다. 현재 최신의 PC 그래픽스 하드웨어의 텍스쳐 메모리는 256MB~512MB 수준이기 때문에 볼륨 데이터의 크기 역시 이 정도로 한정되며 여기에 법선 벡터나 팔진트리 데이터 등 고품질/고속으로 영상을 얻기 위한 추가적인 데이터들의 공간을 제외하면 렌더링 가능한 볼륨 데이터의 크기는 더욱 작아지게 된다. 본 논문에서는 이러한 그래픽스 하드웨어 기반 볼륨 렌더링의 단점을 극복하기 위해 다수의 그래픽스 하드웨어를 병렬로 연결한 GPU 클러스터를 이용하여 VKH 데이터를 빠르게 가시화하고자 하였다. 이를 위하여 볼륨 데이터를 미리 나누어 각 렌더링 노드에 분산시켜 병렬로 렌더링하는 Sort-Last 렌더링 기법을 이용하였고, 이렇게 생성된 이미지들을 빠르게 컴포지팅하기 위해 이진 치환 (binaryswap) 컴포지팅 기법이 이용되었다[3,4]. 본 논문의 2장에서는 VKH 데이터에 대해 소개하고 3장에서는 GPU 기반 볼륨 렌더러에 대해 설명한다. 4장에서는 GPU 클러스터를 이용한 병렬 볼륨렌더링에 대해 소개하고 5장에서 실제 구현환경 및 구현 결과를 설명하며 마지막으로 6장에서 결론을 맺는다.

2. Visible Korean Human(VKH) 데이터

VKH 데이터는 한국과학기술정보연구원에서 구축한 한국인을 대상으로 한 고해상도 인체영상 데이터로써, 미국 NLM에서 제작한 Visible Human 데이터 이후 세계에서 두 번째로 제작된 고해상도 인체영상 데이터이다. Visible Human 데이터는 서양인을 대상으로 생성한 인체영상 데이터로 한국 사람에게는 적용하기 어려울 뿐만 아니라 MRI 데이터가 CT 및 절단면 상생과 완전히 들어맞지 않고 없어진 절단면 영상이 있어 완전하지 못하며 사용하는데 비용이 많이 들었다. 이러한 이유로 한국과학기술정보연구원에서는 한국인의 표준 인체영상 데이터를 제작하여 DB화하기로 하였으며 이러한 노력의 결과로 만들어진 것이 VKH 데이터이다. VKH 데이터는 MRI, CT 및 2차원 절단면 영상, 구역화 영상 자료로 구성되어 있다. MRI 및 CT 데이터의 경우 512X512 해상도, 1mm의 간격으로 촬영하였으며 총 1800장으로 이루어졌고, 슬라이스 당 크기는 769KB, 전체 데이터의 크기는 각 데이터 당 1.4GB이다. 2차원 절단면 영상의 경우 0.2mm의 간격으로 CT나 MRI 데이터보다 정밀하게 촬영하였는데, 3,040X2,008 해상도에 총 9000장의 슬라이스로 이루어졌고, MRI나 CT 데이터와는 달리 24bit 컬러 영상으로 촬영하여 슬라이스 당 크기는 17,890KB, 전체 데이터의 크기는 160GB에 달한다. 구역화 영상은 인체영상 데이터들의 구역을 구분하여 그 데이터의 어느 부분이 실제 인체의 어느 장기인지를 표시한 데이터로써, 2차원 절단면 영상을 가공하여 제작하였다. 따라서 절단면 영상과 같은 간격, 같은 해상도를 가지고 있다.

3. GPU 기반의 볼륨 렌더링

3.1. 샘플링

볼륨 데이터에서 적절한 복셀(voxel)을 찾아오는 샘플링 과정은 매우 많은 시간을 필요로 하므로 이를 빠르게 처리하는 것은 주요한 연구 주제 중 하나였다. 그래픽스 하드웨어의 텍스쳐 매핑 기능은 선형보간 또는 삼선형보간을 하드웨어적으로 매우 빠르게 처리할 수 있도록 해주는데, 텍스쳐 매핑을 이용한 볼륨 렌더링은 볼륨 데이터를 텍스쳐로 간주하고 하드웨어 텍스쳐 매핑을 통하여 샘플링을 매우 빠르게 처리하도록 한다[5]. 이렇게 텍스쳐 매핑을 이용하여 샘플링을 하기 위해서는 텍스쳐가 입혀질 도형이 필요한데 이를 대리 평면(proxy plane)이라고 한다. 보통 2차원 텍스쳐에 볼륨 데이터를 저장할 경우에는 대리 평면을 텍스쳐에 평행하도록 배치하고, 3차원 텍스쳐에 볼륨 데이터를 저장할 경우에는 영상평면(image plane)에 평행하도록 배치한다. 간편하게 대리 평면이 영상평면과 평행하도록 하기 위해서 그래픽스 하드웨어의 꼭지점 쉐이더(vertex shader)를 이용하였는데, 꼭지점 쉐이더에서는 각 꼭지점에 모델뷰 행렬(modelview matrix)과 투영 행렬(projection matrix)를 곱함으로써 각 꼭지점들에 대한 스크린 좌표를 계산하도록 한다. 여기서, 모델뷰 행렬에 회전 행렬을 곱함으로해서 꼭지점이 실질적으로 회전하게 되는데, 대리 평면은 항상 영상평면에 평행으로 회전하면 안되기 때문에 꼭지점 쉐이더에서 꼭지점에 모델뷰 행렬을 무시하고 투영행렬만 곱해지도록 하였고, 대신 각 대리 평면의 꼭지점에 할당되는 텍스쳐 좌표에 모델뷰 행렬을 곱함으로써 볼륨 데이터가 회전하는 것처럼 보이도록 하였다.

3.2. 쉐이딩

샘플링된 복셀들은 전이함수(transfer function)을 통해 색깔 및 투명도가 결정되고 여기에 쉐이딩을 통해 음영이 입혀지게 된다. 이러한 작업은 프로그래밍이 가능한 쉐이더(programmable shader)를 이용하여 수행된다. 전이함수는 종속 텍스쳐(dependent texture) 기법을 이용하면 쉽게 구현이 가능하다[6]. 본 논문에서는 2차원 텍스쳐를 이용한 2차원 전이함수를 사용하였는데, 복셀값과 복셀의 법선벡터의 크기를 인덱스로 하였다. 법선벡터의 크기가 크면 복셀값이 급격히 변화하므로 물질의 경계 부분이라는 뜻이고 그 크기가 작으면 복셀값의 변화가 거의 없는 균일(homogeneous) 영역이라는 뜻이다. 일반적으로 물질의 경계면이 중요하게 인식되므로 이 부분의 투명도를 작게 하고 경계면이 아닌 균일 영역은 상대적으로 덜 중요하므로 투명도를 크게 하였다. 쉐이딩은 퐁의 조명 모델(Phong's illumination model)을 사용하였고, 그래픽스 하드웨어의 픽셀 쉐이딩(pixel shading) 기능을 이용하여 구현하였다. 퐁의 조명 모델을 계산하기 위해서 법선벡터를 복셀값과 함께 3차원 텍스쳐에 저장하였고 이 때문에 텍스쳐의 크기는 본래 볼륨 데이터의 3배 크기가 되었다.

3.3. 속도 향상 기법

3.4.1. 회전

볼륨 데이터를 회전시킬 때에 퐁 쉐이딩을 올바르게 적용시키기 위해서는 각 복셀의 법선 벡터들 역시 회전시켜야 한다. 이 경우 모든 프래그먼트(fragment)마다 법선 벡터를 회전시켜야 하므로 픽셀 쉐이더에서 매우 많은 연산을 처리하게 되어 렌더링 속도가 저하된다. 이러한 점을 해결하기 위해서 본 논문에서는 쉐이딩 계산 때에 복셀 및 법선 벡터는 회전시키지 않고 쉐이딩에 영향을 주는 다른 인자인 광원 및 시점을 회전 방향과 반대 방향으로 회전시켰다. 광원 및 시점은 프래그먼트에 독립적이므로 픽셀 쉐이더에서 계산할 필요없이 매 프레임마다 한번씩만 CPU로 계산해 주면 되기 때문에 픽셀 쉐이더에 부하가 적게 걸리게 된다.

3.4.2. 빈공간 건너 뛰기

본 논문에서는 모든 복셀에 대해 픽셀 쉐이더에서 퐁 쉐이딩을 적용하였는데, 높은 수준의 렌더링 이미지를 얻기 위하여 특별히 반사 광원(specular light)을 포함한 완전한 퐁 쉐이딩을 사용하여 매우 복잡한 연산을 수행하도록 하였다. 이러한 이유로 전체 그래픽스 파이프라인 중 픽셀 쉐이더에서 병목현상을 보여 전체 성능이 저하됨을 알 수 있었다. 이러한 단점을 극복하기 위해서 쉐이딩이 필요 없는 복셀에 대해서는 쉐이딩을 수행하지 않고 건너뛰도록 하여 전체 속도를 향상 시키고자 하였고, 이를 이중 패스 렌더링과 이른 깊이 테스트를 사용하여 구현하였다.

이른 깊이 테스트는 픽셀 쉐이더를 수행하기 전에 미리 깊이 테스트를 하여 테스트를 통과하는 프래그먼트(fragment)에 대해서만 픽셀 쉐이더를 적용하고 통과하지 못한 프래그먼트들은 모두 픽셀 쉐이더 작업을 생략하도록 하는 기법이다. 이러한 이른 깊이 테스트를 이용하여 빈 복셀에 대해서는 퐁 쉐이딩을 수행하지 않도록 하였는데, 이중 패스 렌더링 시 똑같은 대리 도형을 같은 위치에 두 번 그리도록 하고, 처음 그릴 때는 복잡한 퐁 쉐이딩을 적용시키지 않고 단순히 복셀 체크만 수행하여 그 복셀이 비었는지 여부만을 판단한다. 복셀이 비었을 경우에는 깊이 버퍼를 0으로 설정하여 두 번째 렌더링 시 깊이 테스트에서 건너뛰도록 하고 복셀이 비어 있지 않았을 경우에는 깊이 버퍼를 1로 설정하여 두 번째 렌더링 시 복잡한 퐁 쉐이딩을 수행하도록 하였다. 전체적인 알고리즘은 다음과 같다.

```
Firstpass:
    Draw proxy slice
    Check voxel value in pixel shader
    if (voxel value >= threshold) depth buffer = 1
    elseif (voxel value < threshold) depth buffer = 0
Secondpass:
    Draw proxy slice
    Apply Z test
    if (depth buffer == 0) skip pixel shader
    else do pixel shader
```

그림 1. 이른 깊이 테스트를 이용한 빈공간 건너뛰기 기법 알고리즘

4. GPU 클러스터를 이용한 병렬 렌더링

GPU를 이용한 볼륨 렌더링은 기존 CPU를 이용한 렌더링 방식에 비해 속도가 매우 빠르다는 장점이 있으나 GPU 메모리의 크기에 한계가 있고 메모리나 하드디스크 등이 다른 저장장치와의 통신 속도가 매우 느리다는 특징으로 인하여 렌더링 가능한 데이터의 크기에 한계가 있는 단점이 있다. 또한, 통신 속도가 느린 특징으로 인해 데이터를 분산 시켜 렌더링할 경우 렌더링 과정에서 최대한 각 노드 상호간에 통신을 최소화해야 한다. 이러한 점을 극복하기 위하여 본 논문에서는 렌더링될 데이터를 미리 그래픽스 메모리의 크기에 맞도록 나누었으며 이를 Sort-Last 기법을 통하여 렌더링되도록 하였다. Sort-Last 기법이란, 렌더링될 데이터가 최종 렌더링 이미지의 어느 영역에 위치하던지 상관하지 않고 일단 렌더링한 뒤 최종 컴포지팅 단계에서 그 위치를 맞추도록 하는 병렬 렌더링 기법을 말한다. 이 렌더링 기법에서는 디스플레이 작업이 수행되기

까지는 각 렌더링 노드들이 서로 독립적으로 동작하기 때문에 기존의 볼륨 렌더링 방식과 마찬가지 방식으로 각 렌더링 노드에서 할당된 데이터를 렌더링하면 된다. 이때, 생성된 렌더링 이미지는 최종 이미지가 아닌 중간 단계의 이미지이므로, 최종 컴포지팅 단계에서 사용할 수 있도록 각 픽셀의 투명도(alpha) 및 깊이(depth) 정보를 버리지 않고 함께 출력하여야 한다.

Sort-Last 렌더링 방식은 이미지를 컴포지션하는 단계에서 프로세서간의 통신량에 따라 성능에 큰 차이를 보이므로, 일반적으로는 이미지 해상도가 커질수록 컴포지션 시간에 큰 차이를 보이게 된다. 또한, 각 노드간 네트워크는 모든 렌더링 노드에서 생성된 픽셀 데이터를 처리할 수 있는 대역폭을 제공해야 하며, 한 노드에 컴포지션 작업이 집중되지 않도록 작업을 분배해야 한다. 이러한 점을 고려하여, 최종 컴포지팅은 이진치환(binaryswap) 컴포지팅 방법을 이용하였다. 이진치환 컴포지팅 방법은 SLIC고 함께 현존하는 Sort-Last 컴포지팅 방법 중 자원의 분배가 가장 효율적이고 속도 역시 가장 빠른 방법으로 알려져 있다[7,8,9].

5. 구현 결과

본 논문에서 렌더링한 VKH 데이터는 여러 데이터들 중 CT 데이터를 사용하였으며 총 512MB로 8개의 64MB 크기로 나뉘어져서 렌더링되었다. 구현에 사용된 그래픽스 하드웨어는 NVIDIA의 Quadro FX3000G로 그래픽스 메모리는 256MB이지만 퐁 쉐이딩을 위한 법선 벡터 데이터가 실제 볼륨 데이터의 3배 크기를 필요로 하기 때문에 가용한 메모리 크기 256MB의 1/4인 64MB 크기만큼만 렌더링에 사용하였다. 각 렌더링 노드(GPU)에서는 미리 할당된 데이터를 일반적인 GPU 기반 볼륨 렌더러와 마찬가지 방식으로 렌더링하였는데, 이때 시점의 위치 및 방향, 광원의 위치 및 방향 등 렌더링 데이터를 제외한 다른 조건들은 모두 일치시켜야 한다. 이렇게 각 렌더링 노드에서 데이터를 렌더링하면 glReadPixels 함수를 이용하여 RGBA 및 깊이 값을 읽어 들이고 이를 이진치환 컴포지션 기법을 이용하여 분할, 치환 과정을 거쳐 컴포지팅하였다. 여기서 Binaryswap 기법은 MPICH를 이용하여 구현하였고, glDrawPixels 함수를 통해 블렌딩하였으며 back-to-front 순서를 유지하도록 하였다. 총 8개의 노드를 이용하여 컴포지팅하였으므로 3번의 이진치환 과정만 이루어졌으며 이는 전체 렌더링 시간에 별다른 영향을 미치지 않았다. 이렇게 렌더링되 결과는 4,200X2,100의 해상도를 지원하는 타일형 디스플레이에 가시화하도록 하였다. 해상도 별 렌더링 성능 및 가시화한 영상은 표 2 및 그림 2와 같다.

6. 결론

본 논문에서는 GPU를 이용한 볼륨 렌더링 기법을 활용해서 GPU 클러스터에서 병렬 렌더링을 수행하였다. 한정된 크기의 데이터만 렌더링이 가능한 GPU들을 이용하여 대용량 데이터를 인터렉티브한 속도로 렌더링하는 것이 가능했다. 그러나 외부 장치와의 속도가 매우 느린 GPU의 특성상 극단적인 Sort-Last 방식의 볼륨 렌더링만 가능하였다. PCI-Express 방식의 그래픽스 하드웨어에서는 GPU와 메인 메모리등의 외부 장치와의 속도가 획기적으로 개선되었으므로 이러한 장비를 이용한 클러스터에서는 보다 향상된 병렬 렌더링 알고리즘을 사용할 수 있을 것으로 판단된다. 또한 렌더링 노드 수를 늘릴수록 콤포지션에 걸리는 시간이 점점 느려지므로 이를 보완할 방법으로 GPU 기반의 실시간 압축 기법이 요구되어진다.

표 2. 해상도 별 렌더링 속도 (Frame / sec)

해상도	256X256	512X512	4,200X2,100
속도	12.12	3.01	0.98

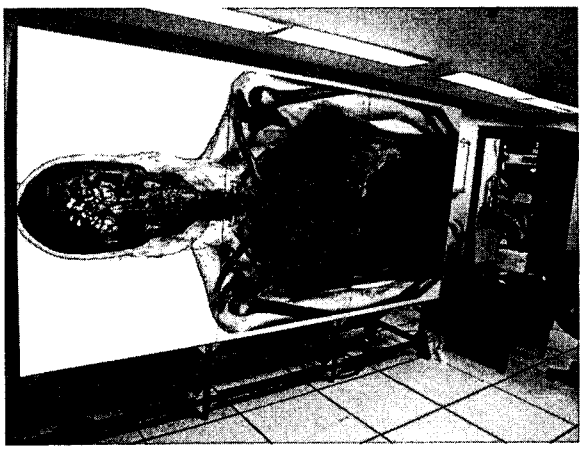

그림 2. 렌더링 영상

7. 참고문헌

[1] Chung, Kim, Hwang, Cho, Park, "Visible Korean Human: Another Trial for Making Serially Sectioned Images", Proceedings of the 10th World Congress on Medical Informatics p. 943 MedInfo 2001

[2] NVIDIA, "NVIDIA GPU Programming Guide version 2.2.0", 2005

[3] Wylie, Pavlakos, Lewis, Moreland, "Scalable Rendering on PC Clusters", Proceeding on IEEE Computer Graphics and Application, July 2001

[4] Ma, Painter, Hansen, "Parallel volume rendering using Binary-Swap Composition", Proceeding on IEEE Computer Graphics and Applications, July 1994

[5] Akeley, "RealityEngine Graphics", Proceeding on SIGGRAPH 93 Conference, 1993.

[6] Engel, Kraus, Ertl, "High-Quality Pre-Integrated Volume Rendering using Hardware-Accelerated Pixel Shading", Proceeding on Eurographics/SIGGRAPH Workshop on Graphics Hardware, 2001.

[7] Stompel, Ma, Lum, "SLIC: Scheduled Linea Image Compositing for Parallel Volueme Rendering", Proceeding on IEEE Symposium on Parallel and Large-Data Visualization and Graphics, 2003

[8] Humphreys, Buck, Eldridge, Hanrahan, "Distributed Rendering for Scalable Displays", Proceeding on Super Computing, 2000

[9] Kniss, McCormick, McPherson, Ahrens, Painter, Keahey, Hansen, "Interactive Texture-Based Volume Rendering for Large Data Sets", Proceeding on IEEE Computer Grahics and Application, July 2001