계층적 자료구조와 그래픽스 하드웨어를 이용한 적응적 메쉬 세분화 데이타의 대화식 가시화

(Interactive Visualization Technique for Adaptive Mesh Refinement Data Using Hierarchical Data Structures and Graphics Hardware)

박 상 훈 † Chandrajit Bajaj††
(Sanghun Park) (Chandrajit Bajaj)

요 약 적응적 메쉬 세분화(AMR)는 여러 과학과 공학 분야에서 이용되는 보편적인 계산 시뮬레이션 기법이다. AMR 데이타가 계층적인 다중해상도 데이타 구조로 이뤄져 있음에도 불구하고, 어떤 적절한 자료구조로의 변형 없이, 이 데이타를 광선추적법이나 스플래팅과 같은 전통적인 볼륨 가시화 알고리즘들을 이용하여 가시화하는 것은 불가능하다. 본 논문에서는 AMR 데이타로부터 생성된 k-d 트리와 팔진트리를 이용하는 계층적 다중해상도 스플래팅에 대해 설명한다. 이 기법은 최신의 범용 PC 그래픽스 하드웨어를 이용하여 AMR 데이타의 가시화를 구현하는데 적합하다. 대화식으로 변환함수와 뷰잉 / 렌더링 파라메터를 설정할 수 있는 기능을 제공하는 사용자 인터페이스에 대해서도 설명한다. nVIDIA GeForce3 그래픽스 카드를 내장한 범용의 PC를 이용해 얻은 실험 결과로부터, 제안된 기법을 이용해 AMR 데이타를 대화식으로(초당 20프레임 이상의 속도로) 렌더링할 수 있음을 보인다. 본 기법은 시간 가변 AMR 데이타의 병렬 렌더링에도 쉽게 적용될 수 있을 것이다.

키워드 : AMR, K-d 트리, 팔진트리, 스플래팅, 텍스춰 매핑, 그래픽스 하드웨어, 과학적 가시화

Abstract Adaptive mesh refinement(AMR) is one of the popular computational simulation techniques used in various scientific and engineering fields. Although AMR data is organized in a hierarchical multi-resolution data structure, traditional volume visualization algorithms such as ray-casting and splatting cannot handle the form without converting it to a sophisticated data structure. In this paper, we present a hierarchical multi-resolution splatting technique using k-d trees and octrees for AMR data that is suitable for implementation on the latest consumer PC graphics hardware. We describe a graphical user interface to set transfer function and viewing / rendering parameters interactively. Experimental results obtained on a general purpose PC equipped with an nVIDIA GeForce3 card are presented to demonstrate that the proposed techniques can interactively render AMR data(over 20 frames per second). Our scheme can easily be applied to parallel rendering of time-varying AMR data.

Key words : AMR, K-d trees, Octrees, Splatting, Texture mapping, Graphics hardware, Scientific visualization

1. 서 론

컴퓨터 그래픽스의 세부 분야인 과학적 가시화(scientific visualization)는 물리학, 천문학, 의학, 지구과학, 수학, 유체역학과 같은 과학 분야에서 생성된 3차원 이상의 구조적(structured) 또는 비구조적(unstructured) 볼륨 형태의 데이타를 3차원 컴퓨터 그래픽스의 다양한 렌더링 기술을 이용하여 가시화함으로써 데이타에 내재된 수치 정보를 분석하고 이해하는 과정이다. 가시화의 대상이 되는 볼륨 데이타들은 시뮬레이션의 결과로 생성되거나 특수한 장치를 이용해 촬영한 것들이 대부분이다. 가장 대표적인 3차원 볼륨 데이타의 예로는

· 본 연구는 2002학년도 대구가톨릭대학교 연구비 지원에 의한 것임
† 종신회원 : 대구가톨릭대학교 컴퓨터정보통신공학부 교수
mshpark@cu.ac.kr
†† 비 회 원 : Professor of Computer Sciences, The University of Taxas at Austin
bajaj@cs.utexas.edu
논문접수 : 2003년 8월 29일
심사완료 : 2004년 3월 19일

CT(Computed Tomography), MRI(Magnetic Resonance Imaging), PET(Positron Emission Tomography) 등과 같은 구조적 볼륨 형태의 의료 영상 데이타와 자연현상을 수학적으로 모델링하고 이로부터 얻어진 계산 결과를 담고 있는 시뮬레이션 데이타가 있다. 함수값 $f(i,j,k)$로 구성된 3차원 형태의 볼륨 데이타를 효과적으로 분석하기 위한 다양한 시도들이 과학적 가시화 분야에서 최근 수 년 동안 계속되어 왔으며 지속적으로 새로운 기술들이 개발되고 있다.

가시화 기술의 발전과 컴퓨터 기술의 성장에 발맞추어, 입력 볼륨 데이타의 크기가 매우 방대해지고 있을 뿐만 아니라, 여러 가지 다양한 형태를 갖는 볼륨 데이타들이 생성되고 있다. 우선, 시간의 흐름에 따른 변화를 관찰하기 위한 시간 가변 볼륨 데이타(time-varying volume data)가 생성되었는데, 이것은 각 시간의 흐름에 따른 3차원 볼륨 데이타들의 리스트로 구성되며 최종적으로 4차원 볼륨 데이타의 형태를 갖는다. 그리고, 기존의 일반적인 볼륨 데이타들이(특히, 의료 데이타들이) 동일한 간격의 격자 구조(uniform structured)에 대한 함수값들로 구성되는데 비해, 최근에는 데이타의 특징을 잘 표현하기 위해 전체 격자에 레벨을 부여하고 세밀한 표현이 요구되는 특정 영역에 대해서는 레벨 격자를 더욱 세분화하는 적응적 메쉬 세분화(AMR: Adaptive Mesh Refinement) 데이타가 소개되었다. 기존의 단순한 형태의 볼륨 데이타에 대해서는 이미 다양한 가시화 방법들이 개발되어 효과적인 렌더링이 가능하지만, 시간 가변 볼륨 데이타나 AMR 데이타와 같이 특별한 구조와 형태를 갖는 볼륨 데이타들에 대해서는 전통적인 가시화 방법들과는 다른 새로운 기법이 요구된다. 본 논문에서는 최근에 가시화 분야에서 많은 관심을 끌고 있는 시간 가변 AMR 데이타를 대화식으로(interactively) 렌더링 하기 위한 새로운 자료구조와 효과적인 가시화 기법에 대해 설명한다.

AMR은 편미분 방정식(PDE: Partial Differential Equations)의 수치 시뮬레이션의 효율성을 향상시키기 위한 목적으로 개발된 계산 기법 가운데 하나이다. 1980년대에 Berger와 Oliger가 기체 역학을 시뮬레이션하기 위해 AMR을 개발한 후에[1], 이 기법은 계산 물리를 포함한 여러 자연과학 분야뿐만 아니라 다양한 공학 분야에서 널리 이용되고 있다. 실험 대상의 특징을 잘 표현하기 위해 섬세하고 중요한 변화가 요구되는 계산 공간은 (지역적 또는 시간적 계산 공간 모두에 대해) 높은 해상도로 세분화를 하고 관심이 적은 공간은 낮은 해상도의 표현을 유지하는 것이 AMR의 기본 아이디어이다. AMR 기법은 많은 편미분 방정식을 풀기 위해 요구되는 계산 비용과 저장 비용을 줄이기 위한 목적으로, 그리고 대기 모델링(global atmospheric modeling), 수치 천문학(numerical cosmology)과 같은 다양한 공학적 응용 분야에서 성공적으로 이용되어 왔다. 실제로, Bryan은 AMR의 변형된 접근 방법이 천문학 연구에서 어떻게 적용될 수 있는지를 보였다[2]. 그림 1은 AMR 데이타의 이해를 돕기 위한 예로서, 2차원 단면에서 AMR 데이타의 구조를 보여주고 있다. 그림을 통해 확인할 수 있듯이 섬세한 표현이 필요한 부분은 고해상도로, 그렇지 않은 부분은 저해상도로 데이타의 함수값을 저장하고 있다.

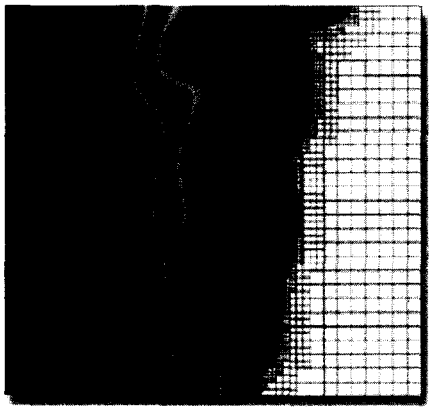


그림 1 AMR 데이타의 예

본 논문에서 실험을 위해 사용한 시간가변 AMR 데이타는 각 시간스텝을 독립된 파일로 저장한다. 그림 2는 여러 시간스텝 가운데 하나의 스텝에 해당하는 AMR 데이타 파일의 구체적인 형식을 보여준다. 각 행은 격자좌표(I, J, K), 함수값들(E1, E2, ..., En), 세분화레벨(NL)로 이뤄져 있다. 정규격자를 갖는 일반적인 볼륨 데이타 파일들은 형태가 이미 정해져 있기 때문에 격자좌표나 세분화레벨의 기록없이 함수값들만 저장하면 되지만, AMR 데이타의 경우에는 이러한 부수적인 정보들이 모두 저장되어야 한다. 따라서 일반 볼륨데이타에 비해 파일의 크기가 더 증가하게 된다.

AMR 데이타가 계층적인 다중해상도(hierarchical multi-resolution) 구조를 갖고 있음에도 불구하고, 간단한 정규 격자 형태를 갖는 볼륨 데이타의 렌더링을 위해 개발된 기존의 전통적인 볼륨 가시화 기법들을 그대로 AMR 데이타에 적용하는 것은 불가능하다. 그리고, 최근까지 AMR 데이타의 가시화에 대해서 그리 많은 연구 결과들이 발표되지 않았다. Norman 등은 큰 단위의 AMR 시뮬레이션으로부터 생성된 데이타를 처리할 때 발생하는 문제점을 언급하고, 이들 데이타를 정렬하고, 조작하고, 가시화하고, 가상 탐색하고, 그리고 원격 서비스하기 위한 해결책을 제시했다[3]. Weber 등은

The variables are given at the nodes of the mesh in float point format. Each line contains:

...
<I> <J> <K> <E0> <E1> ... <En> <NL>
...

where <I>, <J>, and <K> are the nodal coordinates (x, y, z), <Ei> is the i-th variable (n=14), and <NL> is the refinement level in the grid. Interesting variables include:

E0 = Energy Density
(E1, E2, E3) = (Mass Density)*(Velocity_X, Velocity_Y, Velocity_Z)
E4 = Mass Density
E6 = Electron Density

The refinement of level i+1 has grid spacing 1/2 of level i (like simple octree, with coastest level i=1), The nodal coordinates range over (Nx, Ny, Nz), which are the dimensions of the highest refinement level (NLmax). The grid spacing of the lowest level is therefore 2^(NLmax-1) in the coordinate system.

그림 2 실험에 사용된 AMR 데이타 파일의 형식

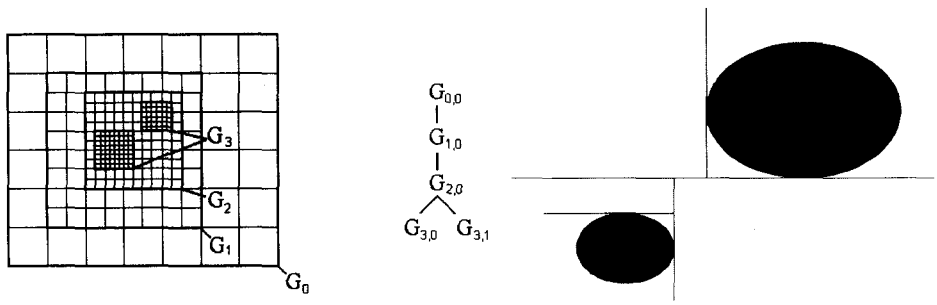
AMR 데이타로부터 깨짐없는 등가면(crack-free iso-surface)을 추출하는 방법을 소개하였다[4]. 그들은 또한 [5]에서 미리 보기 기능을 지원하기 위한 하드웨어 가속 렌더링 인터페이스(hardware-accelerated rendering interface)와 셀-사영 기반 점진적 세분화 렌더링 기법(cell-projection based progressive refinement rendering scheme)을 설명했으며, 또 다른 논문 [6]에서, 점진적 셀-사영 접근 방법(progressive cell-projection approach)과 레벨 의존적인 변환 함수(level-dependent transfer function)를 이용해 AMR 데이타를 렌더링 했다. 물론 이러한 방법들을 이용하여 고화질의 영상을 생성할 수는 있으나, 80×32×32의 루트 격자 해상도를 갖고 세 개의 레벨 계층을 갖는 비교적 작은 AMR 데이타로부터 하나의 영상을 렌더링하는 경우에도 너무 많은 시간(약 23~115초)이 걸린다는 문제를 갖고 있다. AMR 자료구조를 이용한 또 다른 접근 방법이 Kähler 등에 의해 소개되었는데[7], 그들은 방대한 크기의 정규 격자 3차원 스파스 스칼라 볼륨 데이타(sparse scalar volume data)로부터 AMR 자료구조를 계산하고 3차원 텍스쳐 기반 렌더링을 이용하여 가시화를 수행하였다.

본 논문에서는 AMR 데이타의 계층적인 다중해상도 표현 형태로부터 k-d 트리와 팔진트리(octree)와 같이 유용한 자료구조를 추출하고, 그래픽스 하드웨어의 가속 기능을 이용한 스플래팅(splatting) 기법으로 주어진 데이타를 대화식으로 렌더링하는 기법에 대해서 설명한다 [8]. 또한, 렌더링 결과 영상에 영향을 주는 다양한 파라메터들을 사용자가 대화식으로 설정할 수 있는 인터페이스도 구현하였으며, nVIDIA GeForce3 그래픽스 카드와 Pentium III 프로세서를 장착한 범용 PC 상에서 이 기법을 이용해 초당 20프레임 이상의 속도로 시간의 흐름에 따른 시뮬레이션 결과를 가시화 할 수 있다. 이와같은 연구 결과들은 이후의 장에서 자세하게 설명할 것이다. 본 논문의 주된 연구 내용은 다음과 같이 정리될 수 있다:

- AMR 데이타의 계층적인 저장과 빠른 탐색(search)을 위한 k-d 트리와 팔진트리 자료구조의 설계
- AMR 데이타의 대화식 렌더링을 위한 그래픽스 하드웨어 가속 스플래팅 알고리즘의 개발
- 변환함수(transfer function)와 뷰잉(viewing) / 렌더링 파라메터를 대화식으로 설정하기 위한 그래픽스 사용자 인터페이스(GUI: Graphical User Interface)의 구현

2절에서는 구현된 AMR 데이타 가시화 기법에서 사용된 자료구조와 하드웨어 가속 스플래팅 알고리즘에 대해 자세히 설명한다. 3절에서는 실험을 통해 얻은 가시화 결과 영상과 렌더링 성능에 대해 분석한다. 그리고, 마지막 4장에서는 결론과 향후 연구 방향을 제시하고 논문을 정리한다.

2. 가시화 기법의 설계



(a) AMR 데이타 격자의 계층구조 (b) K-d 트리 분할의 예

그림 3 AMR 데이타의 계층구조와 k-d 트리

2.1 계층적 자료구조의 이용

K-d 트리(k-dimensional tree)는 다차원 공간을 각 주축(main axis: 3차원의 경우는 xyz축)에 수직인 평면들로 재귀적으로(recursively) 분할하는 자료구조이다. 전산학에서는 이러한 자료구조를 어떤 영역에 대한 효과적인 탐색을 위해 이용한다. 즉, k-d 트리를 이용하여 어떤 공간에서 평면들로 둘러싸인 제한된 영역(이를 블릭(blick)이라 부른다)내에 속하는 복셀(voxel)들의 집합을 효과적으로 찾을 수 있다. 실제로, 주어진 복셀들을 표현하는 하나의 k-d 트리를 이용하여 원하는 복셀을 탐색하기 위해 필요한 계산 비용은 $O(\sqrt{n}+k)$ 이다(여기서, n은 전체 복셀의 개수이고, k는 탐색된 복셀의 개수이다).

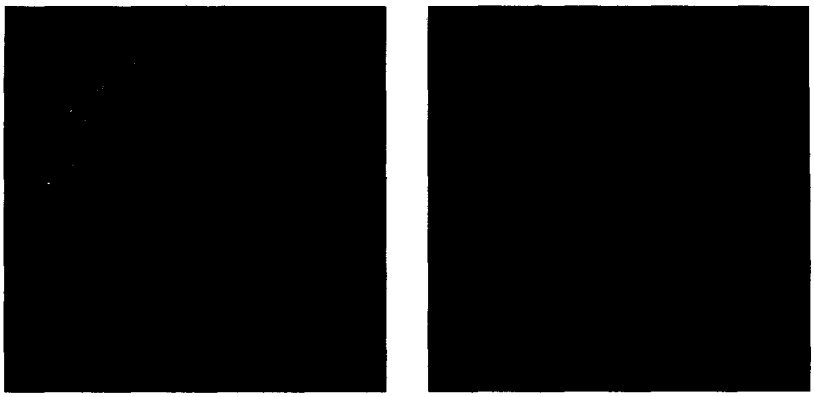
AMR 시뮬레이션 알고리즘은 격자 계층(grid hierarchy) 자료구조를 생성하고(임의의 구조와 깊이를 갖는 트리 형태를 갖는다), 트리내의 각 노드(node)들과 리프(leaf)들은 3차원 격자와 연관되어 있다(그림 3(a) 참조). 이러한 자료구조는 다양한 모양, 크기, 해상도를 갖기 때문에, 효과적인 가시화를 위해서는 주어진 데이타를 유용한 자료구조로 변환해야 한다. 주어진 AMR 데이타를 그림 3(b)와 같은 k-d 트리 구조로 분할하기 위해서, 입력 AMR 데이타가 원래부터 갖고 있는 격자의 계층적인 자료구조를 고려한 변형된 k-d 트리 생성 알고리즘이 설계되어야 한다.

본 연구의 대상이 되는 시간가변 AMR 데이타는 $\bigcup(f_{t,l,v}(i,j,k))$로 표현된다. 여기서, t는 시간가변 AMR 데이타의 시간스텝(time-step)을 나타내고, l은 세분화 레벨, 그리고 v는 함수값들에 대한 인덱스를 의미한다. 이러한 형태의 AMR 데이타를 k-d 트리 구조로 변환하기 위한 첫 번째 단계는 AMR 데이타 공간에서 각 그룹을 둘러싸는 최소 영역 상자(minimum bounding box)를 결정하는 것이다. 각 그룹에 속하는 복셀들은 현재 레벨의 공간 해상도내에서 연결되고, 각 그룹들은 AMR 데이타의 여러 레벨을 포함할 수 있다. 다음 단계는 영역 상자를 변형된 k-d 트리 알고리즘을 이용하여 블릭들의 집합으로 분할하는 것이다. 여기서 생성된 블릭들은 각 레벨에 대한 실제 함수값의 집합을 가리키는 포인터를 갖는다.

앞에서 언급한 바와 같이, k-d 트리는 렌더링을 위해서 뿐만 아니라, AMR 데이타를 계층적으로 정렬하는 데에 유용하게 이용된다. 실제로, AMR 데이타의 원형(raw form)은 하나의 (i,j,k) 복셀 인덱스와 몇몇 함수값 $f_{t,l,v}$, 그리고 하나의 레벨 l로 이루어진 레코드들의 리스트이다(그림 2 참조). 이와 같은 원형 AMR 데이타 구조를 하나의 k-d 트리로 대치하는 것은, 계층적 다중해상도 자료구조의 표현이 가능하고 복셀의 인덱스들을 더 이상 저장할 필요가 없다는 점에서 매우 효과적이다. 그림 4는 실험에 실제로 사용된 AMR 데이타를 k-d 트리 생성 알고리즘을 이용해 블릭들로 분할한 결과를 보여준다. 공간 응집성(spatial coherence)을 이용하기 위해서, 구현된 기법은 비교적 큰 블릭에 대해서 팔진트리 생성 알고리즘을 적용하여 렌더링 속도를 향상시켰다. 팔진트리의 생성과 이용에 관해서는 다음 절에서 구체적으로 설명한다.

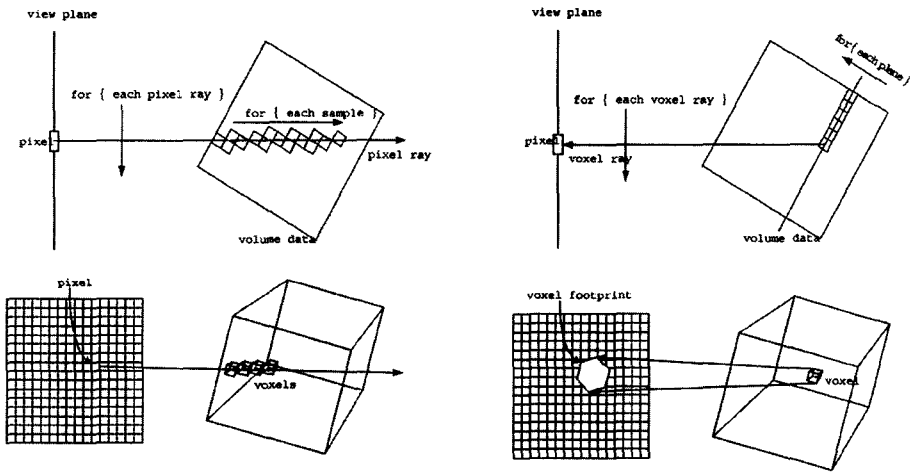
2.2 하드웨어 가속 스플래팅

스플래팅은 객체 공간 직접 볼륨 렌더링 알고리즘(object space direct volume rendering algorithm) 가운데 하나로서 고화질의 영상을 생성하는 기법으로 알려져 있다[9]. 하나의 복셀의 기여도(contribution)은 영상 평면으로 직접 매핑되고, 이 때 함수값에 대한 보간(interpolation) 계산이 필요하지 않고, 단지 관심이 있는 복셀들만 3차원 커널에 의해 가중치를 갖고 표현되기 때문에 일반적으로 광선추적(ray-casting) 알고리즘보다 빠르게 영상을 만들 수 있다. 특히, AMR 데이타가 기존의 데이타와 다른 형태의 다중해상도 계층 구조를 갖기 때문에, 가시화 기법으로 광선추적법을 사용하



(a) 임의의 위치에서 바라 본 영상 (b) 수직인 위치에서 바라 본 영상

그림 4 3차원 공간에서 AMR 데이타의 k-d 트리 분할



(a) 광선추적법 (b) 스플래팅

그림 5 광선추적법과 스프래팅 기법의 비교

기 보다는 스플래팅을 이용하는 것이 여러 가지 면에서 효과적이다. 그림 5를 통해 확인할 수 있듯이, 광선추적법은 각 픽셀에서 출발한 광선과 교차하는 복셀들을 찾아야만 하는데 k-d 트리 자료구조로부터 이런 복셀들을 찾아내는 것보다는, 해당 복셀 또는 블릭이 영상 평면상의 어디에 사영될 것인지를 찾는 스프래팅 기법의 계산이 훨씬 간단하다.

Laur 등은 계층적인 자료구조를 만들고 각 계층 사이의 오차를 고려하면서 점진적인 스플래팅을 수행하는 기법을 개발하였고[10], Bajaj 등은 3차원 이상의 고차원 볼륨 데이타를 가시화하기 위한 스플래팅 기법에 대한 결과를 발표하였다[11]. 이러한 기법들은 다양한 크기의 풋프린트(footprint)를 이용하여 서로 다른 레벨의 볼륨 데이타를 렌더링 한다. 또한, 풋프린트를 다각형에 사영시킴으로써, OpenGL 2D 텍스춰 매핑 하드웨어의 가속 기능을 이용할 수 있다. 실제로, Crawfis 등은 그래픽스 하드웨어를 이용하여 스프래팅을 수행하는 기법을 제안하였다[12]. 이 기법은 풋프린트 테이블을 리샘플링할 때, 그리고 프레임 버퍼로 합성을 할 때 발생하는 복잡한 계산으로부터 CPU의 부담을 줄이도록 설계되었다. 그 외에도, LarMa 등은 일반적인 3차원 볼륨 데이타를 텍스춰 하드웨어의 가속 기능을 이용하면서 다해상도로 가시화할 수 있는 기법을 소개하였다[13].

본 논문의 알고리즘에서는 사용자에 의해 정의된 변환함수와 선택된 함수값 범위에 속한 영역에 대한 렌더링을 위해 스플래팅 기법을 이용하였다. K-d 트리를 이용함으로써 블릭에 대한 빠른 탐색을 구현할 수 있었고, k-d 트리의 각 노드에 팔진트리 구조를 추가로 할

당함으로써 속도를 더욱 최적화시킬 수 있었다. 결국, 본 기법에서 각 블릭은 복셀에 대한 팔진트리로 표현되고, 각 복셀은 실세계 좌표계(world coordinates)로부터 화면 좌표계(screen coordinates)로의 변환을 통해 사영된다(그림 5(b) 참조).

각 블릭과 팔진트리의 효과적인 표현을 위해 등가코드(isovalue code)라는 것을 정의하고 이를 생성·저장하는 방법을 사용하였는데, 이 등가코드를 이용함으로써 연관된 부분 볼륨들만이 탐색 알고리즘의 고려 대상이 된다. 등가코드는 32비트로 이루어지며, 각 비트들은 어떤 범위의 함수값이 현재 부분 볼륨 내에 포함되어 있는지 또는 그렇지 않은지를 표현한다. 전처리(pre-processing) 과정을 통해, 이러한 이진코드를 생성·저장하게 되며, 팔진트리의 각 레벨에 대해 이를 계산한다. 우선 각 리프 노드에 포함된 모든 복셀들을 방문하여 해당 함수값 범위에 속하는 복셀이 존재하는지를 확인하고, 만일 그렇다면 등가코드의 적절한 비트를 1로 설정한다. 만일 등가코드가 n비트로 구성되고, 전체 볼륨 데이타에 속한 복셀들이 r개의 함수값으로 표현된다면, 등가코드의 각 비트는 $\frac{r}{n}$의 범위를 나타낸다. 일단 팔진트리의 각 리프노드들에 대한 등가코드를 생성하고 나면, 부모노드에 대한 등가코드는 자식노드들의 등가코드들에 대해 OR 연산을 수행함으로써 재귀적으로 계산할 수 있다. 팔진트리의 루트에 대한 등가코드는 그 팔진트리를 포함하는 블릭 전체에 대한 등가코드로서 설정된다. 이러한 수행 과정은 벡터 필드내의 각 함수들에 대해서도 동일하게 적용된다. 본 기법은 볼륨 데이타내에 속한 모든 함수값 범위에 대해 동일한 수의 비트를 할당하는데, 이것은 어떤 특정 영역의 함수값만이 의미있는 데이타임을 미리 정의하지 않고 모든 범위의 함수값들이 동일한 중요도를 가지고 사용자에 의해 가시화될 수 있음을 허용하기 위함이다. OR 연산은 사용자가 관심을 갖는 특정 범위의 함수값이나 중요하게 가시화해야 할 함수값을 사용자 인터페이스를 이용해 선택하는 과정에서도 이용되는데, 이를 통해 탐색코드(search code)를 만들게 된다. 예를 들어, 가시화를 위해 사용자가 선택한 함수값의 범위가 $[a,b]$와 $[c,d]$일 때, 탐색코드는 $[a,b]\bigcup[c,d]$가 된다. 사용자가 현재 설정한 범위의 함수값이 어떤 부분 볼륨내에 존재하는지 그렇지 않은지는 간단한 AND 연산만으로 간단히 수행될 수 있다. 이러한 과정은 블릭 레벨과 팔진트리 레벨에서 모두 수행된다. 해당 부분 볼륨에 대한 최대, 최소값을 유지하는 기존의 팔진트리 자료구조에 비해, 본 논문의 자료구조를 이용해 더 좋은 성능을 얻을 수 있다. 팔진트리를 이용하여 얻을 수 있는 또 하나의 유용함은 복셀들에 대한 빠르고 자연스러운 정렬인데, 이것은 사용자에 의해 결정된 카메라의 관찰방향(viewing direction)으로부터 고화질의 영상을 빠르게 얻기 위한 필수적인 기술이다. 유사한 기법으로, 각 블릭들에 대한 정렬은 k-d 트리를 이용하여 쉽게 계산할 수 있다.

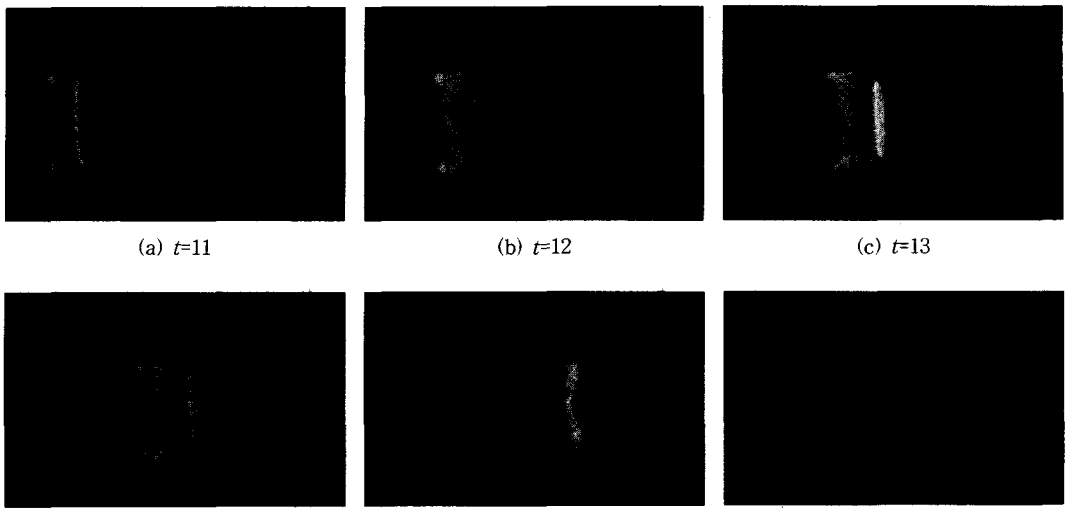
사용자가 인터페이스를 통해 변환함수를 정의하면 룩업테이블(lookup table)로서 컬러테이블이 자동 설정된다. 사용자는 대화식으로 각 함수값마다 다른 불투명도 값(opacity value)을 설정할 수 있으며, 이것은 벡터 데이타에 대해서도 동일한 방법으로 확장 가능하다.

2.3 전체적인 알고리즘

그림 6은 AMR 데이타의 가시화를 위해 설계된 하드웨어 가속 스플래팅 알고리즘의 전체적인 구조를 보여준다. 주어진 AMR 데이타에 대해 k-d 트리와 팔진트리를 생성하는 과정(2번째 행)은 전처리 단계에서 이뤄지게 됨으로, 실제 실시간 렌더링 성능에는 영향을 끼치지 않는다. 일단 k-d 트리가 생성되고 나면, 현재 설정된 카메라의 관찰방향 따른 블릭 리스트 BL이 k-d 트리 방문(traversal)을 통해 효과적으로 결정된다(4번째 행). 물론, 이 과정에서 가시화의 대상이 되는 복셀을 전혀 포함하지 않는 블릭들은 BL에서 제외된다. 어떤 블릭이 랜더링될 것인지 그렇지 않을 것인지에 대한 결

```
1. Load AMR data
2. Create k-d tree and octree data structure
3. Set current transfer function and viewing/rendering parameters, and level of
   detail lod
4. Determine brick list BL according to viewing direction
5. for (Bi in BL) {
6.         Splatting(Bi, lod)
7.         Composite the current partial image
8. }
9. Display final image
```

그림 6 AMR 데이타 가시화를 위한 하드웨어 가속 스플래팅 알고리즘



(a) t=11 (b) t=12 (c) t=13

(d) t=14 (e) t=15 (f) t=16

그림 7 시간 가변 실험 AMR 데이타의 최상위 레벨(l=3)에서의 매스밀도(v=4) 가시화 영상

정은 블릭 전체에 해당하는 등가코드와 탐색코드에 대한 한 번의 AND 연산만으로 수행 가능하다. 그 후, 연관된 블릭들로 이뤄진 정렬 리스트내의 각 블릭 B_i에 대해 스플래팅 기법이 적용되고, 생성된 부분 영상들은 최종 영상에 합성된다(6-7번째 행). BL에 속해 있는 노드들은 여러 레벨에 대응되는 함수값들을 가리키는 포인터를 갖고 있으며, 렌더링되어야 하는 세밀한 정도 *lod*(level of detail)가 선택되면, 이에 따라 적당한 변환함수와 풋프린트 크기가 결정된다. 앞에서 언급한 바와 같이, 알고리즘의 6-7번째 행은 텍스춰 매핑 하드웨어의 가속기능을 이용하여 성능을 향상시켰다.

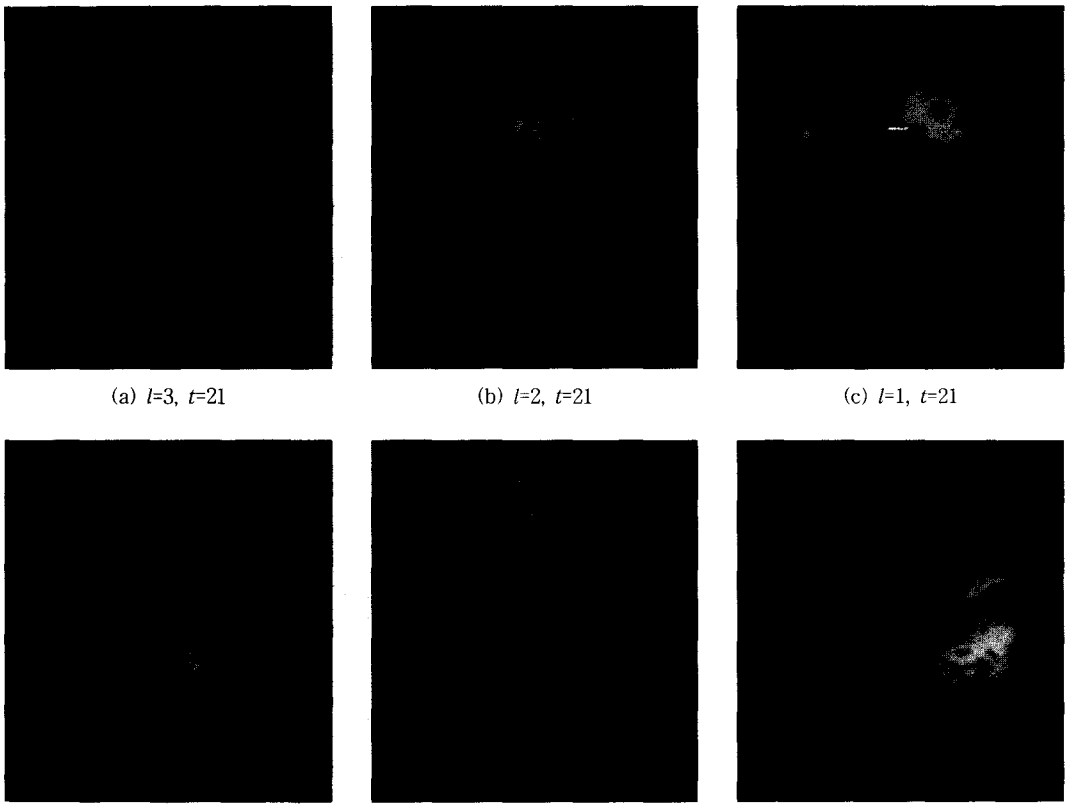
3. 실험결과와 성능분석

본 연구의 목표는 AMR 데이타를 실시간 렌더링에 가까운 초당 20 프레임 정도의 속도로 렌더링하는 것이다. 이러한 가시화 속도를 얻기 위해서는 반드시 고성능의 그래픽스 하드웨어의 가속 기능을 이용해야 한다. 이를 위해 64MB의 텍스춰 메모리를 갖는 nVIDIA GeForce3 그래픽스 카드를 장착하고, 256MB의 주기억장치와 800MHz Intel Pentium III 프로세서를 탑재한 PC를 이용하여 실험을 수행하였다. 제안한 알고리즘의 구현을 위해, OpenGL과 GLUT를 사용하여 프로그램을 개발하였다.

실험에 사용된 시간가변 AMR 데이타는 방사성 제트 충돌 시뮬레이션(simulation of a radiative jet colliding with dense cloud)으로부터 산출된 것이다. 시뮬레이션 결과는 가장 세밀한 격자 해상도로서 64×64×128를 갖고 4개 레벨($0 \le l \le 3$) 계층구조를 갖는 AMR 형식으로 저장되었다. 메쉬의 각 노드들에는 15개 함수값($0 \le v \le 14$)들이 부동소수점 형식으로 기록되어 있다. 실제 구현에서는 주어진 함수값을 [0, 4095]의 정수로 양자화한 데이타를 사용하였다. 실제로, 이 데이타에서 흥미있는 가시화 결과를 얻을 수 있는 함수값들은 에너지밀도(energy density: v=0), 매스밀도(mass density: v=4), 전자밀도(electron density: v=6) 등이다.

그림 7과 8은 실험에 사용된 시간가변 AMR 데이타의 매스밀도와 전자밀도 함수값을 본 논문에서 제안된 기법을 이용하여 각각 가시화한 결과 영상이다. 그림 7는 매스밀도(v=4)의 충돌 과정을 최상위 레벨(l=3)로 시간 t의 변화에 따라($11 \le t \le 16$) 렌더링한 결과 영상이며, 사용자 인터페이스를 통해 미리 설정된 변환함수를 적용하여 가시화한 연속 프레임을 저장한 것이다. 이러한 연속 영상은 거의 실시간으로 렌더링 가능하며, 시간의 변화에 따른 시뮬레이션 결과를 효과적으로 분석할 수 있도록 도와준다.

그림 8의 (a)-(c)는 시간이 고정된 상태에서 전자밀도 데이타를(v=6, t=21) 가시화한 결과로서, 이를 통해 세밀한 정도(lod)의 설정 변화(l=1,2,3)에 따른 화질의 차이를 확인할 수 있다. 물론, 세밀한 정도가 낮은 상태(l=1)에서의 렌더링 속도가 높은 상태(l=3)일 때에 비해 빠르지만 결과 영상의 화질은 낮다. 그리고, 그림 8의 (d)-(f)는 렌더링 / 뷰잉 파라메터, 변환함수, 세밀한 정도 등을 변화시키면서 렌더링한 다양한 다중해상도 결과 영상들을 보여준다.



(a) l=3, t=21 (b) l=2, t=21 (c) l=1, t=21

(d) l=3, t=20 (e) l=3, t=20 (f) l=2, t=18

그림 8 시간 가변 실험 AMR 데이타의 전자밀도(v=6) 가시화 영상

K-d 트리 자료구조의 이용으로 탐색공간(search space)을 효과적으로 제한할 수 있음은 앞에서 설명한 바와 같다. 블럭은 그 안에 포함된 함수값의 범위를 나타내는 등가코드를 갖고 있으며, 이를 통해 시스템의 성능을 향상시킬 수 있었다. 또한, k-d 트리의 각 블럭에 할당된 팔진트리는 탐색공간을 더욱 제한할 수 있도록 만들어준다. 사용하는 팔진트리의 폭이 커질수록 함수값의 범위를 더욱 세밀하게 표현할 수 있고 탐색 공간을 더욱 효과적으로 제한할 수 있으나, 지나치게 세분화된 팔진트리 표현은 메모리 공간을 낭비하는 문제를 발생시킨다. 실험에 의하면, 40,000에서 50,000개 이상의 복셀을 다루는 경우에는 최대폭이 8인 팔진트리를 이용함으로써 훨씬 좋은 성능을 얻을 수 있었고, 이 경우보다 비교적 덜 조밀한 볼륨의 경우에는 크기가 4인 팔진트리를 사용하는 것이 더 효과적이었다. 본 실험에서는 팔진트리를 사용하지 않는 경우와 최대폭이 4, 8, 16인 팔진트리를 이용하는 경우의 성능을 비교, 분석하였다.

표 1의 수치는 각 경우에 따라 탐색된 복셀의 비율과 팔진트리를 사용해서 얻게 되는 탐색이득(search gain)을 그림 7의 연속 영상을 생성하면서 측정한 결과이다. 여기서 탐색이득은 다음과 같이 정의된다: $search\ gain = \frac{\eta_{without} - \eta_{with}}{\eta_{without}}$, 여기서 $\eta_{without}$과 η_{with}는 팔진트리를 사용하지 않을 때와 사용한 경우 탐색된 복셀의 개수를 각각 의미한다.

구현된 시스템을 이용하여 거의 실시간에 가까운 렌더링 속도를 얻을 수 있었다. 작은 폭을 갖는 팔진트리를 이용하는 경우 탐색공간을 제한하는 효과를 얻을 수 있기는 하지만, 여전히 탐색시간은 병목(bottleneck)현상을 나타낸다. 앞에서 팔진트리의 효과를 확인하기 위해 탐색이득을 정의했던 것과 마찬가지로, 팔진트리의 사용으로 얻을 수 있는 가시화 속도 향상에서의 렌더링이득(rendering gain)을 정의함으로써 성능을 다음과 같이 정량적으로 분석할 수 있다: $rendering\ gain = \frac{\tau_{without} - \tau_{with}}{\tau_{without}}$, 여기서 $\tau_{without}$과 τ_{with}는 팔진트리를 사용하지 않을 때와 사용한 경우 각각에 대한 렌더링 속도이다. 표 2는 연속 렌더링 영상(그림 7)을 생성할 때의

표 1 탐색된 복셀의 비율과 가장 세밀한 레벨(l=3)에서의 탐색이득

Frame	Num of total voxels	Num of voxels in searched bricks				Search gain (%)		
		no octree	octree 16	octree 8	octree 4	octree 16	octree 8	octree 4
(a)	1120288	26656 (2.4%)	24057 (2.1%)	14874 (1.3%)	4700 (0.4%)	9.7	44.2	82.4
(b)	1174304	48880 (4.2%)	39234 (3.3%)	17904 (1.5%)	5966 (0.5%)	19.7	63.4	87.7
(c)	1203872	58672 (4.9%)	51148 (4.2%)	31898 (2.6%)	11479 (1.0%)	12.8	44.5	80.4
(d)	1274336	87520 (6.9%)	68328 (5.4%)	39585 (3.1%)	13940 (1.1%)	21.9	32.0	84.0
(e)	1391776	143520 (10.3%)	81052 (5.8%)	37376 (2.7%)	12614 (1.0%)	43.5	73.9	91.2
(f)	1412992	164736 (11.7%)	164736 (7.2%)	47010 (3.3%)	16502 (1.2%)	38.3	71.5	89.9

표 2 렌더링 속도와 가장 세밀한 레벨(l=3)에서의 렌더링이득

Frame	Rendering speed (frames per second)				Rendering gain (%)		
	no octree	octree 16	octree 8	octree 4	octree 16	octree 8	octree 4
(a)	111.9	106.1	120.2	140.4	-5.2	7.4	25.5
(b)	60.6	59.7	75.3	88.3	-1.5	24.3	45.7
(c)	43.3	41.1	45.6	51.3	-5.1	5.3	18.5
(d)	29.6	29.4	54.8	35.7	-0.7	85.1	20.6
(e)	21.1	23.1	28.1	29.3	9.5	33.2	38.9
(f)	18.0	18.4	23.5	23.5	2.2	30.6	30.6

렌더링 속도와 렌더링 이득을 측정한 결과를 보여준다.

구현된 k-d 트리와 팔진트리 자료구조의 조합은 실험에 사용된 AMR 데이타를 가장 세밀한 레벨의 해상도인 64×64×128에서도 대화식으로 가시화할 수 있도록 해준다. K-d 트리를 이용해 연관된 복셀들이 존재하는 공간의 영역을 빠르게 선택할 수 있고, 팔진트리로 탐색 비용을 더욱 줄이고 설정된 카메라의 관찰 방향에 대한 복셀들의 효과적인 정렬이 가능하다. K-d 트리로 분할된 블럭들은 관찰 방향이 바뀔 때마다 적절히 재정렬되어야 하지만, 그 블럭의 숫자가 제한적이기 때문에 이 정렬과정이 병목현상을 발생시키지는 않는다. 일반적으로 볼륨 데이타들의 공간 응집성으로 인해 실제 관심의 대상이 되는 영역이 매우 제한적인 공간에 밀집되어 있음을 고려할 때, 정렬 과정에서 병목현상이 발생하지 않는다는 사실은 당연한 결과라고 할 수 있다.

주어진 AMR 데이타를 k-d 트리, 팔진트리 자료구조로 변환한 후, 실제 렌더링을 수행하는 과정에서 여러 가지 파라메터들이 사용자에 의해서 결정된다. 설정이 필요한 파라메터로는 카메라의 위치와 방향 등을 결정하는 뷰잉 파라메터와, 가시화 될 함수값의 범위와 불투명도(opacity) 등을 포함하는 렌더링 파라메터가 있다. 본 연구를 통해 구현된 사용자 인터페이스는 간단한 메뉴 조작을 통해 이러한 파라메터들을 쉽게 설정할 수 있는 기능을 제공한다. 그림 9는 실험에 사용된 시간가변 AMR 데이타를 가시화하고 있는 사용자 인터페이스를 캡춰한 것이다. 그림을 통해 확인할 수 있는 바와 같이, 시간의 흐름에 따른 변화를 스플래팅 기법으로 가시화한 결과 영상을 상위 윈도우를 통해 확인할 수 있으며, 아래쪽의 컬러맵의 색과 그래프를 바꿔가면서 함수값에 따라 할당될 색깔과 불투명도를 수정할 수 있도록 설계하였다. 특정 색이나 불투명도가 설정되지 않은 함수값들에 대해서는 선형보간을 통해 색과 불투명도를 결정한다. 그리고, 우측하단에 있는 숫자 입력 부분에서 가시화할 함수값의 종류 v, 렌더링에 사용할 데이타의 세분화 레벨 l, 스플래팅 알고리즘에서 사용되는 풋프린

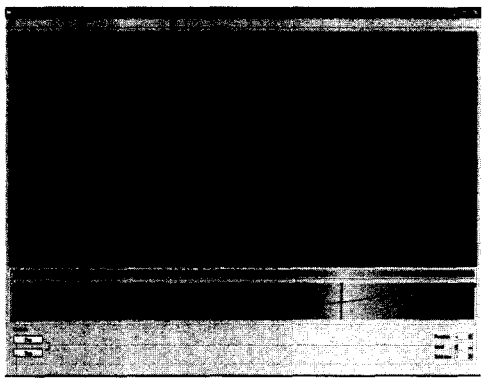


그림 9 대화식 렌더링을 위해 개발된 사용자 인터페이스

트 테이블의 크기 등을 설정할 수 있다. 마지막으로 좌측하단의 작동, 정지 버튼을 이용해 시간의 흐름에 따른 데이타의 변화를 마치 비디오를 보듯이 조절할 수 있다.

4. 결론 및 향후연구

본 논문에서는 AMR 데이타를 대화식으로 가시화하기 위한 기법으로 계층적 자료구조를 이용한 그래픽스 하드웨어 가속 다중해상도 스플래팅 기법의 개발에 대해 설명했다. 이 기법은 전처리를 통해 주어진 AMR 데이타로부터 k-d 트리와 팔진트리 자료구조를 생성하고, 이 자료구조들은 AMR 데이타의 효율적인 렌더링과 저장을 위해 이용된다. 또한, 본 기법은 대화식 스플래팅 알고리즘의 구현을 위해 nVIDIA GeForce3 그래픽스 카드에 의해 제공되는 2차원 텍스춰 매핑의 하드웨어 가속 기능을 이용한다. 또한, 구현된 사용자 인터페이스는 뷰잉/렌더링 파라메터를 자유롭게 변경시켜가면서, 다양한 렌더링 영상을 대화식으로 만들 수 있는 기본적인 수행 환경을 제공한다.

기존의 연구 결과들은 AMR 데이타를 가시화 하는 문제 자체에만 관심을 기울였을 뿐, 실시간 렌더링이나 대화식 렌더링의 속도로 영상을 생성할 수 없었다. 그러나 본 기법은 k-d 트리, 팔진트리 자료구조와 고성능 그래픽스 하드웨어를 효율적으로 결합하여 초당 20 프레임 정도의 속도로 렌더링을 수행할 수 있다는 점에서 기존의 연구 결과와 차별화 된다.

본 기법에 추가적으로 적용될 수 있는 연구 내용으로서, 우선 AMR 데이타의 병렬 렌더링을 생각할 수 있다. 현재의 k-d 트리 기반 스플래팅 기법은 병렬 렌더링으로 확장이 용이한 자료구조를 갖고 있다. 관찰 방향의 변화에 따라 생성되는 블릭 리스트를 병렬 렌더링을 위한 하나의 작업 풀(task pool)로서 고려할 수 있다. 다시말해, 병렬 렌더링 알고리즘의 전형적인 구조인 하나의 마스터 프로세서(master processor)와 여러 개의 슬레이브 프로세서들(slave processors)이 계산 참여하는 상황으로 생각할 수 있다. 마스터 프로세서는 계산을 하지 않고 있는 적절한 슬레이브 프로세서에게 블릭을 할당하고, 슬레이브 프로세서의 렌더링 결과인 부분 영상을 받아 미리 정렬된 블릭 순서에 따라 합성하고 그 슬레이브 프로세서에게 다시 다음 블릭을 할당하는 작업을 반복한다. 각 슬레이브 프로세서들은 할당 받은 블릭을 로드하여 스플래팅 알고리즘을 이용해 해당 블릭에 대한 부분 영상을 생성하고 이를 마스터 프로세서에게 보내는 작업을 반복하게 된다.

또 다른 시도로서, 시간 가변 AMR 데이타의 시간 응집성(temporal coherence)을 이용한 인코딩과 렌더링 방법을 구현하는 것을 고려할 수 있다. 만일 AMR 데이타를 위한 손실 또는 무손실 압축 기법이 개발된다면 방대한 크기의 볼륨 데이타는 더욱 간결한 형태로 저장될 수 있을 것이고, 방대한 볼륨 데이타 가시화에서 병목현상을 일으키는 가장 주요한 원인이 되는 디스크 액세스 비용과 데이타 전달 비용을 줄일 수 있게 되어 작업의 효율성을 향상시킬 수 있을 것이다.

마지막으로, 본 논문의 방법으로 AMR 데이타를 렌더링하고 동시에 OpenGL 파이프라인을 이용해 일반적인 기하 모델(geometric models) 데이타를 렌더링하여 하나의 합성 영상을 만드는 것을 생각할 수 있다. 예를 들어, 다각형들로 이루어진 우주선 모델이 AMR 데이타로 만들어진 우주 데이타(cosmology data) 속을 유영하는 영상이나 애니매이션을 제작하는 것을 고려할 수 있다. AMR 데이타 공간이 k-d 트리에 의해 분할될 수 있으므로, 우주선을 포함하고 있는 블릭을 쉽게 찾을 수 있다. 또한, 스플래팅 기법 자체가 OpenGL을 사용한 하드웨어 가속 기능을 이미 이용하고 있기 때문에, AMR 볼륨 데이타와 일반 기하 모델을 함께 렌더링하는 것은 어렵지 않게 구현될 수 있을 것이다.

참 고 문 헌

[1] M. Berger and J. Oliger, "Adaptive mesh refinement for hyperbolic partial differential equations," *Journal of Computational Physics*, Vol. 53, pp. 484-512, 1984.

[2] G. L. Bryan, "Fluids in the universe: adaptive mesh refinement in cosmology," *Computing in Science & Engineering*, Vol. 1, No. 2, pp. 46-53, 1999.

[3] L. Norman, J. M. Shalf, S. Levy, and G. Daues, "Diving deep: data-management and visualization strategies for adaptive mesh refinement simulations," *Computing in Science & Engineering*, Vol. 1, No. 4, pp. 36-47, 1999.

[4] G. H. Weber, O. Kreylos, T. J. Ligocki, J. M. Shalf, H. Hagen, B. Hamann, and K. I. Joy, "Extraction crack-free isosurfaces from adaptive mesh refinement data," In *Proceedings of the Joint EUROGRAPHICS and IEEE TCVG Symposium on Visualization*, pp. 25-34, May 2001.

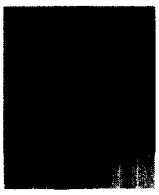
[5] G. H. Weber, H. Hagen, B. Hamann, K J. Joy, T. J. Ligocki, K.-L. Ma, and J. M. Shalf, "Visualization of adaptive mesh refinement data," In *Proceedings of the SPIE (Visaul Data Exploration and Analysis VIII)*, May 2001.

[6] G. H. Weber, O. Kreylos, T. J. Ligocki, J. M. Shalf, H. Hagen, B. Hamann, K. I. Joy, and K.-L. Ma, "High-quality volume rendering of adaptive mesh refinement data," In *Proceedings of the 6th International Fall Workshop on Vision, Modeling, and Visualization 2001*, pp. 121-128, Nov. 2001.

[7] R. Kähler, M. Simon, and H.-C. Hege, "Interactive volume rendering of large sparse data sets using adaptive mesh refinement hierarchies," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 9, No. 3, pp. 341-351, 2003.
[8] S. Park, C. L. Bajaj, and V. Siddavanahalli, "Case study: interactive rendering of adaptive mesh refinement data," *Proceedings of IEEE Visualization 2002*, pp. 521-524, Oct. 2002.
[9] L. Westover, "Footprint evaluation for volume rendering," *Computer Graphics*, Vol. 24, No. 4, pp. 367-376, 1990.
[10] D. Laur and P. Hanrahan, "Hierarchical splatting: a progressive refinement algorithm for volume rendering," *Computer Graphics*, Vol. 25, No. 4, pp. 285-288, 1991.
[11] C. L. Bajaj, V. Pascucci, G. Rabbiolo, and D. R. Schikore, "Hypervolume visualization: A challenge in simplicity," In *Proceedings of IEEE/ACM 1998 Symposium on Volume Visualization*, pp. 95-102, Oct. 1998.
[12] R. Crawfis and N. Max, "Texture splats for 3D scalar and vector field visualization," In *Proceedings of IEEE Visualization'93*, pp. 261-267, Oct. 1993.
[13] E. LaMar, B. Hamann, and K. I. Joy, "Multi-resolution Techniques for Interactive Texture-Based Volume Visualization," *Proceedings of IEEE Visualization '99*, pp. 24-29, Oct. 1999.

박 상 훈

1993년 서강대학교 수학과(이학사). 1995년 서강대학교 컴퓨터학과(공학석사). 2000년 서강대학교 컴퓨터학과(공학박사) 2000년 3월~2000년 6월 서강대학교 컴퓨터학과, 박사후연구원. 2000년 7월~2002년 8월 Univ. of Texas at Austin, Post-Doc. 2002년 9월~현재 대구가톨릭대학교 컴퓨터정보통신공학부 전임강사. 관심분야는 컴퓨터그래픽스, 과학적가시화, 실시간렌더링, 고성능컴퓨팅



Chandrajit Bajaj

CAM Chair in Visualization. Professor of Computer Sciences, The University of Texas at Austin. Director of Center for Computational Visualization. B. Tech. in Electrical Engineering (1980), Indian Istitute of Technology, New elhi, India. M.S. and Ph.D in Computer Science (1983, 1984), Cornell University, Ithaca, New York. Research Interests: Computer Graphics, Computational Mathematics, Geometric Design, Data Visualization