분산 메모리 구조를 갖는 병렬 컴퓨터 상에서의 압축 기반 볼륨 렌더링

(Compression-Based Volume Rendering on Distributed Memory Parallel Computers)

구 기 범 † 박 상 훈 †† 송 동 섭 ††† 임 인 성 ††††

(Gee-Bum Koo) (Sanghun Park) (Dongsub Song) (Insung Ihm)

요 약 본 논문에서는 분산 메모리 구조를 갖는 병렬 컴퓨터 상에서 방대한 크기를 갖는 볼륨 데이타의 효과적인 가시화를 위한 병렬 광선 투사법을 제안한다. 데이타의 압축을 기반으로 하는 본 기법은 다른 프로세서의 메모리로부터 데이타를 읽기보다는 자신의 지역 메모리에 존재하는 압축된 데이타를 빠르게 복원함으로써 병렬 렌더링 성능을 향상시키는 것을 목표로 한다. 본 기법은 객체-순서와 영상-순서 탐색 알고리즘 모두의 장점을 이용하여 성능을 향상시켰다. 즉, 블록 단위의 최대-최소 팔진트리의 탐색과 각 픽셀의 불투명도 값을 동적으로 유지하는 실시간 사진트리를 응용함으로써 객체-공간과 영상-공간 각각의 응집성을 이용하였다. 본 논문에서 제안하는 압축 기반 병렬 볼륨 렌더링 방법은 렌더링 수행 중 발생하는 프로세서간의 통신을 최소화하도록 구현되었는데, 이러한 특징은 프로세서 사이의 상당히 높은 데이타 통신 비용을 감수하여야 하는 PC 및 워크스테이션의 클러스터와 같은 더욱 실용적인 분산 환경에서 매우 유용하다. 본 논문에서는 Cray T3E 병렬 컴퓨터 상에서 Visible Man 데이타를 이용하여 실험을 수행하였다.

Abstract This paper proposes a new parallel ray-casting scheme for very large volume data on distributed memory parallel computers. Our method, based on data compression, attempts to enhance the speedup of parallel rendering by quickly reconstructing data from local memory rather than expensively fetching them from remote memory spaces. Futhermore, it takes the advantages of both object-order and image-order traversal algorithms. It exploits object-space and image-space coherence, respectively, by traversing min-max octree block-wise and using a run-time quadtree which is maintained dynamically against pixels' opacity values. Our compression-based parallel volume rendering scheme minimizes communications between processing elements during rendering, hence is also very appropriate for more practical distributed systems, such as clusters of PCs and/or workstations, in which data communications between processors are regarded as quite costly. We report experimental results on a Cray T3E for the Visible Man dataset.

· 이 논문은 1999년도 한국학술진흥재단의 대학교수 해외파견 연구지원에 의하여 연구되었음.

† 비 회 원 : (주)인젠 연구원
kgb@inzen.com
†† 비 회 원 : The University of Texas at Austin
hun@brahma.ticam.utexas.edu
††† 비 회 원 : LG전자 연구소 연구원
lower case@lge.co.kr
†††† 종신회원 : 서강대학교 컴퓨터학과 교수
ihm@sogang.ac.kr

논문접수 : 2000년 1월 7일
심사완료 : 2000년 7월 24일

1. 서 론

최근 자연 과학, 공학, 의학 등의 다양한 분야에서 방대한 크기를 갖는 볼륨 데이타들이 많이 산출되고 있다. 특히, 미국의 국립 의학 도서관 NLM(National Library of Medicine)에서 Visible Human 프로젝트를 통해 CT, MRI, RGB로 이루어진 남자, 여자에 대한 인체 데이타를 만들었는데 각각의 크기는 15 기가 바이트(giga bytes)에서 40 기가 바이트에 이른다[1]. 이렇게 방대한 볼륨 데이타의 가시화는 많은 계산뿐만 아니라 매우 큰 메모리 공간을 요구한다. 기존의 다양한 볼륨 렌더링 기

법들 가운데 광선 투사법은 다른 기법들에 비해 고화질의 이미지를 얻을 수 있다는 장점을 갖지만, 상당히 많은 양의 계산 비용을 필요로 한다. 더욱이, Visible Human 데이타와 같이 방대한 크기의 볼륨 데이타를 단일 프로세서와 제한된 주기억장치를 갖는 컴퓨터를 이용해 렌더링하는 경우, 아무리 효과적인 알고리즘을 이용한다고 하더라도 실시간(real-time) 또는 대화식(interactive) 가시화는 매우 어려운 문제이다.

이와 같이 계산 집중적이고(CPU-intensive) 메모리 집중적인(memory-intensive) 방대한 볼륨 데이타의 가시화 작업을 효과적으로 수행하기 위한 다양한 연구들이 병렬, 분산 프로그래밍 분야에서 진행되어 왔다. Nieh[2]는 이미지 평면을 PE(Processing Element) 개수만큼의 이미지 블록으로 나누고 각 이미지 블록을 다시 작은 타일로 세분했다. PE는 렌더링을 시작하기 전에 하나의 이미지 블록을 할당받으며 스캔라인 순서에 따라 타일을 렌더링 한다. 할당받은 작업을 먼저 끝낸 PE는 아직 렌더링이 끝나지 않은 PE의 이미지 블록의 타일을 가져와서 렌더링을 수행한다. 여기서 사용한 작업 큐 영상 분할(task queue image partitioning) 방법은 확장성이 좋고 동적 부하 균형의 구현이 용이하다는 장점을 갖고 있다. Montani[3]는 영상-공간 분할 기법과 객체-공간 분할기법을 모두 이용한 병렬 볼륨 렌더링 알고리즘을 제안했다. 렌더링에 사용되는 PE는 여러 개의 클러스터로 구성되며 각 클러스터는 볼륨 데이타를 클러스터내의 PE에 분산시킨다. 다시 말해 클러스터의 수만큼 데이타가 중복되어 있고 렌더링 도중 다른 PE로부터 데이타를 가져와야 할 경우 동일 클러스터 내의 PE로부터 가져오도록 해서 PE간의 통신을 분산시켰다. 또한 이미지 평면도 작게 나누어 모든 클러스터에 고르게 분배하도록 하였다. Ma[4]는 객체-공간 분할 기법을 응용한 병렬 볼륨 렌더링 알고리즘을 제안했다. 볼륨 데이타는 k-D 트리를 이용해서 분할되며 나누어진 데이타 블록은 각 PE에 저장된다. PE는 할당받은 데이타를 렌더링한 부분 이미지를 저장하고 있다가 모든 PE의 렌더링이 끝난 후 최종 이미지를 합성하는 과정에 참여하도록 하였다. 이와 같은 대부분의 병렬, 분산 렌더링 기법들은 각 PE에 분산 저장되어 있는 부분 볼륨 데이타를 주고받으면서 가시화를 수행하기 때문에 데이타 통신으로 인해 발생하는 성능 저하를 피할 수가 없다. 이러한 현상은 볼륨 데이타의 크기가 커질수록 더욱 심각하게 나타나게 된다[5].

본 논문에서는 병렬 및 분산 환경에서 방대한 볼륨 데이타의 효과적인 가시화를 위해 선형적인 속도 향상에 더욱 접근하기 위한 새로운 병렬 볼륨 렌더링 알고리즘을 제시하고 이를 구현하였다. 본 논문은 병렬 컴퓨터를 이용하여 Visible Human 데이타와 같은 방대한 볼륨 데이타의 효과적인 가시화를 수행할 수 있는 병렬 광선 투사 알고리즘의 개발을 목표로 하며, 실제로 분산 메모리 병렬 컴퓨터(distributed-memory parallel computer)인 Cray T3E-900 상에서 Visible Man CT 데이타의 병렬 렌더링 기법을 구현하였다. 이 기법은 방대한 볼륨 데이타의 가시화에 있어 발생하는 두 가지 큰 문제의 해결에 초점을 맞추고 있다. 첫째로, 계산 집중적인 작업의 문제는 분산 메모리 구조를 갖는 Cray T3E상에서 효과적인 병렬 광선 투사 알고리즘의 구현을 통해 해결한다. 이를 위해 영상 순서(image-order)와 객체 순서(object-order) 방법이 갖는 장점들을 최대한 이용하기 위한 자료구조와 알고리즘을 개발하였다. 둘째로, 메모리 집중적인 작업의 문제는 데이타 압축을 통하여 해결하였다. 기존의 병렬 볼륨 가시화 기법들 중에서도 데이타의 통신 시간을 줄이기 위해 압축을 시도하는 경우가 있었으나, 본 논문에서 제시하는 압축은 프로세서 사이에서 볼륨 데이타를 주고받기 위해 발생하는 데이타 통신 시간을 완전히 없애기 위한 것이다. 다시 말해, 방대한 전체 볼륨 데이타를 64~128 메가 바이트(mega bytes) 정도의 크기로 압축하고, 각 프로세서가 압축된 전체 볼륨 데이타를 자신의 로컬 주기억장치에 로드(load)한 상태에서 렌더링에 필요한 부분의 정보만을 빠르게 복원하면서 병렬 가시화를 수행하도록 만드는 것이다. 이를 위하여 방대한 볼륨 데이타의 가시화에서 좋은 성능을 나타내는 웨이블릿(wavelets)에 기반을 둔 효과적인 3차원 압축 기법을 이용하였는데, 이 기법은 비교적 높은 압축율과 데이타의 임의의 지점에 대한 매우 빠른 랜덤 액세스가 가능하다는 장점을 갖고 있다[6,7,8,9]. 이러한 압축 기반 볼륨 가시화 기법은 고가의 병렬 컴퓨터에서뿐만 아니라 범용의 PC와 워크스테이션 클러스터(cluster)로 구성된 분산 시스템에서도 매우 유용하게 이용될 수 있다. 물론, 이용된 압축 기법은 데이타 통신 시간보다 빠른 복원 속도를 제공하며, 실험을 통해 기존의 볼륨 데이타 재분배를 기반으로 한 기법들보다 우수한 성능을 확인할 수 있었다[10].

본 논문은 다음과 같이 구성된다. 2장에서는 볼륨 데이타의 압축에 사용된 웨이블릿 기반 압축 기법에 대해 간단히 살펴보고, 3장에서는 구현된 압축 기반 병렬 볼륨 광선 투사법과 이를 병렬 볼륨 렌더링 알고리즘으로 확장하는 방법에 대해 설명한다. 그리고, 4장에서는 실험 결과에 대해 분석하고, 5장에서 결론과 추후의 연구

방향에 대해 언급한다.

2. 볼륨 데이타의 웨이블릿 기반 3차원 압축

본 논문에서 제안하는 병렬 볼륨 렌더링 알고리즘의 구현을 위해 사용되는 압축 기법은 다음과 같은 조건들을 만족시켜야 한다.

- 높은 압축율(high compression ratio): 수 기가 바이트의 데이타를 64~128 메가 바이트 정도의 크기로 압축할 수 있어야 한다.
- 오차의 최소화(minimal distortion): 복원된 데이타와 원본 데이타의 오차는 최소화되어야한다.
- 빠른 랜덤 액세스(fast random access ability): 추가적인 메모리 없이 임의로 선택된 복셀의 정보를 빠르게 랜덤 액세스할 수 있어야 한다.

본 논문에서는 Ihm[7,8]에서 초기 연구 결과가 발표되고, Bajaj[6] 및 Park[9]에서 성능이 크게 향상된 제로비트 인코딩 스킴(zerobit encoding scheme)을 이용하여 Visible Human 데이타를 압축하였다. 제로비트 인코딩 스킴은 웨이블릿을 기반으로 하는 3차원 압축을 통해 위의 세 가지 조건을 만족하도록 구현되었고, 방대한 데이타를 실시간으로 처리해야 하는 컴퓨터 그래픽스의 다양한 분야에 응용될 수 있다[6,10,11]. 이 기법을 이용하면 Visible Human 데이타 전체를 각 프로세서의 지역 메모리보다 작은 64~128 메가 바이트 정도의 크기로 압축할 수 있고, 비교적 적은 오차로 복원이 가능하다. 특히, 압축된 자료 구조로부터 원하는 부분에 대한 정보만을 빠르게 랜덤 액세스할 수 있다는 특징은 제로비트 인코딩 스킴의 가장 중요한 장점이며, 이를 이용해 병렬 렌더링의 성능 향상을 얻게 된다. 제로비트 인코딩 스킴이 위에서 제시한 세 가지 조건을 만족한다는 사실은 Visible Man fresh CT 데이타의 압축을 통해 얻어진 실험 결과를 통해 확인할 수 있다. 이 실험은 195 MHz MIPS R10000 CPU와 256 MByte의 메인 메모리를 갖는 SGI Octane 워크스테이션에서 수행되었다. 표 1, 2는 웨이블릿 변환을 수행한 후 생성된 전체 계수들 가운데 실제 인코딩 되는 다양한 계수의 비율에 대한 실험 결과를 보여준다[6,9]. 표 1에서 압축율은 전체 Visible Man fresh CT 데이타 (512×512×1440×2 byte = 720 MByte)를 이용해 얻은 결과이고, 표 2는 상위 512개 슬라이스 (512×512×512×2 byte = 256 MByte)만을 대상으로 한 것이다. 우선, 표 1은 압축 성능에 대한 실험 결과이다. 복원 오차의 정도를 측정하기 위해 SNR(Signal to Noise Ratio)과 PSNR(Peak Signal to Noise Ratio)을 3차원 볼륨 데이타에 대한 경우로 확장하여 적용하였고, 동일한 샘플링 위치에서 압축하지 않은 데이타와 압축된 데이타 각각의 노말 벡터(normal vector)가 이루는 각도의 오차를 계산하였다. 실제로 7% 이상의 계수를 사용하여 압축한 데이타를 이용해 광선 투사법을 수행한 결과 영상은 압축하지 않은 데이타를 이용해 볼륨 렌더링 한 영상과 거의 동일한 화질을 나타낸다 (그림 8). 표 2는 데이타의 랜덤 액세스 속도에 대한 실험 결과이다. 랜덤 액세스 속도의 측정을 위해 두 가지 복원 방법을 고려했다. 첫 번째는 순수한 랜덤(Pure Random) 액세스의 경우이다. 복원 과정의 오버헤드(overhead)를 측정하기 위해 압축하지 않은 볼륨 데이타를 배열로부터 백만 번에 걸쳐 랜덤 액세스하고 동일한 수행을 압축된 데이타에 대해 적용하였다. 두 번째는 4×4×4 크기의 셀(cell) 이라 불리는 압축의 기본 단위에 대해 셀 단위(Cell-Wise) 액세스를 수행하는 경우이다. 실험에서는 데이타에 포함된 모든 셀을 복원하는 경우(All)와 피부로 분류된 셀만을 복원하는 경우(Skin)로 구분하여 소비되는 시간을 측정하였다. 셀 단위 액세스는 규칙적인 액세스 패턴을 갖는 응용 분야에서 더욱 효과적으로 이용될 수 있다. 이러한 실험 결과로부터 제로비트 인코딩 스킴이 비교적 좋은 복원 화질에서도 높은 압축을 나타내고, 특히 빠른 랜덤 액세스를 지원한다는 사실을 확인할 수 있다.

표 1 압축율 및 오차에 대한 실험 결과

		Desired Ratio of the Wavelet Coef's Used			
		3%	5%	7%	10%
Compression Performance	Size (MB)	24.59	35.06	45.43	60.50
	Ratio	29.27	20.54	15.58	11.90
Errors in Voxel Values	SNR(dB)	22.64	25.94	28.57	31.99
	PSNR(dB)	44.49	47.79	50.41	53.84

표 2 복원 속도에 대한 실험 결과 (단위: 초)

		Uncompressed	Desired Ratio of the Wavelet Coef's Used			
			3%	5%	7%	10%
Pure Random		2.78	3.33	3.49	3.62	3.85
Cell-Wise	All	18.88	3.88	4.88	5.99	7.52
	Skin	6.50	2.28	2.91	3.53	4.36

3. 압축 기반 병렬 볼륨 렌더링

3.1 영상-순서와 객체-순서 볼륨 렌더링

병렬 볼륨 렌더링 알고리즘은 크게 영상-순서(image-order)와 객체-순서(object-order) 방법으로 구분된다. 영상-순서 방법은 영상 평면(image plane)의 각 픽셀에서 출발한 광선에 대한 색깔을 결정함으로써 최종적인 영상을 만드는 것으로, 광선 투사법이 대표적인 예이다. 언급한 바와 같이, 이 기법은 비교적 많은 시간을 소비하지만 가장 좋은 영상을 생성하는 기법으로 알려져 있다[12]. 또한, 영상-순서 방법은 샘플링을 빨리 종료할 수 있는 조기 광선 종료(early ray termination) 기법[13], 그리고 볼륨 데이타에 내재된 응집성(coherency)을 계층적인 자료 구조로 표현하고 이를 렌더링 과정에서 이용하는 팔진트리(octree), 피라미드(pyramid), k-D 트리 기법과 같은 기존의 효과적인 렌더링 속도 향상 기법들을 이용할 수 있다는 장점을 갖고 있다[13,14,15]. 그러나, 병렬 광선 추적 과정에서 데이타의 액세스 패턴이 불규칙적이기 때문에 영상-순서 볼륨 렌더링 방법은 객체-순서 방법에 비해 병렬화가 자연스럽지 못하다는 단점을 갖는다.

객체-순서 방법은 볼륨 데이타 공간에서 각 복셀을 영상 평면으로 사영 (projection)시킴으로써 영상을 생성하며 스플래팅(splatting)이 대표적인 기법이다. 이 방법은 데이타를 규칙적으로 액세스하기 때문에 데이타 응집성을 매우 효과적으로 이용할 수 있으며 병렬화가 직관적이고 간단하다는 장점을 갖는다[16]. 그러나 조기 광선 종료나 계층적인 자료 구조들을 이용하는 속도 향상 기법들을 적용하기가 쉽지 않고, 좋은 화질의 영상을 만드는데 한계가 있다는 단점을 갖고 있다.

3.2 효과적인 광선 추적법

본 논문에서 개발한 볼륨 렌더링 기법은 영상-순서 방법과 객체-순서 방법이 갖는 장점들을 최대한 활용함과 동시에 병렬화가 용이하도록 설계되었다. 전처리과정으로, 볼륨 데이타는 앞에서 설명한 제로비트 인코딩 스킴을 이용하여 수십 MB 크기로 압축된다. 영상 평면은 동일한 크기의 작은 타일(tile)로 분할되어 PE가 처리해야 할 작업의 풀(pool)을 이루게 되며, PE는 한 번에 하나의 타일을 할당받아 해당 영역에 대한 광선 투사 영상을 계산한다. 다수의 PE를 사용하여 렌더링할 때에는 PE를 몇 개의 그룹(group)으로 나누어서 각 그룹에 대한 작업 풀을 구축한다. PE는 자신이 속한 그룹의 풀로부터 타일을 가져와 렌더링을 수행하고, 특정 그룹의 작업이 모두 끝나면 아직 종료하지 못한 다른 그룹에 할당된 타일을 렌더링한다. 이러한 과정을 반복해서 모든 작업 풀에 더 이상 타일이 남아있지 않을 때 전체 렌더링이 끝나게 된다.

3.2.1 볼륨 데이타의 압축

일반적으로 병렬 볼륨 렌더링 알고리즘은 ① PE의 고른 작업량 분배 ② 효율적인 PE간의 통신 ③ 확장성 문제를 해결하는데 중점을 둔다. 본 논문에서처럼 분산 메모리를 갖는 병렬 컴퓨터나 PC/워크스테이션 클러스터 환경에서 렌더링을 수행할 때 가장 주의해야 할 점은 PE간의 통신을 줄여서 많은 PE를 사용할 때 나타나는 성능저하를 최소화해야 한다는 것이다. 기존의 많은 병렬 볼륨 렌더링 알고리즘들은 데이타를 각 PE에 분산시키고 필요할 경우 다른 PE로부터 데이타를 가져오는 방법을 채택했기 때문에 전체적으로 좋은 성능을 보여주지 못했다. 이 문제를 해결하기 위해 본 논문에서는 2장에서 설명한 제로비트 압축 기법으로 데이타를 압축해서 모든 PE가 압축된 전체 데이타를 로드할 수 있도록 했다. 이와 같이 하면 렌더링 도중 다른 PE로부터 데이타를 가져올 필요가 없기 때문에 렌더링 할 타일의 인덱스를 가져오는 것과 렌더링이 끝난 타일의 부분 영상을 전송하는 것이 PE간 통신의 전부이며, 이것은 볼륨 데이타가 재분배되어야 하는 경우에 비해 극히 적은 통신 비용이다.

전체 데이타는 전처리 과정을 통해서 16x16x16 크기의 단위 블록(unit block)을 기본 단위로 하여 인코딩된다. 각 단위 블록은 내부에 포함된 16x16x16개 복셀의 최대-최소값에 대한 정보를 유지하도록 설계되었는데, 이를 이용해 렌더링 결과에 영향을 주지 않는 복셀만을 포함하는 단위 블록의 복원을 피할 수 있고, 결과적으로 전체 렌더링 속도를 향상시킬 수 있다. 압축에 사용된 제로비트 인코딩 스킴은 높은 압축율을 갖고 있으며 매우 빠른 복원 속도를 제공하기 때문에 데이타의 어떤 부분이라도 단시간 내에 액세스할 수 있다는 장점을 갖고 있다. 이렇게 생성된 단위 블록은 3.2.2와 3.2.3에서 설명하는 알고리즘의 기본 자료구조가 된다.

3.2.2 단일 PE에서의 볼륨 렌더링: 객체-순서 알고리즘의 응용

PE는 작업 풀로부터 타일을 하나씩 가져와서 그 타일에 해당하는 부분 영상을 렌더링한다. 렌더링할 타일이 결정되면 제일 먼저 이에 대응되는 뷰-볼륨(view-volume)을 계산한다 (그림 1). 그 다음, 최종 부분 영상에 영향을 주지 않는 단위 블록의 복원을 피하기 위해 뷰-볼륨에 포함되거나 교차하는 단위 블록의 리스트를 결정한다. 그런데 각 타일을 렌더링할 때마다 데이타를

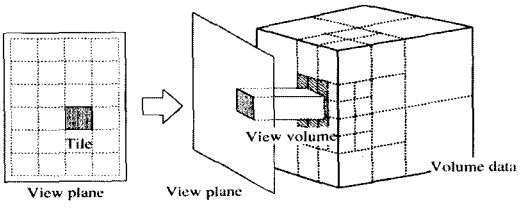


그림 1 View volume의 결정

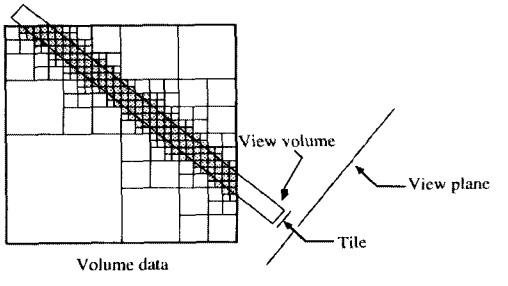


그림 2 단위블록의 선택

구성하는 모든 단위 블록과 뷰-볼륨의 교차 여부를 판단하는 것은 비효율적이며, 특정 타일의 뷰-볼륨과 교차하는 단위 블록의 리스트를 직접 구하는 것도 간단한 문제가 아니다. 따라서, 본 논문에서는 전처리과정으로 단위 블록의 위치 좌표에 대한 최대-최소값을 유지하는 팔진트리(octree)를 만들고, 이를 이용하여 단위 블록의 리스트를 효율적으로 구할 수 있도록 하였다. 팔진트리의 리프 노드는 압축된 단위 블록에 대응되며, 인접한 여덟 개의 노드가 모여서 부모 노드를 구성한다. 리프 노드를 제외한 모든 노드는 자식 노드에 대해 일정한 인덱스(index)를 부여하는데, 이것은 타일로부터 가까운 거리에 있는 단위 블록을 먼저 처리하기 위해 영상 평면과 볼륨 데이타의 상대적인 위치에 따라 매핑 테이블(mapping table)을 생성하여 노드의 검색 순서를 변화시키기 위한 것이다. 예를 들어 그림 3에서와 같이 뷰방향(viewing direction)이 y축에 가까운 경우 0, 1, 4, 5, 2, 3, 6, 7번의 순서로 노드를 검색하게 된다.

팔진트리의 각 노드와 뷰-볼륨과의 교차여부는 노드를 영상평면에 투영시켰을 때 나타나는 영역이 현재 렌더링을 하는 타일과 교차하는 부분이 있는가에 따라서 결정을 하게 된다. 또한, 불필요한 계산을 줄이기 위해 단위 블록의 리스트는 한번에 구하지 않고, 팔진트리의 검색 도중 해당 단위 블록을 구했을 때 그 때까지의 팔진트리 검색상황을 임시로 저장하고 3.2.3절에서 설명하는 영상-순서 알고리즘을 이용해서 타일을 렌더링 하도록 하였다. 그 단위 블록의 처리가 끝나면 팔진트리의

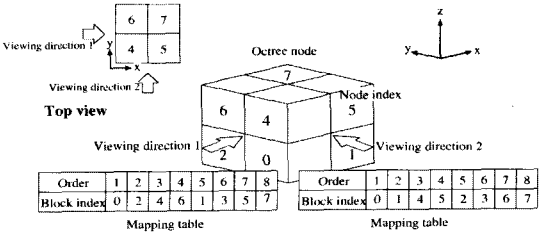


그림 3 팔진트리의 노드 검색 순서

검색이 끝난 시점부터 다시 검색을 시작하며, 이 과정은 해당 타일에 대한 렌더링이 완전히 종료될 때까지 계속된다. 그림 2는 이러한 과정을 통해 결정되는 단위 블록을 2차원적으로 보여주고 있다. 이렇게 구해진 단위 블록 리스트와 영상-순서 알고리즘을 이용하면, 특정 타일을 렌더링하기 위해 한 번 복원된 단위 블록은 그 타일을 렌더링한 후에는 더 이상 고려될 필요가 없기 때문에 데이타의 응집성(coherency)을 효과적으로 이용할 수 있게 된다.

3.2.3 단일 PE에서의 볼륨 렌더링: 영상-순서 알고리즘의 응용

하나의 단위 블록을 렌더링할 때에는 영상-순서 방법을 적용한다. 렌더링 속도를 향상시키기 위해, 단위 블록을 영상 평면으로 사영시켰을 때 나타나는 영역 상자(bounding box)를 계산함으로써 불필요한 광선 투사의 횟수를 줄이고, 기존의 빠른 렌더링을 위해 개발된 기법인 조기 광선 종료(early ray termination)와 사진트리(quadtree)를 이용하였다. 일반적으로 하나의 단위 블록이 타일에 사영되었을 때 나타나는 영역은 타일의 일부분만을 차지하게 된다. 따라서 투영된 영역에 속하는 픽셀에 대해서만 광선 추적법을 적용하는 것이 합리적이다. 객체-순서 알고리즘은 데이타의 응집성을 효율적으로 이용하기에 적합하나 광선 추적법 구현에 있어 가장 중요한 성능 향상 기법 중의 하나인 조기 광선 종료 기법을 사용하기가 어렵다. 따라서 본 기법에서는 Lee[17]에서 사용된 것과 유사한 방법으로 영상 공간에서의 사진트리(quadtree)를 사용하여 조기 광선 종료 기법 효과를 얻음과 동시에 영역 상자에 들어오는 픽셀만을 빠르게 선택해서 렌더링을 수행할 수 있도록 설계하였다(그림 4).

본 논문에서 사용한 사진트리는 하나의 타일 전체를 루트 노드(root node), 각 픽셀을 리프 노드(leaf node)로 정의하며, 인접한 네 개의 픽셀에 해당하는 리프 노드가 모여서 하나의 부모 노드를 표현하도록 했다. 사진트리의 모든 내부 노드(internal node)는 카운터를 갖고

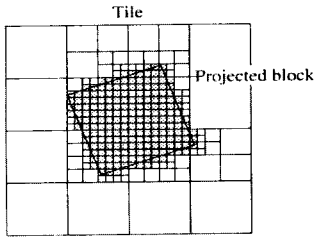


그림 5 사진트리의 응용

있으며, 렌더링 수행 중 카운터의 값이 동적으로 갱신된다. 보다 자세히 설명하면, 한 픽셀로부터 출발한 광선을 처리할 때 불투명도가 한계값(threshold) 이상이 될 경우, 그 픽셀에 해당하는 리프 노드의 부모 노드(parent node)의 카운터가 1 증가하게 된다. 모든 내부 노드는 자신의 카운터가 4가 될 때 부모 노드의 카운터를 1 증가시킨다. 한 노드의 카운터가 4가 되었다는 것은 그 노드에 해당하는 모든 픽셀의 불투명도가 한계값을 넘었음을 뜻하며, 더 이상 광선 추적 계산을 할 필요가 없음을 의미하는 것이다. 이와 같은 과정은 루트 노드의 카운터의 값이 4가 되거나 더 이상 렌더링 할 단위 블록이 없을 때까지 계속되며, 이러한 알고리즘을 통해 조기 광선 종료 기법을 구현할 수 있다. 사진트리를 이용해서 얻을 수 있는 또 하나의 장점은 블록이 타일에 사영되었을 때, 트리를 검색하는 것만으로 투영된 영역에 속하는 픽셀을 빠르게 찾아낼 수 있다는 것이다.

이와 같이 구현된 객체-순서, 영상-순서 기법을 응용한 단일 PE에서의 전체적인 렌더링은 알고리즘 1과 같이 정리될 수 있다.

```
for each tile T
  while (traverse octree)
    if (found unit block B projected onto T)
      while (traverse tile quadtree)
        if (found pixel P that is in projected area &&
            opacity < threshold)
          raycast (P)
          update quadtree
        endif
      end while
    end if
  end while
end for
```

알고리즘 1 단일 PE에서의 렌더링 알고리즘

3.2.4 캐쉬의 이용

데이타의 응집성(coherency)으로 인해 한 번 복원된 블록은 다른 광선을 처리할 때 다시 사용될 가능성이 높다. 따라서 데이타의 복원 회수를 최소화하기 위해 복원된 데이타를 캐쉬 자료 구조에 저장하고 이를 재 사용하도록 구현하였다. 또한, 캐쉬의 적중율(hit ratio)을 높이기 위해 타일의 크기를 적절하게 조절하였는데, 실제 실험에 사용된 타일의 크기는 32×32 또는 16×16 이다. 이것은 다양한 타일 크기에 대한 반복적인 실험을 통해, 가장 빠른 렌더링 결과를 얻을 수 있는 타일 크기를 결정한 것이다.

캐쉬의 크기, 즉, 캐쉬가 저장할 수 있는 단위 블록의 개수를 적절히 조절하는 것도 성능 향상에 중요한 요소로 작용한다. 너무 작은 크기의 캐쉬는 평균 적중율(hit ratio)을 낮추어 렌더링 성능의 저하라는 결과를 낳는다. 반면, 지나치게 캐쉬의 크기를 크게 하는 것은 렌더링 성능은 향상시키지만 불필요하게 메모리 공간을 낭비하는 결과를 가져온다. 본 논문에서는 각 PE당 512 KByte를 캐쉬 메모리로 할당하여 사용하였는데, 이것은 복원된 단위 블록 64개를 동시에 저장할 수 있는 크기이다.

```
divide PEs into groups
assign tiles to each group pool
/* Parallel Processing of Each PE in a Group */
repeat
    repeat
        get a tile T from group pool
        render (T)
        send partial image to image master PE
    until group pool exhausts
    Join another group that hasn't finished its work
until all pools exhaust
```

알고리즘 2 병렬 렌더링 알고리즘

3.3 병렬 볼륨 렌더링으로의 확장

3.2절에서 설명한 볼륨 렌더링 알고리즘은 어렵지 않게 병렬 볼륨 렌더링 알고리즘으로 확장 가능하다. 제안하는 병렬 볼륨 렌더링 알고리즘은 타일과 단위 블록이 렌더링의 기본 단위가 된다. 3.2절에서 설명한 것과 같이, 볼륨 데이타는 압축이 되어 렌더링에 참여하는 모든 PE의 지역 메모리에 로드되며 렌더링에 참여하는 PE들은 1개 이상의 그룹으로 나누어진다. 렌더링 작업은 타일 단위로 각 그룹에 분배된다. 가장 좋은 렌더링 속도를 보여주는 그룹의 수는 실험적으로 결정하게 된다. 각 그룹에는 하나의 마스터(master) PE가 있어서 자신이 속한 그룹의 PE들이 렌더링 해야 할 타일의 풀(pool)을 관리한다. 마스터 PE를 포함한 그룹내의 모든 PE들은

마스터 PE로부터 렌더링 할 타일의 인덱스를 가져와서 렌더링을 수행하고, 결과로 나오는 부분 영상은 합성을 전담하는 영상 마스터 PE에게 보내진다. 타일의 렌더링이 끝나자마자 부분 영상을 영상 마스터 PE에게 보내기 때문에 Ma[4]의 접근 방법과는 달리 별도의 영상 합성을 위한 단계가 불필요하다.

특정 그룹에 할당된 모든 타일의 렌더링이 끝나면 그 그룹은 곧바로 해체되며 그 그룹에 속해있던 PE들은 아직 렌더링을 끝내지 못한 다른 그룹에 분산 배치되어 계속 렌더링을 수행한다. 이러한 방법으로 효과적인 동적 부하 균형(dynamic load balancing)을 얻을 수 있으며, 이 과정은 모든 타일에 대한 렌더링이 끝날 때까지 계속된다 (그림 5).

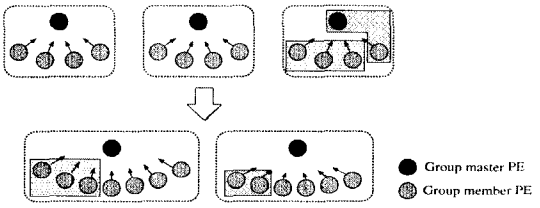


그림 6 병렬 렌더링에서 한 그룹의 작업이 끝났을 때 그룹의 재구성의 예

4. 실험 및 결과

본 논문에서 제안한 알고리즘은 MIMD 구조의 Cray T3E-900에서 구현되었다. 이 시스템은 DEC의 450 MHz ALPHA 21164 프로세서 136개를 장착하고 있으며, 각각의 PE는 128MB의 지역 메모리를 가지고 있다. 각 PE들은 고-대역폭(high-bandwidth), 저-지연(low-latency)의 양방향 3D 토러스 시스템 인터네트워크(bidirectional 3D torus system internetwork)로 연결되어 있다. PE간 통신은 SHMEM 라이브러리(Cray Shared Memory Access Library)[18]로 구현하였는데, PVM(Parallel Virtual Machine)이나 MPI (Message Passing Interface)보다는 사용이 까다롭지만 다음과 같은 장점을 갖는다.

① Cray의 하드웨어적 특성을 최대한 활용하기 때문에 통신 속도가 PVM, MPI에 비해 빠르다.

② 원격 PE의 작업을 중단시키지 않으면서 원격 PE의 지역 메모리를 읽고 쓸 수 있다.

③ 타일의 인덱스를 받거나 부분 영상을 전송할 때 발생할 수 있는 critical region 문제를 쉽게 해결해 준다.

실험에 사용한 데이타는 512×512×1440 해상도의 Visible Man fresh CT 데이타로서 원래의 720 MByte의 크기를 45.43 MByte로 압축한 것인데 (웨이블릿 계수 7%로 압축된 데이타), 이미 언급한 바와 같이 광선 투사법에 의해 생성된 결과 영상은 압축하지 않은 데이타를 렌더링한 결과 영상과 눈으로 구별하기 어려울 정도로 유사하다 (그림 6(a)와 (b)). 또한 그림 6(d)는 3%로 압축된 24.59 Mbyte 데이타에 대한 렌더링 영상이며, 이렇게 높은 압축율의 데이타를 이용하더라도 중요한 특징들은 거의 유지되고 있음을 확인할 수 있다

(a) 압축하지 않은 데이타의 최종 영상 (b) 7% 압축. 삼선형 보간

(c) 7% 압축. 최근 보간 (d) 3% 압축. 삼선형 보간

그림 6 생성된 최종 영상의 비교

구현된 알고리즘의 성능을 타일의 크기, 그룹의 수를 변화시키면서 측정하였다. 실험에 사용된 타일의 픽셀 해상도는 16×16 또는 32×32이고, 최종 영상 해상도가 512×1024인 그림 10(b)와 같은 피부 영상을 삼선형 보간(tri-linear interpolation)을 이용해 렌더링 하는데 소비되는 시간의 변화를 측정하였다. 그림 7는 PE의 개수의 증가에 따른 렌더링 시간의 변화를 보여주고 있다. 32×32 크기의 타일을 사용하고 삼선형 보간을 이용할 때, 한 개의 PE를 사용할 경우 209초가 걸리던 것이 96개의 PE를 사용할 경우 2.9초에 수행됨을 확인할 수 있다. PE수에 따른 성능 향상 결과는 그림 8과 같은데, 80개의 PE까지 80% 이상의 성능 향상 결과를 보여주고 있다. 이러한 수치는 직접, 간접 볼륨 렌더링 기법을

표 3 주요 병렬 볼륨 렌더링 알고리즘의 성능 비교

본 논문의 알고리즘			Montani[3]		Parker[23]	
	성능 향상	효율	성능 향상	효율	성능 향상	효율
1PE	1	1	1	1	1	1
32PE	30.06	0.94	28.11	0.88	30.8	0.96
64PE	60.34	0.94	54.52	0.85	57.8	0.90

이용하는 기존의 연구 결과[19,20,21,22]에서 얻어진 결과들과 비교 할 때 상당히 우수한 결과라 할 수 있다(표 3). Parker[23]는 공유메모리 구조(shared memory architecture) 병렬 컴퓨터상에서 등가면(iso-surfacing)에 대한 병렬 볼륨 렌더링을 수행하였고 상당히 우수한 렌더링 성능을 얻었다. 그러나 그의 방법은 공유메모리 병렬 컴퓨터만을 기반으로 하기 때문에 분산 메모리 구조의 병렬 컴퓨터 환경이나, PC/워크스테이션으로 이루어진 분산 환경에서는 이용될 수 없다는 한계를 갖고 있으나, 본 논문의 알고리즘은 어떤 병렬 컴퓨터 환경에서도 적용 가능하며 매우 방대한 데이타를 효과적으로 렌더링 하는 것을 기본 목표로 한다는 점에서 구별된다.

렌더링 시간은 볼륨 광선 투사법에서 현재의 샘플링 위치에서의 밀도값을 근사화(approximation)하기 위해 어떤 보간 방법을 사용하느냐에 따라 달라진다. 그림 7, 그림 8, 그림 9와 같은 실험 수치들은 삼선형 보간법을 이용해 얻어진 결과이지만, 동일한 상황에서 단순히 샘플링 위치에서 가장 가까운 복셀값을 사용하는 최근(nearest) 보간을 이용하는 경우에 화질과 렌더링 속도에 사이에 상반 관계(trade-off)가 존재한다. 최근 보간을 이용하여 그림 10(b)와 같은 해상도의 512×1024 피부 영상을 렌더링하는 경우, 삼선형 보간을 이용하는 경우보다 우수한 속도를 얻을 수 있었다 (한 개의 PE에서 72.5초, 88개의 PE에서는 0.9초). 특히, 표 6에서 보는 것과 같이 60개 이상의 PE를 사용할 때에도 90% 이상의 성능 향상을 얻을 수 있었다. 하지만 그림 6(b)와 (c)를 통해 확인할 수 있듯이, 삼선형 보간을 이용한 경우의 화질이 더 우수하다는 사실을 알 수 있다. 따라서, 렌더링 시스템을 구현할 때, 사용자의 요구에 따라 적절한 보간 방법을 선택할 수 있게 함으로써 사용자가 원하는 속도와 화질을 조절할 수 있을 것이다.

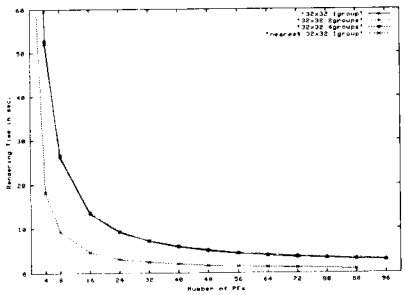


그림 7 렌더링 시간

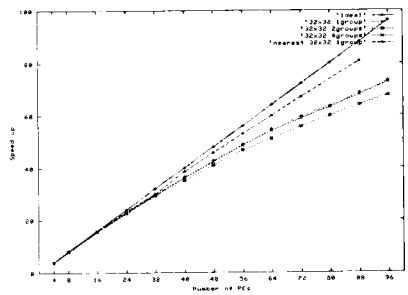


그림 8 성능 향상

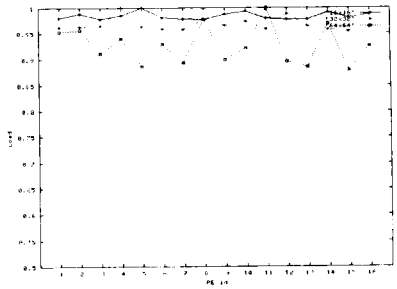
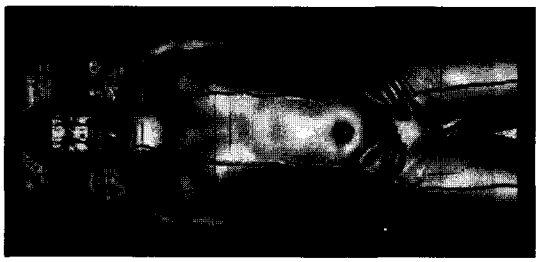


그림 9 부하균형

병렬 볼륨 렌더링 알고리즘에서 렌더링에 참여하는 PE가 고르게 작업을 수행할 수 있는 부하 균형(load balancing)을 유지하는 것은 병렬 렌더링 성능 향상을 위한 중요한 요소 중 하나이다. 부하 균형의 성능은 타일 크기에 따라 변하는데, 일반적으로 더 작은 크기의 타일을 이용하는 것이 더 좋은 부하 균형을 나타낸다. 하지만 작업(타일)이 지나치게 작게 분할될 경우에는 타일을 각 PE에게 할당하거나 렌더링 된 부분 영상을 모으는 일과 같은 추가적인 작업의 부담이 증가하게 된다. 본 병렬 렌더링 시스템에서는 16×16 또는 32×32 의 타일이 가장 좋은 성능을 나타내는 크기임을 실험을 통해 확인할 수 있었다. 그림 9는 16개의 PE를 단일 그룹으로 구성하고 타일 크기를 변화시키면서 (16×16, 32×32, 64×64) 렌더링을 수행했을 때, 통신, 복원 시간 등을 포함한 전체 렌더링 작업 시간에 대한 순수 렌더링 시간의 비율을 각 PE별로 나타낸 것이다. 타일의 크기가 작은 경우에는 각 PE가 비교적 고르게 작업을 할당

받게 됨을 알 수 있지만, 큰 경우에는 PE사이의 작업량의 차이가 비교적 크게 나타남을 확인할 수 있다. 현재, 각 PE에 할당되는 작업의 크기를 동적으로 변화시킴으로써 성능을 향상시키기 위한 연구가 진행 중이다.

(a) 뼈 영상



(b) 피부 영상

(c) 뼈, 피부, 내장 영상

그림 10 Visible Man의 광선 투사 영상

지금까지 설명한 바와 같이, 이렇게 향상된 렌더링 속도를 얻을 수 있는 이유는 구현된 압축 기반 병렬 렌더링 기법의 수행 과정에서 발생하는 데이타 통신 비용을 최소화하였기 때문이다. 실제 구현에서 렌더링 수행 중에 발생하는 통신 비용은 각 PE가 렌더링 해야 하는 타일의 인덱스를 전달하기 위한 경우와 할당받은 타일에 대한 부분 렌더링 영상을 보내기 위한 경우밖에 없으며, 이 시간은 렌더링에 소비되는 전체 시간과 비교할 때 매우 적은 부분을 차지한다. 실제로, 64개의 PE를 사용한 경우 전체 렌더링 시간에 대한 데이타 통신 시간의 평균 비율은 0.0001 이하이며 이 정도의 수치는 무시할 수 있을 정도로 작은 값이다. 따라서, 본 논문의 압축 기반 볼륨 렌더링 기법은 Cray와 같은 고성능 병렬 컴퓨터를 이용한 경우보다 상당히 느린 이더넷상에서 통신하는 범용의 PC/워크스테이션의 클러스터를 이용한 분산 볼륨 렌더링을 구현할 때 더 좋은 성능을 발휘할 수 있는 가능성을 제시한다.

5. 결론 및 향후 연구 방향

본 논문에서는 방대한 크기의 볼륨 데이타의 효과적인 가시화를 위해 분산 메모리 병렬 컴퓨터를 이용한 압축 기반 병렬 광선 투사법을 제안하였다. 구현된 알고리즘은 방대한 볼륨 데이타를 효과적으로 압축함으로써 병렬 렌더링 과정 중 병목 현상의 주된 원인이 되는 데이타 통신 오버헤드를 제거하여 속도 향상을 얻을 수 있음을 보였다. 이러한 접근 방법은 이더넷과 같이 비교적 느린 네트워크를 통해 통신하는 범용의 PC/워크스테이션의 클러스터로 이루어진 분산 환경에서 더 좋은 성능을 발휘할 수 있다. 또한, 영상-순서 및 객체-순서 볼륨 렌더링 알고리즘의 장점들과 기존의 다양한 속도 향상 기법들을 고려한 병렬 광선 투사 알고리즘의 개발을 통해 방대한 볼륨 데이타의 병렬 렌더링의 성능을 향상시킬 수 있었다.

현재, 압축을 수행한 후에도 PE의 지역 메모리에 로드할 수 없을 정도로 방대한 볼륨 데이타를 효과적으로 가시화하기 위한 기법을 연구 중에 있다. 이 알고리즘은 필요할 때마다 다른 PE에 저장되어 있는 데이타를 가져오도록 한다는 점에서 기존의 병렬 볼륨 렌더링 알고리즘과 유사하지만, 재분배되는 데이타들 자체가 단위 블록이라는 단위이고 이들 각각이 압축된 상태이기 때문에 데이타 통신 비용을 최소화할 수 있을 것이다. 또한, 분산 메모리 구조를 갖는 병렬 컴퓨터에 적합하도록 구현된 현재의 기법을 범용의 PC/워크스테이션의 클러스터에서도 적용할 수 있도록 확장하고 있다.

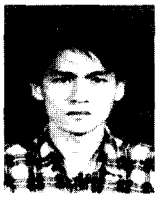
참 고 문 헌

[1] NLM, *http://www.nlm.nih.gov/research/visible/visible_human.html*, 1998.

[2] J. Nieh and M. Levoy, "Volume rendering on scalable shared-memory MIMD architectures," *1992 Workshop on Volume Visualization*, pp.17-24, 1992.

[3] C. Montani, R. Perego, and R. Scopigno, "Parallel rendering of volumetric dataset on distributed

memory architectures," *Concurrency: Practice and Experience*, Vol.5 No.2, pp.153-167, 1993.
[4] K. Ma, et. al., "Parallel volume rendering using binary swap compositing," *IEEE Computer Graphics and Applications*, Vol.14, No.4, pp.59-68, 1994.
[5] U. Neumann, "Communication costs for parallel volume-rendering algorithms," *IEEE Computer Graphics and Applications*, Vol.14, No.4, pp.49-58, 1994.
[6] C. Bajaj, I. Ihm, and S. Park, "3D RGB image compression for interactive applications," *TICAM Report 99-41*, The University of Texas at Austin, October 1999.
[7] I. Ihm and S. Park, "Wavelet-based 3D compression scheme for very large volume data," *Graphics Interface '98*, pp.107-116, Vancouver, Canada, June 1998.
[8] I. Ihm and S. Park, "Wavelet-based 3D compression scheme for interactive visualization of very large volume data," *Computer Graphics Forum*, Vol.18, No.1, pp.3-15, 1999.
[9] S. Park, G. Koo, and I. Ihm, "Wavelet-based 3D compression schemes for the visible human dataset and their applications," *The 2nd Visible Human Project Conference*, Maryland, USA. October 1998.
[10] C. Bajaj, I. Ihm, G. Koo and S. Park, "Parallel ray casting of visible human on distributed memory architectures," *VisSym '99: Joint EUROGRAPHICS-IEEE TCVG Symposium on Visualization*, pp.269-276, Vienna, Austria, May 1999.
[11] C. Bajaj, I. Ihm and S. Park, "Making 3D texture practical," *Pacific Graphics '99*, pp.259-268, Seoul, Korea, October 1999.
[12] M. Levoy, "Display of surface from volume data," *IEEE Computer Graphics and Applications*, Vol.8, No.3, pp.29-37, 1998.
[13] M. Levoy, "Efficient ray tracing of volume data," *ACM Transactions on Graphics*, Vol.9, No.3, pp.245-261, 1990.
[14] D. Cohen and Z. Sheffer, "Proximity clouds-an acceleration technique for 3D grid traversal," *The Visual Computer*, Vol.11, pp.27-38, 1994.
[15] K. Subramanian and D. Fussell, "Applying space subdivision techniques to volume rendering," *IEEE Visualization '90*, pp.150-159, October 1990.
[16] L. Westover, "Footprint evaluation for volume rendering," *Computer Graphics*, Vol.24, No.4, pp.367-376, 1990.
[17] R. Lee and I. Ihm, "On enhancing the speed of splatting using both object- and image-space coherence," *Graphical Models*, Vol.62, No.4, pp.263-282, July 2000.
[18] *SHMEM User's Guide*, Cray Research Inc., 1994.
[19] M. Amin, A. Grama and V. Singh, "Fast volume rendering using an efficient scalable parallel formulation of the shear-warp algorithm," *The 1995 Parallel Rendering Symposium*, pp.7-14, Atlanta, October 1995.
[20] P. Lacroute, "Real-time volume rendering on shared memory multiprocessors using the shear-warp factorization," *The 1995 Parallel Rendering Symposium*, pp. 15-22, Atlanta, October 1995.
[21] A. Law and R. Yagel, "Multi-frame thrashless ray casting with advancing ray-front," *Graphics Interface '96*, pp. 70-77, Toronto, Canada, May 1996.
[22] P. Li, S. Whitman, R. Mendoza, and J. Tsiao. "ParVox - a parallel splatting volume rendering system for distributed visualization," *The 1997 Symposium on Parallel Rendering*, pp. 7-14, Phoenix, October 1997.
[23] S. Parker, M. Parker, Y. Livnat, P. Sloan, C. Hansen, and P. Shirley, "Interactive ray tracing for volume visualization," *IEEE Transactions on Visualization and Computer Graphics*, Vol.5, No.3, pp.238-250, 1999.



구 기 범
1997년 2월 서강대학교 전자계산학과 졸업(공학사). 1999년 2월 동 대학원 컴퓨터학과 졸업(공학석사). 현재 (주)인젠 연구원. 관심분야는 볼륨 렌더링, 병렬 및 분산 처리 등임.

박 상 훈
1993년 2월 서강대학교 수학과 졸업(이학사). 1995년 2월 동 대학원 전자계산학과 졸업(공학석사). 2000년 2월 동 대학원 컴퓨터학과 졸업(공학박사). 현재 The University Texas at Austin 박사 후연구원. 관심분야는 컴퓨터 그래픽스, 과학적 가시화, 영상처리 등임.

송 동 섭
1998년 2월 서강대학교 컴퓨터학과 졸업(공학사). 2000년 2월 서강대학교 컴퓨터학과 졸업(공학석사). 현재 LG전자 연구소 재직. 관심분야는 볼륨 렌더링, 병렬 및 분산 처리 등임.

임 인 성

1985년 2월 서울대학교 계산통계학과 졸업(이학사). 1987년 5월 Rutgers University 전자계산학과 졸업(이학석사). 1991년 Purdue University 전자계산학과 졸업(이학박사). 1999년 7월 ~ 2000년 6월 The University Texas at Austin의 TICAM (Texas Institute of Computational and Applied Mathematics) 방문연구원. 1993년 3월 ~ 현재 서강대학교 컴퓨터학과 부교수. 관심분야는 컴퓨터 그래픽스, 과학적 가시화, 고성능 계산 등임.