순차 Shear-Warp 알고리즘을 이용한 병렬볼륨렌더링의 구현

김 응 곤†

요 약

본 논문에서는 볼륨렌더링을 위한 빠른 병렬 알고리즘을 제안하고, 이를 4,096개의 프로세서를 가진 MasPar MP-2 범용병렬 컴퓨터에서 C 언어와 MPL(MasPar Programming Language) 언어를 이용하여 구현하였다. 본 알고리즘은 현재 가장 빠른 순차 볼륨 렌더링 알고리즘으로 알려진 Lacroute의 Shear-Warp 알고리즘을 병렬화한 것이다.

본 병렬 알고리즘은 밀림변환 공간 분할 기법과 이전의 렌더링 단계에서 얻은 부하정보를 이용하여 다음 렌더링시 부하를 균형화하는 부하균형화 기법을 이용함으로써 통신 오버헤드를 줄이며, 연속길이부호화 기법에 의한 볼륨 데이타 구조를 이용함으로써 처리할 복셀의 수를 크게 줄인다.

MasPar MP-2에서 128×128×128 복셀로 구성된 인체 두뇌 볼륨 데이타세트에 대하여 실험한 결과 초당 3~4 프레임의 속도로 렌더링하였으며, 본 알고리즘의 확장성에 의하여 16,384 개의 프로세서를 가진 MasPar MP-2 시스템에서는 초당 12~16 프레임의 렌더링이 가능할 것으로 기대된다. 또한 더 큰 볼륨에 대해서도 최근의 SIMD 또는 MIMD 머신상에서는 초당 30~60 프레임의 렌더링이 가능할 것으로 기대된다.

Implementation of Parallel Volume Rendering Using the Sequential Shear-Warp Algorithm

Eung-Kon Kim†

ABSTRACT

This paper presents a fast parallel algorithm for volume rendering and its implementation using C language and MPL(MasPar Programming Language) on the 4,096 processor MasPar MP-2 machine. This parallel algorithm is a parallelization based on the Lacroute's sequential shear-warp algorithm currently acknowledged to be the fastest sequential volume rendering algorithm.

This algorithm reduces communication overheads by using the sheared space partition scheme and the load balancing technique using load estimates from the previous iteration, and the number of voxels to be processed by using the run-length encoded volume data structure.

Actual performance is 3 to 4 frames/second on the human brain scan dataset of 128×128×128 voxels. Because of the scalability of this algorithm, performance of 12~16 frames/second is expected on the 16,384 processor MasPar MP-2 machine. It is expected that implementation on more current SIMD or MIMD architectures would provide 30~60 frames/second on large volumes.

✱ 이 연구는 한국과학재단 1997년도 전반기 해외 Post-Doc. 연수지원에 의하여 수행되었음.

† 정 회 원 : 순천대학교 컴퓨터과학과 교수

논문접수 : 1997년 8월 29일, 심사완료 : 1998년 4월 17일

1. 서 론

기하학적 모델링(Geometric Modelling), 과학적 시각화(Scientific Visualization) 및 가상현실(Virtual Reality)을 비롯한 컴퓨터 그래픽스의 여러 응용 분야에서는 볼륨데이타를 현실감있게 렌더링하여 대화식 또는 실시간으로 디스플레이하는 것이 요구된다. 그러기 위해서는 방대한 양의 볼륨 데이타세트를 고속으로 처리해야 하므로 메모리 용량, 메모리 대역폭, 연산속도 등이 큰 문제가 된다.

여러 렌더링기법중 광선추적법[1,2]은 프로그래밍하기가 간단하면서도 반사, 굴절, 그림자 등과 같은 실감있는 장면의 효과를 쉽게 낼 수 있으나, 주어진 장면에 대해 모든 광선과 물체가 교차하는지 검사해야 하기 때문에 계산시간이 많이 소요되는 문제가 있다. 예를 들어 복셀당 16비트, 256×256×256 개의 데이타세트를 초당 30 프레임(frame)의 속도로 렌더링하기 위해서는 32MB의 메모리, 초당 1GB의 메모리 전송률과 초당 대략 500억개의 명령어 수행이 요구된다[3,4].

이러한 문제를 극복하기 위한 방법[5]으로는 원래의 데이타를 간소화하여 데이타를 줄이는 방법, 소프트웨어에 기반을 둔 최적화 알고리즘 및 가속화 방법, 그래픽스 전용 하드웨어나 렌더링을 위한 아키텍처로 구현하는 방법, 그리고 범용 병렬 머신상에서 구현하는 방법 등이 있다. 데이타를 줄이는 방법은 영상의 질이 떨어지며, 가속화 기법은 많은 메모리를 요구하며, 여전히 대화식이나 실시간으로 처리하는데는 한계가 있다. 렌더링 아키텍처에 의하여 구현하는 방법은 특수 응용분야에만 활용된다.

Cabral[6]은 그래픽스 전용 하드웨어인 SGI Reality Engine에서 볼륨 렌더링을 구현하였으며, 볼륨 렌더링 전용 렌더링 하드웨어는 Meagher[7]의 볼륨 렌더링 가속기와 Kaufman[8]의 Cube 아키텍처가 있다. 범용 병렬 컴퓨터에서의 볼륨 렌더링은 Amin 등[9]이 1,024개 프로세서의 CM-5 머신에서, Lacroute [10]가 16개 프로세서의 SGI Challenge에서 각각 구현하는 등, 최근 범용 병렬 컴퓨터를 이용한 실시간 볼륨렌더링에 관한 연구가 활발히 진행되고 있다.

본 논문에서는 일반적으로 오랜 컴퓨터 처리시간이 요구되는 광선추적기법에 의한 볼륨렌더링의 속도를 빠르게 하기 위하여 기존의 순차 렌더링 알고리즘중 가장 빠른 것으로 알려진 Shear-Warp 알고리즘[4]을 4,096개 프로세서로 구성된 범용 병렬 컴퓨터인 MasPar MP-2상에서 병렬화한다.

볼륨 데이타세트로부터 빈 복셀을 제거하여 메모리의 사용을 크게 줄이기 위한 볼륨 데이타 구조와 통신 오버헤드를 줄이기 위한 공간 분할 기법과 부하 균형화 기법을 제안하고, 이를 구현하여 인체의 CT 스캔 데이타세트와 MRI 스캔 데이타세트를 대상으로 실험한 결과에 대하여 논한다.

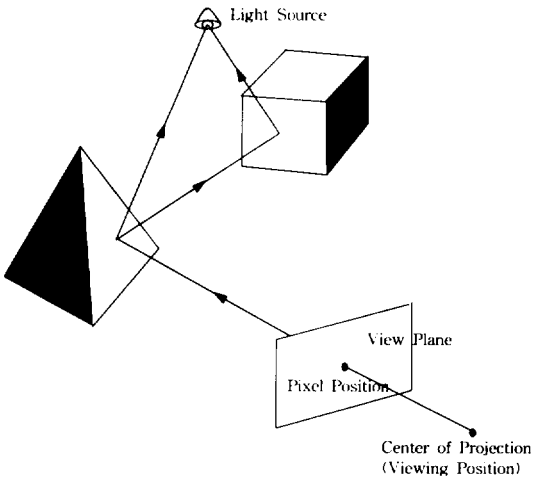
2장에서는 광선추적법과 순차 Shear-Warp 알고리즘에 대하여, 3장에서는 본 논문의 병렬 볼륨 렌더링 알고리즘에 대하여, 4장에서는 알고리즘의 분석과 MasPar MP-2에서 구현한 결과에 대하여 각각 논하며, 5장에서는 결론을 맺는다.

2. 광선추적법과 Shear-Warp 알고리즘

광선추적 알고리즘[1,2]은 시각 시스템으로 인식할 수 있는 모든 빛을 추적하는 방법으로 보통 전진방향추적(forward ray tracing)과 후진방향추적(backward ray tracing)방법으로 이루어진다. 전진방향추적에서는 광원의 입장에서 광선을 추적하여 나가기 때문에 자연에서 일어나는 현상을 정확하게 모델링하고 있으나 반사되는 빛들의 대부분이 산란되는 경우 계산상의 많은 낭비를 가져오게 된다. 후진방향추적방법은 이러한 문제를 극복하여 속도향상을 위한 광선추적 알고리즘에 적용되고 있다.

관찰자의 눈과 시야에 들어오는 화면, 그리고 실제 물체들이 조명을 받고 있는 상황에서 후진방향추적하는 방법을 다음과 같이 설명할 수 있다[2]. 추적하고자 하는 광선을 관찰자의 눈에서 출발하여 화면의 한점을 통과하여 지나가게 한다. 이 빛이 실제 물체들과 교차하는 지를 계산하고 교차하는 경우, 그 물체의 특성에 따라 반사, 굴절 및 투과 현상을 시뮬레이션하고 각기 합당한 빛의 강도를 계산한다. 이러한 반사, 굴절 및 투과된 빛들은 또 다시 다른 물체와 교차하는지를 계산하고 이와 같은 과정을 계속 반복한다. 결국 빛의 강도가 문턱값 이하로 약해지거나 빛이 어느 물체와도 교차하지 않았을 때 그 화면에 대한 점의 빛 역추적 과정이 끝나게 된다. 이 방법을 화면상에 존재하는 모든 화소에 적용해서 완전한 영상을 구성하게 된다. 그림 1은 광선추적 알고리즘에서 빛을 역추적하는 간단한 광선의 경로를 나타내고 있다. 이를 이용한 대표적인 순차 알

고리즘은 한 화소별로 빛을 역추적하여 전체의 영상을 만들어 내기 때문에 화면의 서로 다른 화소간의 계산과정은 상호 독립적이다. 따라서, 모든 화소들을 각 프로세서에 분산시켜 계산하면 병렬성을 최대로 이용할 수 있다는 장점이 있다[11,12]. 자료의 분산을 한 화소의 단위로 할 때 모든 점들을 한 개의 프로세서에 할당하여 계산을 하고 그 결과를 제어기에 보내어 조합하는 경우에 최대의 효과를 얻을 수 있다[2].



(그림 1) 광선추적 과정
(Fig. 1) Ray Tracing

Lacroute와 Levoy에 의해 발표된 Shear-Warp 알고리즘[4]은 볼륨 공간과 영상 공간의 응집성(coherence)을 이용하면 최적화가 가능하므로 현재까지 가장 빠른 순차 볼륨렌더링 알고리즘으로 알려져 있다.

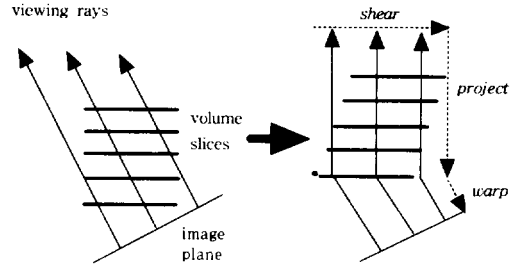
볼륨렌더링 알고리즘은 물체순서(object-order) 알고리즘과 영상순서(image-order) 알고리즘으로 분류되며, 물체공간에서 영상공간으로 사상(mapping)시킬 때 물체순서 렌더링 알고리즘에서는 필터링과 투영이 매우 복잡하게 되며, 영상순서 알고리즘에서는 부가적인 산술연산이 요구된다. 이러한 문제를 해결하기 위하여 Shear-Warp 알고리즘에서는 그림 2와 같이 중간 좌표계를 도입하여 볼륨을 이 좌표계로 밀림변환하여 2차원 영상으로의 투영을 매우 효율적으로 수행할 수 있게 한다. 그림 2에서 수평선들은 단면으로 보여지는 볼륨데이타의 슬라이스들을 나타낸다. 볼륨을 슬라이스들에 수평방향으로 밀림변환하게 되면 슬라이스들과 관측광선들이 관측방향에 각각 수직이 된다. 밀림변환이 슬라이스들에 평행하게 이루어 지므로 변환연산을 매우 간단하고 효율적으로 수행할 수 있게 된다. Shear-Warp 알고리즘에 의한 볼륨렌더링 절차는 다음과 같으며, 그림 3은 이를 설명하고 있다.

(1) 각 슬라이스를 관측방향에 수직이 되도록 평행이동시킨다. 이 때 볼륨을 관측방향에 수직인 슬라이스들에 평행이 되도록 밀림변환하며, 그 결과 관측광선들은 각 슬라이스에 수직이 된다.

(2) 가능한 3가지 슬라이스 방향중 관측방향에 거의 수직인 슬라이스들을 선택한다.

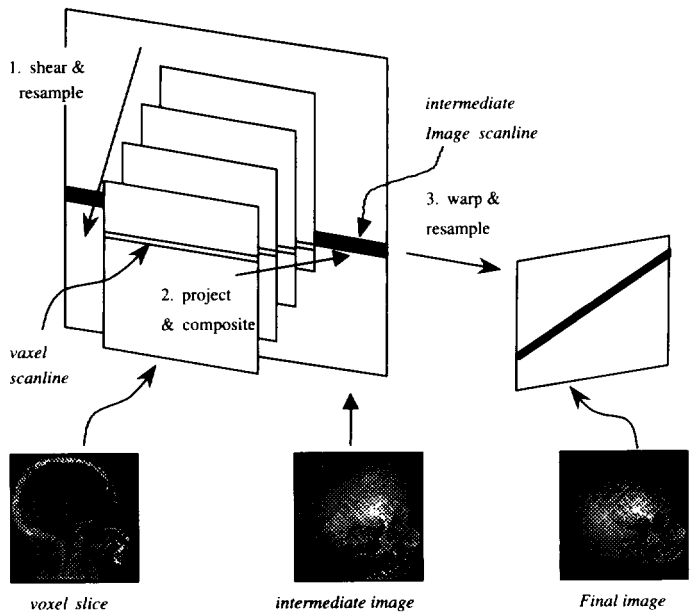
(3) 표본화한 슬라이스들을 "over" 연산자를 이용하여 앞에서부터 뒤로(front-to-back order) 합성한다. 이 단계에서 밀림 변환된 물체공간으로 투영하여 생성된 영상을 중간영상(intermediate image)이라고 정의한다.

(4) 단계 (3)에서 생성된 중간 영상을 warping하여 영상공간으로 변환시킨다. 이 단계에서 생성된 영상을 최종영상(final image)이라고 정의한다.



(그림 2) Shear-Warp 알고리즘의 변환과정
(Fig. 2) Transformation for Shear-Warp Algorithm

단계 (1)과 (3)에서 필요한 물체공간 좌표계로부터 중간영상 좌표계로의 변환은 평행이동과 밀림변환이므로 연산이 단순하며, 2차원 영상이 3차원 볼륨보다 훨씬 작으므로 단계 (4)에서와 같이 2차원 중간 영상을 warping함으로써 연산을 효율적으로 수행할 수 있다. Shear-Warp 알고리즘은 음영데이타를 이용하여 음영 lookup 테이블을 만드는 전처리 단계, 볼륨 슬라이스들을 중간 영상에 투영하는 합성단계 및 중간영상을 warping하는 단계 등 세 가지 주요 기능으로 구성되며, SGI Indigo R4000 워크스테이션에서 순차알고리



(그림 3) Shear-Warp 알고리즘
(Fig. 3) Shear-Warp Algorithm

즘을 수행하는데 걸리는 시간을 표 1에 나타내었다[4]. 이 표의 예는 256×256×167 복셀의 인체 두뇌 데이타세트를 256×256 영상크기로 나타낼 경우인데 1 프레임을 렌더링하는데 1.09초의 시간이 소요됨을 볼 때 대화식 또는 실시간으로 렌더링하기 위해서는 효율적으로 병렬화하여야함을 알 수 있다.

〈표 1〉 순차 Shear-Warp 알고리즘의 수행시간[4]
〈Table 1〉 Rendering Time by Sequential Shear-Warp Algorithm

알고리즘 기능	소요시간(msec)
음영 테이블 계산	70
합성	900
warping	120
계	1090

(시스템 : SGI Indigo R4000 워크스테이션)

3. 병렬 볼륨렌더링

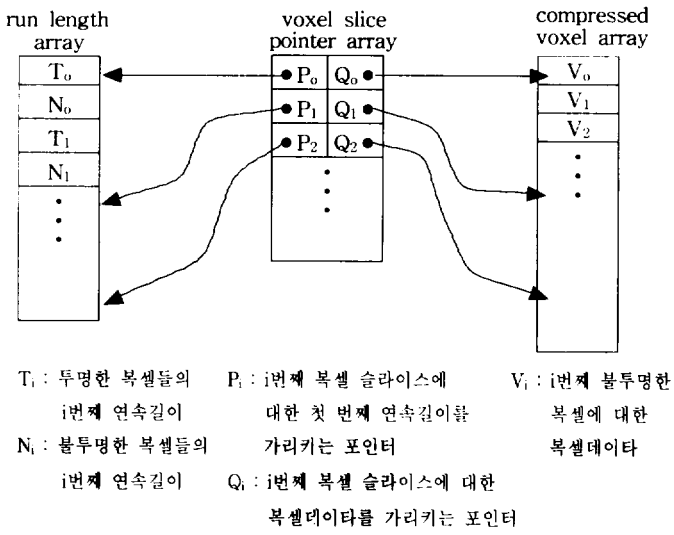
순차 볼륨렌더링 알고리즘중 가장 빠른 것으로 알려진 Shear-Warp알고리즘[4]을 범용 병렬 컴퓨터에서 병렬화하기 위한 알고리즘에 대하여 논한다.

병렬 알고리즘에서는 동시성(concurrency), 통신 오버헤드의 최소화 및 프로세서사이의 부하를 균형화하는 것이 중요하다. 각 프로세서에 전체의 볼륨을 복사하는 것이 가능하다면 통신문제를 쉽게 해결할 수 있으나 전반적인 메모리 요구량을 증가시키므로 비효율적이다. 또한 각 스캔라인은 매우 다양한 계산량을 가지므로 하나의 스캔라인을 한 프로세서에 할당하는 것은 심한 부하의 불균형을 초래할 수 있다. 그러므로 스캔라인들을 프로세서에 할당하는 효율적인 부하 균형화기법이 필요하다. 이 장에서는 주어진 복셀 데이타세트로부터 빈 복셀을 제거하고 렌더링을 빠르게 수행하기 위한 볼륨의 데이타 구조와 볼륨 슬라이스 합성기법에 대하여 논하며, 기존의 볼륨 공간 분할 및 상 공간 분할 기법에 대하여 서술하고, Shear-Warp 알고리즘을 병렬화하기 위하여 본 논문에서 구현한 밀림변환 볼륨 공간 기법과 부하균형화 기법에 대하여 논한다.

3.1 볼륨의 데이타 구조와 볼륨 슬라이스 합성

(1) 볼륨 데이타 구조

광선추적시 주어진 복셀 데이타세트로부터 투명한



(그림 4) 볼륨 데이타 구조
(Fig. 4) Data Structure of a Volume

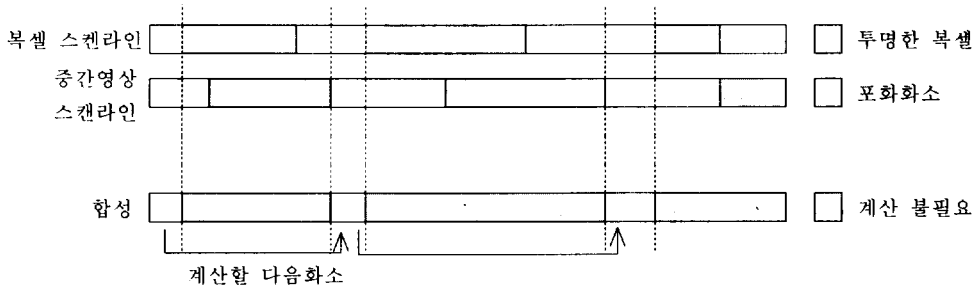
빈 복셀을 제거하고 포화도에 도달한 화소들을 건너 뛰게 함으로써 처리할 복셀의 수를 크게 줄이기 위하여 렌더링하기 이전에 전처리 단계에서 볼륨을 연속길이 부호화 기법(run-length encoding)을 이용하여 부호화하였다. 볼륨의 데이타 구조를 그림 4와 같이 연속길이 배열(run-length array), 복셀 슬라이스를 가리키는 포인터 배열(voxel slice pointer array) 및 압축 복셀 배열(compressed voxel array)의 3 가지 배열로 구성하였다.

연속길이 배열은 투명한 빈 복셀들의 연속길이(T_i)와 불투명한 복셀들의 연속길이(N_i)를 저장하며 각 연속길이는 1 바이트 크기로 한다. 복셀 배열은 모든 불투명한 복셀을 저장하며, 메모리 소비를 줄이기 위하여 투명한 복셀들은 저장하지 않는다. 볼륨을 구성할 때는 연속길이 배열을 용하여 투명한 복셀들을 압축된 불투명한 복셀들 사이에 끼워 넣으면 된다. 복셀배열의 각 복셀은 불투명도와 음영처리를 위해 필요한 음영 데이타를 포함한다. 음영 데이타는 렌더링 알고리즘이 복셀의 색상을 계산하는데 사용된다. 복셀의 색상은 관측지점과 광원의 위치에 따라 달라지므로 미리 계산할 수 없다. 불투명도는 1 바이트로 표현하였으며, 음영 데이타는 사용자가 구성할 수 있게 하였다. 음영 데이타를 3 바이트로 표현하여 복셀배열에서 각 복셀을 4 바이트로 표현하였다. 연속길이 부호화 기법으로 부호화된 볼륨을 처리할 때 문제점은 부호화된 복셀 데이타에 직접 접근이 어렵다는 점과 복셀이 부호화된 순서대로만 순회할 수 있다는 점이었다. 각 복셀에 직접접근이 가능하게 하기 위하여 복셀 슬라이스 포인터 배열에 볼륨의 한 슬라이스에 대한 첫 번째 투명한 복셀과 불투명한 복셀을 각각 가리키는 포인터를 저장하였다. 또한 렌더링하기 이전에 미리 볼륨을 세 개의 관측축에 대하여 각각 하나씩 연속길이 부호화함으로써 관측변환에 따라서 적절한 것을 선택할 수 있도록 하였다. 세 개의 볼륨을 동시에 메모리에 로드하더라도 볼륨 부호화시에 투명한 빈 복셀은 저장하지 않았으므로 원래의 볼륨보다 더 커지지 않는다.

압축 복셀배열은 각 복셀 슬라이스에 대하여 불투명한 복셀들의 데이타만을 저장한다. 각 복셀 슬라이스는 많은 투명한 복셀을 가질 수 있으므로 그림 4와 같은 연속길이 부호화 기법에 의한 데이타구조를 이용함으로써 투명한 복셀 데이타는 저장하지 않고 불투명한 복셀만을 압축 복셀 배열에 저장한다.

(2) 볼륨 슬라이스 합성

렌더링과정에서 중간영상의 불투명도가 일정한 경계값을 넘으면 그 다음의 볼륨 슬라이스들은 최종영상에 더 이상 변화를 주지 않으며, 투명한 빈 복셀들은 중간영상에 더 이상 변화를 주지 않으므로 렌더링할 때 불



(그림 5) 볼륨 슬라이스 합성
(Fig. 5) Composition of Volume Slices

투명도가 일정한 경계값을 넘은 포화화소와 투명한 빈 복셀들에 대해서 계산하지 않고 건너뛰게 하면 매우 효율적이다.

중간영상의 포화화소를 계산하지 않고 건너뛰기 위해서 그림 5의 데이타 구조와 같이 그 다음 불포화화소를 가리키는 포인터를 두어 구현하였다. 또한 투명한 복셀들을 계산하지 않고 건너뛰기 위해서 그림 4와 같은 데이타 구조를 이용하였다. 볼륨 슬라이스를 합성할 때는 그림 5와 같이 두 포인터를 따라서 불투명 복셀과 불포화화소만을 고려하면 된다. 즉, 그림 5에서 복셀 스캔라인에서의 투명한 복셀은 중간 영상에 더 이상 영향을 주지 않으며, 중간영상 스캔라인에서 불투명도가 일정한 값을 초과하는 포화화소는 최종영상에 더 이상 변화를 주지 않으므로 볼륨 슬라이스를 합성할 때 그림 5와 같은 데이터구조를 이용하여 중간영상의 포화화소나 투명한 복셀을 건너 뛰도록 한다.

3.2 데이타 분할 기법

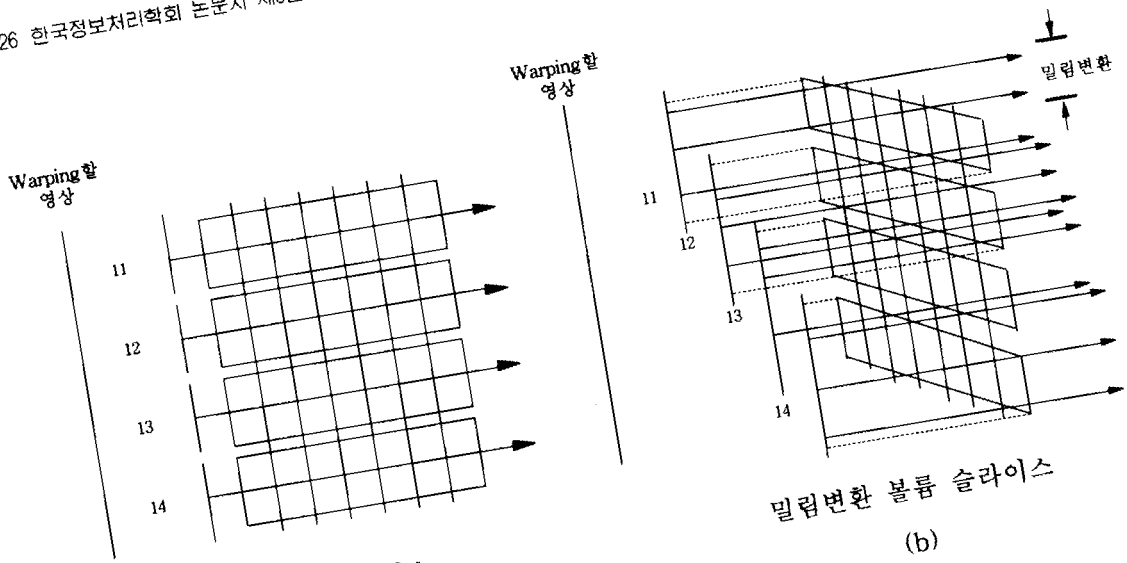
(1) 볼륨 공간 분할 기법과 영상 공간 분할 기법

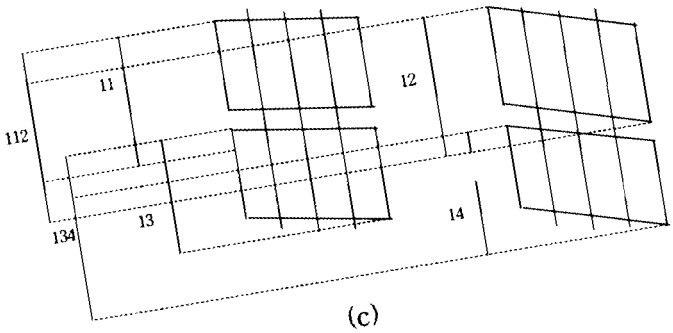
데이타 분할 기법은 볼륨 공간 분할 기법과 영상 공간 분할 기법이 있다[8]. 볼륨 공간 분할 기법에서는 볼륨 데이타를 여러 프로세서사이에 분할시킨다. 프로세서의 수가 p일 때 볼륨 데이타를 p 부분으로 나눈다. 각 프로세서는 할당받은 볼륨 세그먼트에 광선을 통과시킨다. 그 결과, 그림 6과 같이 프로세서가 4개인 경우 중간영상 I1, I2, I3, I4를 합성한다. 이 알고리즘은 볼륨에 밀림변환이 없을 때 잘 동작하며, 그림 6(a)와 같은 경우이다. 그러나 볼륨에 밀림변환이 있을 때는 심한 오버헤드가 발생하며, 그림 6(b)와 같이 중간영상의 중첩부분에 대한 통신 오버헤드가 발생한다. 각 프로세서에 두 개의 스캔라인이 할당되고 밀림변환의 각도가 45도인 경우를 고려해보자. 총 n개의 볼륨 슬라이스가 있고 밀림변환이 n개의 스캔라인에 대하여 이루어지는 경우, 각 프로세서는 두 개의 스캔라인을 계산한 후에 n개의 스캔라인에 해당하는 중간 영상 데이타를 통신해야 하므로 심한 오버헤드가 발생한다. 이 경우 계산상의 오버헤드도 발생한다. 중간 영상 I2의 일부가 I1에 의해서 가리게 되므로 그 중첩된 부분에 대해서 계산상의 낭비에 의한 오버헤드가 생기며, 프로세서의 수가 커질 때 주요 오버헤드가 된다. 밀림변환이 있을 때의 또 다른 볼륨 공간 분할 기법은 그림 6(c)와 같이 각 슬라이스를 분할하고 슬라이스 자체도 프로세서 사이에 분할시키는 방법이다. 그림의 경우 처음 n/2개 슬라이스의 상위 n/2개의 스캔라인들이 처음 프로세서에 할당되고, 마지막 n/2개 슬라이스의 스캔라인들이 두 번째 프로세서에 할당된다. 이 분할 기법은 많은 수의 프로세서를 사용할 수 있게 하지만 계산상의 낭비에 의한 오버헤드가 매우 심하다.

영상 공간 분할 기법은 최종 영상을 스캔라인에 따라 나누고 서로 다른 프로세서에 그것들을 할당하는 방법이다. 이 방법은 프로세서의 수가 많아짐에 따라 스캔라인들이 많은 세그먼트로 분할되어 결국 볼륨과 영상의 응집성(coherence)의 장점을 이용하지 못하므로 Shear-Warp 알고리즘을 이용하는 렌더링의 성능향상에 도움이 되지 않는다.

(2) 밀림변환 볼륨 공간 분할 기법

볼륨 공간 분할 기법의 주요 단점은 효과적으로 광선추적을 조기에 종료시킬 수 없기 때문에 과도한 계산

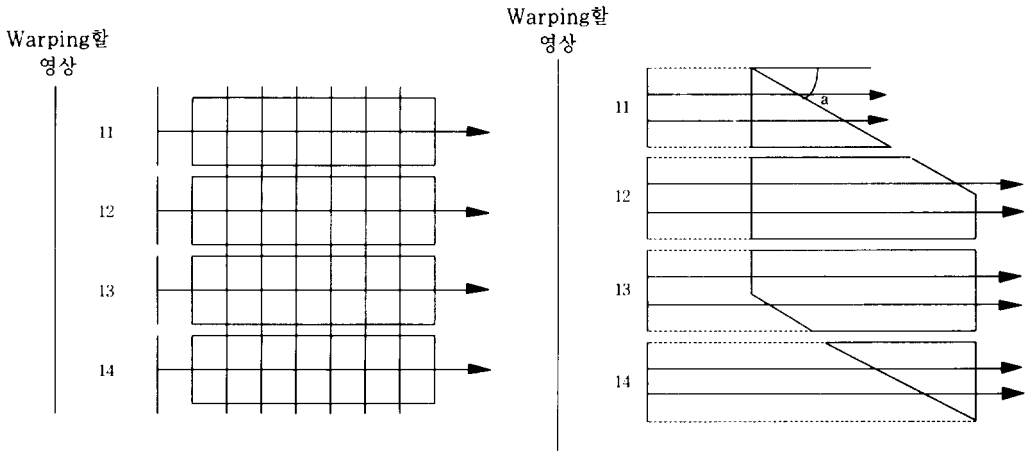


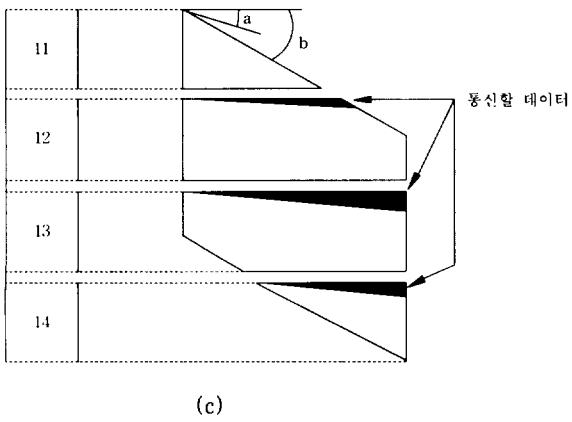


(그림 6) 볼륨 공간 분할
(Fig. 6) Volume Space Partition

을 필요로 한다는 점이다. 이러한 문제를 해결하기 위
한 한 가지 방법은 광선들에 평행한 평면으로 분할하는
것이다. 따라서 본 논문에서는 그림 7과 같이 어떠한
분할 평면이든지 스캔라인에 평행하게 볼륨 공간을 분
할하는 기법을 이용하였다. 이 분할 기법에서는 볼륨을
먼저 밀림변환시키고 볼륨 슬라이스들에 수직인 방향으
로 분할하여 각 프로세서가 할당받은 볼륨 세그먼트에
광선을 통과시킨다. 그 결과 각 프로세서에서 얻어지는
중간 영상들은 서로 중첩된 부분이 없으며 서로 독립적
으로 warping할 수 있다. 그림 7(a)는 밀림변환이 없
는 경우이며, 그림 7(b)는 밀림변환이 있는 경우이다.

이 알고리즘의 주요 오버헤드는 볼륨이 밀림변환될 때 볼륨 데이타의 통신이며, 그림 7(c)는 통신
드를 나타내고 있다. 기존의 볼륨공간 분할기
공간 분할 기법은 중간영상에 중첩된 부분이
나 프로세서의 수가 많아짐에 따라 스캔라인
세그먼트로 분할되어 통신 오버헤드가 발생
문의 밀림변환 볼륨 공간 기법에서는 각 프
얻어지는 중간영상이 서로 중첩되지 않도록
통신할 데이타의 양이 줄어든다. 그림 7
밀림변환이 a에서 b로 변화될 때 검은
인들이 통신된다. 통신의 양은 두 방향
인 밀림변환의 함수이다. 실시간 렌더링
움직임을 자연스럽게 표현해야 하므로
의 밀림변환은 매우 작으며, 따라서 이





(그림 7) 밀림변환 공간 분할 기법
(Fig. 7) Sheared Space Partition

한 통신 오버헤드는 중요한 문제가 되지 않는다.

3.3 부하 균형화 기법

광선추적시 빈 복셀들과 포화도에 이른 픽셀들을 계산하지 않고 건너 뛰게 하여 수행속도를 빠르게 할 수 있도록 본 논문에서는 스캔라인들에 대해서 연속길이 부호화 기법을 사용하였다. 서로 다른 스캔라인들은 매우 다양한 계산량을 가지며, 한 스캔라인에 대한 계산량을 미리 정확히 결정하는 것은 불가능하므로 부하를 결정하고 부하를 균형화하기 위해서는 휴리스틱 방법을 사용해야 한다. 따라서 본 논문에서는 다음과 같이 부하의 불균형을 최소화할 수 있도록 스캔라인을 프로세서에 할당하는 부하 균형화 기법을 적용하였다.

(1) 연속적인 두 프레임 사이에 볼륨이 크게 변화하지 않으므로 다음 렌더링 단계를 처리하는 순환동안 이전의 렌더링 순환 단계에서 계산된 부하정보를 사용한다.

(2) 스캔라인을 처리하는데 걸리는 시간을 부하의 양으로 하기 위해서 각 프로세서는 자신에게 할당되는 스캔라인들을 처리하는데 걸리는 시간을 계산한다.

(3) 전체 부하와 각 프로세서에서의 부하를 알게 되므로 프로세서는 새로운 데이타 분할을 독립적으로 계산한다.

3.4 알고리즘 개요

순차 Shear-Warp 알고리즘을 병렬화하는 본 논문의 병렬볼륨렌더링 알고리즘의 각 프로세서에 대한 제어흐름을 의사코드로 표현하면 그림 8과 같다.

먼저 각 프로세서는 음영정보를 만들어 lookup 테이블로부터 선택하고(Select_Lookup_Table) 다른 프로세서들이 끝내기를 기다린 후(Barrier), 태스크 큐로부터 태스크를 선택하여 처리할 중간영상을 계산한다(Get_Task). 태스크는 각 볼륨 슬라이스에 대하여 할당받은 중간영상과 교차하는 복셀 스캔라인을 찾은 다음(Intersect) 그 스캔라인들을 재표본화하고 합성한다(Resample). 모든 태스크들이 완료되었을 때(flag=1) 프로세서는 동기화하고(Barrier) 영상을 warping한 후(Warping) 다시 동기화하여(Barrier) 최종적으로 영상을 디스플레이한다(draw).

```
procedure PVR()
Select_Lookup_Table(); //select a portion of a
lookup table for shading voxels
Barrier //wait for other processor to finish
while(!flag)
    image_scanelines=Get_Task();   //sheared
    object partition
    if(scanlines!=NULL)
        Clear(image_scanlines);
        foreach(voxel_slice from front to
        back)
            voxel_scanlines=Intersect
            (image_scanlines,voxel_slice);
            //find the voxel scanlines that
            intersect the intermediate image
            image_scanlines=image_scanlines
            over
                    Resample(Shade(voxel
                    _scanlines));
            //resample and composite the
            voxel scanlines
        end
    else
        flag=1;
    endif
end
Barrier //synchronize
final_image=Warping(); //warp the image
Barrier //synchronize
Draw(final_image); //draw the image
end
```

(그림 8) 프로세서 제어 알고리즘
(Fig. 8) Flow of Control for Each Processor

4. 알고리즘 분석 및 구현결과

본 알고리즘의 분석을 위하여 볼륨이 n×n×n 복셀의 데이타세트, 최종 렌더링 영상의 크기가 n×n화소, p 개의 프로세서들이 직접 연결된 병렬 컴퓨터에서 한 프로세서에서 다른 프로세서로 크기가 b 바이트인 메시지를 전송하는데 T_s+T_wb의 시간이 걸린다고 가정하였다. 여기서 T_s는 start-up 시간상수이고 T_w는 네트워크링크 대역폭의 역이다.

음영테이블은 대략 8K 개의 엔트리로 구성된 lookup 테이블이다. 테이블 엔트리의 계산을 병렬화하기 위하여 프로세서사이에 엔트리를 균등하게 분할하여 각 프로세서에서 할당받은 엔트리를 계산한 후, 각 프로세서는 전체의 테이블 엔트리를 필요로 하므로 프로세서간의 통신에 의해 전체 테이블 엔트리를 구성하였다. 엔트리를 4 바이트 크기의 실수로 표현하면 한 프로세서에서 다른 프로세서로 전송하는 메시지의 크기 b는 8Kx4/p이며 MasPar MP-2 네트워크에서 전송하는데 걸리는 통신 오버헤드는 $T_s\log p+T_wbp$[13]이다. 따라서 음영 테이블의 엔트리를 계산하는데 걸리는 시간과 통신 오버헤드는 프로세서의 수가 일정할 때 테이블의 크기에 따라 선형으로 증가하지만 하나의 엔트리를 계산하는 시간은 그것을 통신하는데 걸리는 시간에 비하여 매우 크므로 테이블의 크기가 증가함에 따라 속도향상이 커짐을 알 수 있다.

렌더링할 때 볼륨의 방향이 변화함에 따라 밀림변환도 변화하며, 이 때 그림 7(c)와 같이 프로세서사이에 스캔라인의 통신이 필요하다. 그림 7(c)에서 연속적인 두 프레임사이의 밀림변환 θ(라디안)=b-a는 매우 작은 크기이며, β라디안의 밀림변환인 경우 슬라이스 i는 $i\tan\beta\approx i\beta$만큼 하향 변위하게 되며, θ(라디안)=b-a의 밀림변환은 i번째 슬라이스에 대하여 $i\theta$만큼 하향 변위한다. 따라 $\frac{n(n+1)}{2}$서 모든 슬라이스에 대하여 하향 변위량의 총합은 θ스캔라인이다. 각 프로세서에 n/p 개의 스캔라인을 균등하게 할당한다면 최대 하향변위

$(n\theta)$만큼 프로세서 p_i와 p_{i+w}사이에 통신이 이루어진다. 따라서 불투명 복셀을 4 바이트로 표현하므로 본 논문의 밀림변환 볼륨 공간 분할에 의한 통신시간은 다음 식과 같이 된다.

$$O(wT_s + \frac{n(n+1)}{2}\theta nx4xT_w) = O(wT_s + 2n^3\theta T_w)$$

두 프레임사이의 밀림변환의 각도가 1도이면 $\theta = 0.017$이고 $w = p\theta$이므로 밀림 볼륨 공간 분할에 의한 통신 오버헤드는 매우 적음을 알 수 있다.

Warping 단계에서는 최종영상의 화소를 구하기 위하여 중간영상에서 4 개의 최인접 화소의 값을 찾아 보간하였다. n^2 개의 화소가 있고 각 화소에 대한 계산시간이 일정하므로 이 단계의 시간 복잡도는 $O(n^2)$이다. 최종영상이 프로세서사이에 균등하게 할당되어 있고 부하균형화가 이루어질 경우 병렬 warping 시간은 $O(n^2/p)$가 된다.

Warping 단계를 거친 후, 각 프로세서는 최종영상의 일부분을 목적지 프로세서에 보내며, 목적지 프로세서는 이들 영상을 조립하여 최종영상을 디스플레이한다. 한 프로세서에서 화소당 1 바이트의 $O(n^2/p)$ 화소 데이타를 최종 목적지 프로세서에 보내야 하므로 통신하는데 걸리는 시간은 $O(p(T_s + T_w n^2/p))$이 되는데 렌더링할 때 배경색과 같은 빈 공간이 많은 부분을 차지하므로 본 논문에서는 이러한 화소들을 최종 목적지 프로세서에 보내지 않음으로써 통신시간을 줄일 수 있었다.

본 논문의 병렬볼륨렌더링 알고리즘을 범용 병렬 컴퓨터인 MasPar MP-2상에서 C언어와 MPL(MasPar Programming Language) 언어[13]를 이용하여 구현하였다. MasPar MP-2 시스템의 특성은 다음과 같다.

- 모 델 : MasPar MP-2(MP-2204)
- 프로세서 수 : 4,096(64x64 mesh)
- Clock Speed : 12MHz
- 메모리 : 총 256MB(PE : 64KB)
- 성 능 : 1,600 MFLOPS(SP), 600MFLOPS (DP) :
- 메모리 대역폭(bandwidth)
  . Direct : 5,000MBytes/sec
  . Indirect : 1,950MBytes/sec
- ACU CMEM size : 512 KBytes
- IOCTLR : 8 MBytes
- IORAM : 1024 MBytes

인체 두뇌에 대한 128×128×128 복셀의 MRI 스캔데이타와 인체 머리에 대한 256×256×226 복셀의 CT 스캔데이타를 대상으로 실험하였으며, 그림 9과 10은 각각의 테스트 데이타 세트를 나타낸 것이다. 렌더링시 볼륨이 회전함에 따라 통신 오버헤드가 변화하므로 볼륨이 1 도 회전함에 따라 1 프레임씩 렌더링하여 90도 회전할 때까지 90 프레임의 렌더링 수행시간을 평균하여 알고리즘 수행시간을 구하였다. 표 2와 3은 각 스캔데이타에 대하여 4,096개의 프로세서를 가진 MasPar MP-2 시스템에서 다음과 같이 세 가지 방법으로 나누어 병렬 렌더링의 수행시간을 측정한 것을 나타낸 것이다.

실험 1) 볼륨 데이타의 빈 복셀을 제거하지 않은 3차원 배열 구조를 이용하며, 부하를 균형화하지 않고 Shear-Warp 알고리즘을 병렬화한다.

실험 2) 볼륨 데이타의 빈 복셀을 제거하기 위하여 그림 4와 같은 볼륨 데이타 구조를 이용하며, 부하를 균형화하지 않고 Shear-Warp 알고리즘을 병렬화한다.

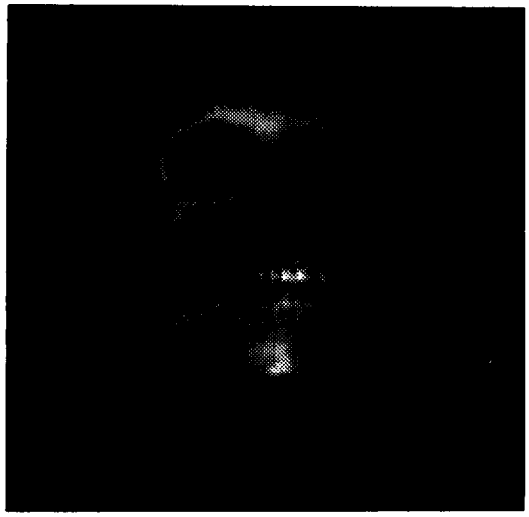
실험 3) 볼륨 데이타의 빈 복셀을 제거하기 위하여 그림 4와 같은 볼륨 데이타 구조를 이용하며, 부하를 균형화하기 위하여 본 논문의 데이타 분할 기법과 부하균형화 기법을 이용하여 Shear-Warp 알고리즘을 병렬화한다.

표 2는 인체 두뇌에 대한 128×128×128 복셀의 MRI 스캔데이타를 대상으로 병렬 렌더링을 수행한 결과를 나타낸 것으로 실험 2)의 경우 빈 복셀을 제거함으로써 처리할 복셀의 수가 크게 감소하였으므로 실험 1)에 비하여 수행속도가 2.2배 향상되었으며, 빈 복셀을 제거하고 본 논문의 데이타 분할 기법과 부하균형화 기법을 이용한 실험 3)의 경우 실험 1)에 비하여 수행속도가 5.18배 향상되었다. 따라서 4,096개의 프로세서를 가진 MasPar MP-2에서 128×128×128 복셀의 MRI 스캔데이타를 1초에 3~4 프레임의 속도로 렌더링하므로 본 알고리즘의 확장성(scalability)에 의해 16,384개의 프로세서를 가진 MasPar MP-2에서는 1초에 12~16 프레임의 렌더링이 기대된다.

표 3은 인체 머리에 대한 256×256×226 복셀의 CT

스캔데이타를 표 2와 같은 방법으로 수행시간을 구한 것으로 실험 1)과 실험 2)의 경우는 MasPar MP-2의 SIMD 프로세서당 메모리의 한계 때문에 수행이 불가하였으며, 본 논문의 병렬 알고리즘을 적용한 실험 3)의 경우 1프레임당 1.2101초가 소요되어 MasPar MP-2에서 실시간 렌더링에 대한 한계를 발견할 수 있었다.

(그림 9) 인체 두뇌의 MRI 데이타 세트
(128×128×128 복셀)
(Fig. 9) MRI Brain Data Set



(그림 10) 인체 머리의 CT 데이타 세트
(256×256×256 복셀)
(Fig. 10) CT Head Data Set

〈표 2〉 128×128×128 복셀의 병렬 볼륨 렌더링 수행 시간

〈Table 2〉 Run Time for 128×128×128 Voxels

구 분	복셀 수	빈 복셀 제거 여부	부하 균형화 여부	수행시간 (sec)
실험 1)	2,097,152	×	×	1.6102
실험 2)	274,487	○	×	0.7331
실험 3)	274,487	○	○	0.3109

〈표 3〉 256×256×256 복셀의 병렬 볼륨 렌더링 수행 시간

〈Table 3〉 Run Time for 256×256×256 Voxels

구 분	복셀 수	빈 복셀 제거 여부	부하 균형화 여부	수행시간 (sec)
실험 1)	16M	×	×	수행불가
실험 2)	1,339,400	○	×	수행불가
실험 3)	1,339,400	○	○	1.2101

Amin 등[9]은 CM-5에서 Shear-Warp 알고리즘을 병렬화하였는데, 이 방법은 볼륨 데이타세트로부터 빈 복셀을 제거하지 않고 처리하므로 많은 메모리 공간이 요구되며, 통신 오버헤드가 문제가 된다. 표 4는 Amin 등[9]에 의한 방법과 본 논문의 방법에 대하여 실험한 결과를 나타낸 것이다. CM-5 시스템은 40MHz로 동작하는 1,024 개의 Sparc 마이크로프로세서로 구성되고, 이론상 128 GFLOPS의 연산능력을 갖는 시스템으로 본 논문에서 사용한 MasPar MP-2보다 성능이 우수함을 알 수 있다.

표 4에서 보는 바와 같이 Amin 등[9]에 의한 Shear-Warp 알고리즘의 병렬화는 CM-5에서 128x128x84 크기의 인체두뇌에 대한 복셀을 렌더링하는데 0.593초가 소요된 반면, 본 논문에서는 128x128x128 크기의 인체 두뇌 복셀에 대하여 0.3109초가 소요되어 처리속도가 향상되었음을 알 수 있다.

또한 Ma[12]에 의한 Binary-Warp 영상합성법을 이용한 병렬 볼륨렌더링은 분할과 정복(divide and conquer) 기법에 의하여 광선추적 알고리즘을 수행하고, 병렬로 영상을 합성하는 병렬 볼륨렌더링을 CM-5에서

구현하였는데 128x128x128 크기의 인체두뇌 데이타 세트에 대하여 약 2초가 소요되었다. CM-5 시스템에서 구현한 Amin 등[9]에 의한 방법과 Ma[12]에 의한 방법의 수행시간을 비교할 때, 순차 Shear-Warp 알고리즘을 병렬화한 볼륨렌더링이 Binary-Warp 영상합성법에 의한 병렬 볼륨렌더링보다 수행속도가 빠름을 알 수 있다.

〈표 4〉 순차알고리즘의 병렬화 비교
〈Table 4〉 Comparison on Parallelizations of Sequential Algorithm

구 분	시스템	복셀 수	수행시간 (sec)
Amin등의 병렬화[9]	CM-5	128x128x84	0.593
본 논문의 병렬화	MasPar MP-2	128x128x128	0.3109

5. 결 론

본 논문에서는 빠른 볼륨 렌더링을 위하여 순차 Shear-Warp 알고리즘을 4,096개의 프로세서를 갖는 범용 병렬 컴퓨터인 MasPar MP-2 시스템에서 병렬화하였다.

광선추적시 빈 복셀들과 포화도에 도달한 화소들을 계산하지 않고 건너 뛰게 하여 처리할 복셀들의 수를 크게 줄이기 위하여 렌더링하기 이전에 전처리 단계에서 볼륨을 연속길이 부호화 기법을 이용하여 부호화한 볼륨 데이타 구조를 이용함으로써 128×128×128복셀의 인체 두뇌 MRI 스캔 데이타에 대하여, 압축하지 않은 3차원 배열 구조를 이용하여 Shear-Warp 알고리즘을 병렬화한 경우에 비하여 렌더링 속도를 2.2배 향상시켰다.

병렬 렌더링시 변환연산을 매우 간단하고 효율적으로 수행하기 위하여 볼륨을 먼저 밀림변환시키고 볼륨 슬라이스들에 수직인 방향으로 분할하는 밀림변환 공간분할 기법을 이용하여 볼륨을 분할하고, 연속적인 두 프레임사이에 볼륨이 크게 변화하지 않으므로 다음 렌더링 단계를 처리하는 순환동안 이전의 렌더링 순환에서 계산된 부하정보를 사용하여 부하를 균형화하는 부하균형화 기법을 이용하였다. 그 결과 128×128×128 복셀의 인체 두뇌 스캔 데이타세트에 대하여, 3차원 배열 구조를 이용하고 부하를 균형화하지 않고 Shear-Warp 알고리즘을 병렬화한 경우에 비하여 렌더링 속도를 5.18배 향상시켰다.

4,096개의 프로세서를 가진 MasPar MP-2에서 128×128×128 복셀의 MRI 스캔데이타를 1초에 3~4 프레임의 속도로 렌더링하므로 본 알고리즘의 확장성(scalability)에 의해 16,384개의 프로세서를 가진 MasPar MP-2에서는 1초에 12~16 프레임의 렌더링이 기대된다. 또한 더 큰 볼륨에 대해서도 최근의 SIMD 또는 MIMD 머신상에서는 초당 30~60 프레임의 렌더링이 가능할 것으로 기대된다.

인체 머리에 대한 256×256×226 복셀의 CT 스캔 데이타의 경우, 빈 복셀을 제거하지 않은 3차원 배열 구조를 이용하고 본 논문의 데이타 분할 기법과 부하균형화 기법을 이용하지 않으면 MasPar MP-2의 SIMD 프로세서당 메모리의 한계때문에 수행이 불가하였으며, 본 논문의 병렬 알고리즘을 적용한 경우 1프레임당 1.2 101초가 소요되어 MasPar MP-2에서 실시간 렌더링에 대한 한계를 발견할 수 있었다.

향후 연구과제로는 본 병렬 알고리즘의 확장성을 확인하기 위하여 16,384개의 프로세서를 갖는 MasPar MP-2와 SGI Challenge 등과 같은 다른 머신과 최근의 SIMD 또는 MIMD 머신에서 구현하는 문제이다.

참 고 문 헌

[1] M. Levoy, "Display of Surfaces from Volume Data," IEEE Computer Graphics and Applications, Vol.8, No.3, pp.29-37, May 1988.

[2] 이효종, 임훈철, "트랜스퓨터 시스템에서의 병렬 광선추적 알고리즘", 정보과학회논문지, 제 23권 9호, pp.924-933, 1996.

[3] Jurgen Knittel, Hanspeter Pfister, and Arie Kaufman, "Three Architectures for Volume Rendering," Eurographics '95, Vol.14, No.3, pp.111-122, 1995.

[4] Terry S. Yoo, Ulrich Neumann, Henry Fuchs Stephen M. Pizer, Tim Cullip, John Rhoades, and Ross Whitaker, "Direct Visualization of Volume Data," IEEE Computer Graphics and Applications, Vol.12, No.4, pp.63-71, July

1992.

[5] Philippe Lacroute and M. Levoy, "Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation," In Proceedings of the SIGGRAPH 94 Conference, pp.451-457, July 1994.

[6] Brian Cabral, Nancy Cam, and Jim Foran, "Accelated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware," In Proceedings 1994 Symposium on Volume Visualization, pp.91-98, Oct. 1994.

[7] Donald Meagher, "Fourth-Generation Computer Graphics hardware Using Octrees," National Computer graphics Assoc., Chicago, pp. 316-325, Apr. 1991.

[8] H. Pfister and Arie Kaufman, "Real-Time Architecture for High Resolution Volume Visualization," Proceedings of 8th Eurographics Workshop on Graphics Hardware Proceedings, pp.72-80, Sep. 1993.

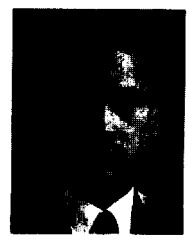
[9] Minesh B. Amin, Ananth Grama, Vineet Singh, "Fast Volume Rendering Using an Efficient, Scalable Parallel Formulation," Parallel Rendering Symposium, Atlanta GA, pp.7-14, 1995.

[10] Philippe Lacroute, "Real-Time Volume Rendering on Shared Memory Multiprocessors," Parallel Rendering Symposium, Atlanta GA, pp.15-22, 1995.

[11] Craig M. Wittenbrink and Kwansik Kim, "Data Dependent Optimizations for Permutation Volume Rendering," Symposium on Electric Imaging : Science and Technology, 4-19, Jan. 1997.

[12] Kwan Liu Ma, James S. Painter, Chars D. Hansen, and Michael F. Krogh, "A Data Distributed, Parallel Algorithm for Ray-Tacced Volume Rendering", Proceedings of the 1994 Parallel Rendering Symposium," pp.15-22, 1993.

[13] 'MPEE(MasPar Programming Environment) User Guide Software Version 3.0', MasPar Computer Corporation, July 1992.



김 응 곤

1980년 조선대학교 전자공학과 졸업(학사)

1987년 한양대학교 대학원 전자공학과 졸업(석사)

1992년 조선대학교 대학원 전기공학과 졸업(박사)

1984년~1986년 금성반도체(주) 연구소 연구원

1987년~1991년 국방과학연구소 선임연구원

1991년~1993년 여수수산대학교 컴퓨터공학과 전임강사

1997년~1998년 미국 University of California, Post-Doc.

1993년~현재 순천대학교 컴퓨터과학과 부교수

관심분야 : 컴퓨터 그래픽스, CAD, 정보 시각화