Multi-Textures를 이용한 Volume Rendering

박채영 °, 이병일, 최흥국
인제대학교 전산학과

Volume Rendering Using Multi-Textures

Jae-Young Park °, Byeong-Il Lee, Heung-Kook Choi
Dept, of Computer Science Inje University

요 약

직접 volume rendering 방식에서 좋은 해상도의 이미지를 얻기 위해서 계산되는 많은 trilinear interpolation은 고성능 그래픽 워크스테이션이나 특별한 목적의 하드웨어 사양을 요구하며 제한적으로 구현이 되고 있다. 따라서 본 논문에서는 PC 그래픽 하드웨어 상에서 2D-Texture를 이용하여 volume rendering을 MRI head set 영상을 적용하여 구현해 보았다. 또한 최근에 지원되는 PC 그래픽 보드의 multi-texturing성능을 이용하여 volume rendering 할 수 있는 방법을 보여준다. 이러한 OpenGL 확장 기능을 이용하여 픽셀 연산과 rendering 성능을 PC 기반에서 향상 시켜 보았다.

Abstract

Direct volume rendering has yet been restricted to high-end graphic workstations and special-purpose hardware, due to the large amount of trilinear interpolation, that are necessary to obtain high image quality. In this paper, we implemented the volume rendering techniques using the 2D-texture at the environment of standard PC hardware. In addition, we show how multi-texturing capabilities of modern PC graphics board are enable to volume rendering. Besides using extended OpenGL function, we improved pixel operations and rendering capacity.

1. 서론

Volume rendering은 오브젝트 내부를 가시화를 할 수 있는 장점 때문에 기계, 과학, 의료 분야 등에서 많이 사용되고 응용 프로그램 개발도 최근 들어서 활발히 진행이 되고 있다. 하지만 trilinear interpolation 과정에서 매우 많은 연산을 요구하므로 고성능 워크스테이션이나 매우 한정된 그래픽 하드웨어서만 구현이 되고 있다. 그러나 최근 PC기반의 그래픽 가속 카드들이 2D-texture 기반으로 semi-transparent한 오브젝트를 표현 할 수 있는 기능을 하드웨어적으로 지원을 하고 있다[1]. 제조 업체마다 방식이 조금씩 다르지만 일반적으로 OpenGL 1.2 방식을 표준으로 정해놓고 있으며 일부 제품들은 OpenGL 1.2 확장 함수를 지원하여 보다 효율적으로 volume rendering을 지원하고 있다. Volume rendering은 non-transparent한 오브젝트와 semi-transparent를 어떻게 하면 동시에 효율적으로 표현을 할 수 있느냐가 가장 큰 문제인데, 최근의 PC 그래픽 카드들이 2D-texture 기반으

로 지원을 잘 하고 있다. 따라서 2D-texture 기반으로 volume rendering을 할 수 있도록 volume data의 기본이 되는 2D 슬라이스 영상들을 하나의 trxture로 표현을 하고 PC 그래픽 주사 방식에 적용이 되도록 폴린곤 단위로 매핑 시킬 필요가 있다. 본 논문에서는 OpenGL 1.2 함수들을 이용하여 2D 슬라이스 영상들을 texture로 표현을 하고 하드웨어 가속이 지원이 되도록 PC 그래픽 가속 카드들이 제공하고 있는 multi-texture 방식으로 조합 시켜 보았다.

이러한 방식으로 2D 슬라이스 영상들을 조합을 하면 trilinear interpolation이 bilinear interpolation방식으로 하드웨어 기반으로 수행이 되므로 보다 빠르고 정확하게 렌더링을 할 수 있다. 본 논문에서 제시한 방법이 얼마나 좋은 결과를 가지고 왔는지 서로 다른 그래픽 카드를 사용하여 렌더링 성능을 테스트하여 보았다.

2. 관련 연구

오브젝트 내부를 가시화 하는 방법에 대한 매우 다양한 어플리케이션 응용 분야가 있다. 최근에는 간접적 방법인 isosurface 추출법과 직접적 방법인 voxel data 방법이 있다[2]. 본 연구에서는 직접적 방법에 대해서 다룬다.

Object-aligned 슬라이스 방법의 기본 아이디어는 Lacroute와 Levoy과 이용했던 bilinear interpolation에 의해 trilinear로 대체하는 것인데, 실제고 구현 시에는 하드웨어를 가속을 지원 받지 못하는 방식이다[3,4]. PC상에서 volume 결과물을 표현하는 기술은 2D-texture mapping 기반하에 이루어지는 것이 요즘 그래픽 하드웨어 방식이다.

3. PC Graphic Hardware

하드웨어 가속 렌더링을 지원하기 위해서 최근 그래픽 카드들은 표준 디스플레이 파이프라인에 따라서 매우 유연성 있는 rasterization을 지원을 한다. Multi-texturing은 OpenGL 1.2에 소개되었던 확장 옵션이며 하나의 폴리곤은 multi-textures에서 얻어진 이미지 정보를 가지도록 해준다.

그림 1처럼 OpenGL 1.2 확장 기능을 지원하는 PC 그래픽 카드들은 multi-stage rasterization방식으로 color, opacity, texture들을 픽셀로 조합해서 semi-transparent한 부분도 렌더링을 지원한다.

그림 1과 같은 방식으로 texture를 조합 할 수 있는 그래픽 카드들은 하드웨어 기반으로 texture를 조합을 하며, 조합을 할 수 없는 그래픽 카드들은 소프트웨어적으로 렌더링을 수행한다. 즉, 그래픽 카드 제조 업체마다 성능과 결과에서 조금씩 차이가 날 수도 있다.

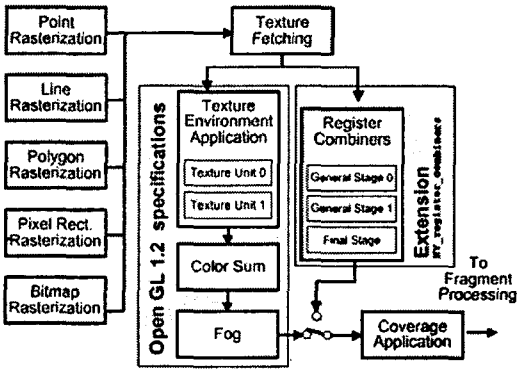

그림 1. OpenGL 확장 모드

4. Mult-Texture 기반의 Volume Rendering

Texture 기반에서 2D 슬라이스의 각각의 픽셀들은 참조해서 interpolation을 적용시키는 것이 아니라, 2D 슬라이스 영상들을 각각의 폴리곤에 매핑을 시켜서 texture기반으로 조합을 하게 된다. 이런 식의 조합은 그림 1처럼 이미 OpenGL 1.2의 확장 모드를 지원하는 PC 그래픽 카드에 의해 하드웨어적으로 제공되고 있다. 따라서 그림 2처럼 슬라이스 S_i와 S_{i+1}은 texture0과 texture1로 명시할 수 있으며 multi-texture로 확장이 되어진다. 그림 2는 입력된 슬라이스들을 texture 모드로 조합을 해서 어떤 랜더링 단계를 거치는지 보여주고 있다.

4.1 Multi-texture interpolation

슬라이스 영상들을 texture 모드로 매핑을 시키기 위해서 다음과 같은 함수를 적용 시켰다.

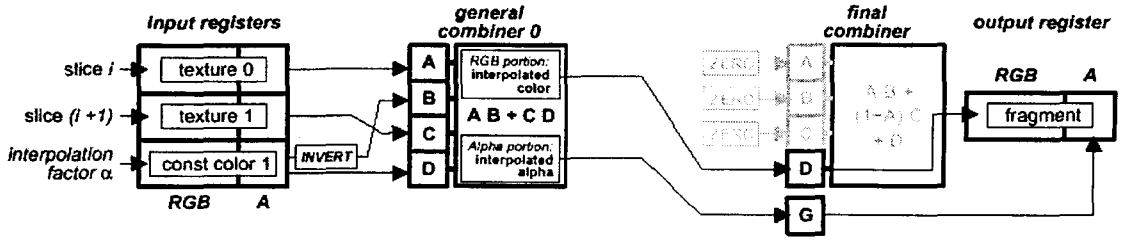

그림 2. 중간 슬라이스 interpolation을 위한 결합 단계

glTexImage2D(..) (1)
glTexParameterf(..) (2)

함수(1)에서는 2D 슬라이스 이미지의 정보를 읽어오고 함수(2)에서는 texture 모드로 매핑을 시켜주며, S_i와 S_{i+1} 사이에 빈 공간이 생기는데 triinterpoation을 통해서 이러한 빈 공간을 적당하게 채워주는 역할을 해서 공간 해상도를 정확하게 표현을 해 주도록 한다.
이렇게 texture 조합 단계에서 얻어진 R, G, B, A 값을 적당하게 조합을 해서 그림 2에서 보듯이 frame buffer로 픽셀 정보를 넘겨주고 화면으로 출력을 해주면 원하는 결과영상이 모니터로 출력이 된다[5,6]. 다음 함수는 이런 단계를 가능하게 해준다.

glBlendFunc(..) (3)
glDrawPixels(..) (4)

함수 (3)은 back-to-front 방식으로 픽셀 정보를 조합을 해주며, 함수 (4)는 이런 픽셀들을 출력하기 위해서 frame buffer로 넣어준다.

5. 결 과

Texture 방식과 OpenGL 1.2 확장 모드로 프로그램을 구현했기 때문에 PC 기반에서 하드웨어 가속 모드로 렌더링이 가능하였다. Volume data를 OpenGL 기본 함수를 사용하며 보다 쉽게 다룰 수 있도록 하여 프로그램의 유연성을 제공하였으며 렌더링한 결과 영상을 바로 저장하도록 하였다.

5.1 Datasets

실험에 사용한 데이터는 다음과 같다.

-Data : MRI-HEAD 슬라이스
-Images : 110장
-Size : 128 x 128

5.2 결과 영상

OpenGL 방식에 따라서 volume data를 3차원 공간에 투영 시켰기 때문에 간단한 마우스 조작만으로 volume data의 매우 쉽게 다른 측면을 렌더링 할 수 있었다.
다음 그림 3은 여러 가지 관찰자 지점에서 렌더링한 결과 이미지를 보여준다.

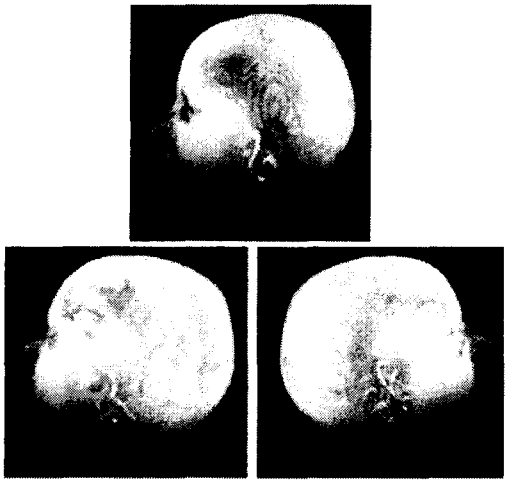

그림 3. 관찰자 시점에 따른 결과 영상

5.3 성능 테스트

PC 그래픽 가속 카드에 따라서 얼마나 영향을 받았는지 다름 그림 4와 같이 테스트하여 보았다. 성능 테스트를 하기 위해서 Windows 2000에서 OpenGL 가속을 하드웨어 모드에서 지원하는 카드와 소프트웨어 모드에서 지원하는 카드를 선택

하였다.

FireGL 1000Pro는 소프트웨어 모드로 가속을 지원하는 카드이며 S3 Savage4와 Geforce 256은 하드웨어 모드로 렌더링을 지원한다.

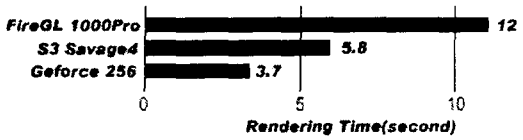

그림 4. OpenGL 가속 카드에 따른 성능 비교

그림 4에서 보듯이 하드웨어 가속에 따라서 다양한 결과를 보여주고 있는데, 하드웨어 모드로 OpenGL 1.2의 multi-textures가속을 지원하는 카드들이 더 좋은 성능을 보여주는 것을 알 수 있다. 즉 PC 그래픽 카드의 성능만 좋으면 전체적으로 volume rendering의 성능을 향상시킬 수 있으며, 하드웨어에 융통성 있게 대처를 할 수 있다.

6. 결 론

본 논문에서는 PC 그래픽 시스템 환경에서 효율적으로 volume rendering을 할 수 있는 방법을 살펴보았고, OpenGL 1.2 확장 함수를 이용하여 렌더링 하여 보았다.

Volume rendering은 volume data와 하드웨어에 매우 종속적이기 때문에 렌더링 하고자 하는 volume data와 시스템 환경에 따라서 적절하게 triinterpolation 해야만 한다. 따라서 PC 기반에서 경제적으로 렌더링 할 수 있는 방법이 계속 연구되고 있으며, 본 논문에서도 OpenGL를 이용하여 PC 기반에서 경제적이고 효율적으로 하드웨어 가속이 지원 되도록 하여 volume rendering을 구현하여 보았고 렌더링 성능도 향상시켜 보았다. 또한 OpenGL 1.2 확장 모드를 이용하여 2차원 슬라이스 영상들을 multi-textures모드로 조합을 하기 때문에 최근 PC 그래픽 가속 카드에서 유용하게 이용을 할 수 있다.

Texture방식으로 trilinear interpolation을 하면 opacity를 고려할 수 없기 때문에 gradient를 적절하게 이용을 할 수 없고, 하드웨어 기반으로 픽셀 정보가 처리가 되므로 다양한 transfer function를 적용 할 수가 없다. 따라서 오브젝트 내부에 따라서 다양한 transfer function을 적용시키는 연구가 필요하다.

[참고문헌]

[1] "Volume Visualization with Texture", SGI, October, 1999.

[2] Lorensen, W.E. and Cline, H.E., "Marching Cubes: a high resolution 3D surface reconstruction algorithm," Computer Graphics, Vol. 21, No. 4, pp 163-169.

[3] Marc Levoy, University of North Caraolina, "Volume Rendering", IEEE May, 1998.

[4] Marc Levoy, "Display of Surface from Volume Data", IEEE Computer Graphics and Applications, Vol. 8, No. 5, pp.29-37 May 1988.

[5] OpenGL Architecture Review Board. OpenGL reference Manual. Addison - Wesley Press, 2nd.. 1997.

[6] Donald Hearn, M.Pauline Baker. "Computer Graphics". Prentice-Hall Inc. 1997.