원형기반 객체순서의 원근 투영 볼륨 렌더링
(Template-Based Object-Order Volume Rendering with Perspective Projection)

구 윤 모 † 이 철 희 †† 신 영 길 †††
(Yun-Mo Koo) (Cheol-Hi Lee) (Yeong Gil Shin)

요 약 원근 투영에 의해 만들어 진 영상은 물체에 대한 거리감을 얻게 해 줌으로써 복잡한 구조물을 해석하는 데 도움을 준다. 현재 원근 투영 볼륨 렌더링의 가장 큰 단점은 실행시간이 길다는 점에 있다. 본 논문에서는 원근 광선들 간의 규칙성을 이용하여 원근 투영 볼륨 렌더링을 효율적으로 처리하는 방법을 소개한다. 볼륨의 한 면과 평행인 중간 화상의 각 수평 스캔라인과 수직 스캔라인으로부터 출발한 광선들에 대해 원형(template)들을 만든다. 광선 위의 샘플들은 원형에 미리 계산되어 있는 가중치를 사용하여 샘플 주변의 복셀들을 가중치 평균하여 얻는다. 이러한 원형의 사용은 볼륨 렌더링에서 많은 부분을 차지하고 있는 재샘플링 시간을 줄일 수 있게 해 준다. 뿐만 아니라, 본 알고리즘은 화상순서가 아니라 객체순서로 렌더링을 하므로 각 복셀을 한번 씩 만 접근하여 처리하고 런-길이 부호화된 볼륨을 사용하여 데이타 응집성의 특성을 이용함으로써 알고리즘을 더욱 가속화한다. 또한, 볼륨을 시점으로부터의 거리에 따라 여러 구역으로 나누어 각 구역에서 광선의 간격을 새로 조정함으로써 원근 광선의 발산으로 인해 샘플링 밀도가 감소하는 문제를 해결한다. 성능 평가 및 다른 방법과의 비교를 통해 원형의 사용과 객체순서 처리로 인한 속도 향상을 분석한다. 또한, 샘플링 밀도의 유지로 인해 화질이 향상된 결과를 보인다.

Abstract Perspective views provide a powerful depth cue and thus aid the interpretation of complicated images. The main drawback of current perspective volume rendering is the long execution time. In this paper, we present an efficient perspective volume rendering algorithm based on coherency between rays. Two sets of templates are built for the rays cast from horizontal and vertical scanlines in the intermediate image which is parallel to one of volume faces. Each sample along a ray is calculated by interpolating neighboring voxels with the pre-computed weights in the templates. We also solve the problem of uneven sampling rate due to perspective ray divergence by building more templates for the regions far away from a viewpoint. Since our algorithm operates in object-order, it can avoid redundant access to each voxel and exploit spatial data coherency by using run-length encoded volume. Experimental results show that the use of templates and the object-order processing with run-length encoded volume provide speedups, compared to the other approaches. Additionally, the image quality of our algorithm improves by solving uneven sampling rate due to perspective ray divergence.

1. 서 론

최근 볼륨 렌더링은 과학, 공학 또는 의학 분야에서 방대한 3차원 데이타를 분석하기 위한 중요한 기술로 대두되고 있다. 그러나 현재 상용되는 대부분의 볼륨 렌더링에서는 평행 투영을 사용하고 있다. 평행 투영은 관찰자가 관찰의 대상이 되는 물체로부터 멀리 떨어져 있는 경우에는 어느 정도 사실적이나, 물체를 매우 가까이서 관찰할 경우에는 물체의 각 부분에 대한 거리감이 상실되어 물체에 대한 올바른 해석이 어렵게 된다. 반면, 원근 투영에 의한 영상은 관찰자에게 이러한 물체에

· 본 논문은 BK21 정보사업단의 지원을 받은 것임
† 비 회 원 : 서울대학교 컴퓨터공학부
yunmo@cglab.snu.ac.kr
†† 비 회 원 : (주)엑스폼닷컴 대표이사
chlee@cglab.snu.ac.kr
††† 종신회원 : 서울대학교 컴퓨터공학부 교수
yshin@cglab.snu.ac.kr
논문접수 : 1999년 8월 11일
심사완료 : 2000년 4월 20일

대한 거리감을 제공하여 줌으로써 복잡한 구조체에 대한 입체적인 인지를 가능케 해 준다.

그러나 현재까지 연구된 원근 투영 볼륨 렌더링의 가장 큰 단점은 실행 시간이 길다는 점에 있다. 볼륨 렌더링은 일반적으로 광선을 따라가면서 각 샘플에서 재샘플링(resampling)을 한 후에 그 결과 값을 해당하는 화소에 합성시킨다. 여기서 재샘플링이란 샘플의 주변 복셀들로부터 그들의 색상과 불투명도 값을 가져와 이들을 보간하여(interpolate) 샘플 점에서의 색상과 불투명도 값을 결정하는 연산을 의미한다. 볼륨 렌더링에서는 볼륨 데이타의 크기가 방대하므로 처리해야 할 샘플의 수는 매우 크다. 뿐만 아니라, 각 샘플에서의 재샘플링 연산이 많은 시간을 요구하므로 렌더링 시간이 길어지게 된다. 한편, 평행 투영 볼륨 렌더링에서는 가속화 방법이 많이 연구되어 왔다. 그 중 대표적인 쉬어-왑(shear-warp) 렌더링과 원형기반의 렌더링에서는 평행 광선의 규칙성을 이용하여 재샘플링 연산을 단축시켰다[5,12-13,15-16]. 이들은 볼륨의 한 면과 평행한 평면 위에서 중간화상을 만든 후에 와핑을 통하여 최종적인 화상을 만든다. 이때 중간화상에서 발사된 평행 광선들은 모두 같은 형태를 갖는 특징이 있다. 따라서, 광선의 진행경로와 각 샘플에서의 재샘플링 가중치를 계산하여 원형에 저장한 후에, 광선을 만들 때마다 이 원형을 사용할 수 있다. 모든 광선은 미리 만들어 놓은 원형에 따라 생성되므로 재샘플링 연산에 드는 시간이 크게 감소하게 된다. 그러나, 원근 투영에서는 광선들이 각기 다른 각도로 발산되어 서로 다른 형태를 가지기 때문에 평행 투영에서 사용된 방법을 이용하기 어렵다. 현재의 원근 투영 볼륨 렌더링 기술에서는 이와 같은 샘플 계산에서 평행 투영에 비해 많은 시간을 소모하므로 실행 시간이 길다.

원근 투영 볼륨 렌더링의 기존 연구들은 주로 효율적인 데이타 탐색을 통해 알고리즘을 가속시켰다 [1-5]. Novins외 2인의 알고리즘은 광선 발사법(ray casting)을 슬라이스 순서로 처리한다[4]. 각 슬라이스는 한번씩 만 메모리로 적재되므로 데이타 처리가 매우 효율적이다. 그러나 이 알고리즘 역시 각 샘플 값을 얻을 때마다 재샘플링 가중치 계산을 새로 해야 하므로 전체적인 렌더링 시간은 느리다. Lacroute와 Levoy의 원근 투영 쉬어-왑 렌더링에서도 슬라이스 순서로 렌더링을 수행한다[5]. 각 슬라이스는 시점에 대한 거리에 따라 축소되어 쉬어(shear)된 후에 중간화상으로 투영된다. 이 알고리즘은 쉬어-왑 분해의 장점을 그대로 이용하므로 효율적이다. 그러나 재샘플링 가중치 계산은 각 샘플마다 매번 수행되고 또 각 슬라이스를 중간화상에 투영시킬 때 다 대 일의 비율로 사상되므로 알고리즘의 효율성을 떨어뜨린다. 또한, 합성하기 전에 필터링을 수행하기 때문에 잘못된 가시성(visibility)이 생기기도 한다[5]. Brady외 3인은 최근 원근 투영 렌더링으로 연속적인 프레임을 효율적으로 만들 수 있는 방법을 제안했다 [1-2]. 이 알고리즘은 볼륨을 시점으로부터의 거리에 따라 여러 구역으로 나눈 후에 시점에서 가까운 구역만을 새로 렌더링하고, 멀리 있는 구역에 대해서는 바로 전에 렌더링한 결과를 사용한다. 이는 약간의 부정확성을 포함하게 되지만 연속적으로 프레임을 만들 때에 유용하다. 그러나 기본적으로 볼륨을 새로 렌더링할 때 일반적인 광선 발사법을 사용하므로 한 프레임을 생성하는 자체는 시간이 많이 걸린다.

본 논문에서는 볼륨 렌더링에서 원근 투영을 효율적으로 지원할 수 있는 방법을 제안한다. 본 알고리즘의 효율성은 원형의 사용과 객체순서식 처리에 기인한다. 볼륨의 한 면과 평행인 중간화상의 수직 스캔라인과 수평 스캔라인 별로 원형을 만든다. 동일한 스캔라인에서 나오는 광선들은 2차원상에서 동일한 형태를 가지게 된다. 본 알고리즘에서는 이러한 스캔라인 광선들의 일관성을 이용하여 샘플의 위치와 재샘플링을 위한 가중치를 계산해 둠으로써 샘플을 계산하는데 걸리는 시간을 단축시킨다. 뿐만 아니라, 객체순서로 복셀들을 처리해 감으로써 효율적인 데이타 탐색이 이루어지며, 런-길이 부호화(run-length encoding)와 같은 자료구조를 사용하여 데이타 응집성의 특성을 이용한다.

원근 투영에서 다루어야 할 또 다른 문제는 광선의 발산으로 인한 샘플 밀도의 불균등 현상이다. 광선의 밀도를 시점 부근의 볼륨 데이타의 밀도에 맞추게 되면 시점으로부터 멀어질수록 광선의 간격이 복셀의 간격보다 커지게 되므로 샘플의 밀도는 복셀 데이타의 밀도보다 매우 작아지게 된다. 반면, 광선의 밀도를 볼륨 데이타의 뒤 부근에 맞추게 되면, 후방에 있는 복셀들은 모두 통과할 수 있지만 볼륨의 앞 부근에서는 광선의 밀도가 커지게 되므로 오버샘플링(oversampling) 현상이 생겨 불필요하게 렌더링 시간이 증가하게 된다. Novins의 알고리즘에서는 이러한 현상을 처리하기 위해 광선의 밀도가 정해진 기준보다 떨어지는 지점에서 광선을 쪼개어 발사시킴으로써 적응(adaptive) 샘플링을 가능케 하였다. Brady의 알고리즘에서는 시점으로부터의 거리에 따라 볼륨을 나누어 각 구역에 맞게 광선의 수를 조절하였다. Kreeger의 알고리즘에서는 구역을 나눌 때 시점으로부터의 거리를 2의 제곱 승 배 씩 증가시키면

서 나누었다[3]. 이는 Lee의 논문에서 제안한 방법과 매우 유사한 것으로써, 원근 투영 볼륨 렌더링에 적용한 것이다 [8-9]. 2의 제곱 승 배씩 증가시키면서 구역을 나눌 때 광선이 분할되는 모양이 일정한 형태를 가지게 되어 박스(box) 필터보다 더 정밀한 삼각(triangle) 필터를 사용할 수 있게 되므로 화질이 향상된다. 본 알고리즘에서는 적응 샘플링을 지원하기 위하여 시점으로부터의 거리에 따라 볼륨을 여러 구역으로 나누고 각 구역에 따라 원형의 간격을 재조정한다. 따라서 불규칙한 샘플 밀도의 문제를 효율적으로 해결할 수 있다.

본 논문의 구성은 다음과 같다. 2절에서는 원근 광선들 사이에 존재하는 광선의 일관성을 소개하며 이를 이용하여 원형을 사용하는 기법을 제안한다. 또한 적응 샘플링을 지원하기 위한 방법을 설명한다. 3절에서는 객체순서로 처리하는 기법을 설명하며, 광선 조기 종료법(early ray termination) 및 런-길이 부호화 기법을 사용하여 알고리즘을 최적화하는 기술을 설명한다. 4절에서는 실험결과를 분석한다.

2. 원근 투영 볼륨 렌더링에서의 원형기반 기법

2.1 수직 및 수평방향으로의 일관성

본 알고리즘에서는 객체 좌표계와 중간 객체 좌표계의 두 종류의 좌표계를 사용한다. 객체 좌표계는 볼륨 데이타가 정의된 좌표계이며 임의의 좌표를 (u, v, w)로 표시한다. 원점은 볼륨의 한 끝점이며, 복셀 데이타는 정수 좌표에 규칙적으로 분포되어 있다. 중간 객체 좌표계의 각 좌표는 (x, y, z)로 나타내며, 객체 좌표계의 좌표축을 돌려 시각 방향의 주축이 z축이 되도록 조정한 좌표계이다. 여기서 주축이란 x, y, z축 중에서 시각 방향과 가장 가까운 축을 의미한다. 복셀 슬라이스는 z축과 수직이다. 중간 화상은 x-y 평면과 평행이며, 중간 화상의 해상도는 볼륨의 해상도와 항상 일치하지는 않는다. 즉, 한 복셀 당 여러 개의 광선을 쏘는 경우에는 중간 화상의 해상도가 볼륨 데이타의 해상도보다 커지게 된다. 중간화상의 각 좌표는 (i, j)로 나타낸다. 본 알고리즘에서는 샘플의 위치를 정할 때 기존 광선 발사법과는 다른 방법을 사용한다. 기존 광선 발사법에서는 일반적으로 그림 1(a)과 같이 유클리디안(Euclidean) 거리에 따라 샘플의 위치를 정한다 [1,2,10,11]. 반면, 본 알고리즘에서는 그림 1(b)와 같이 볼륨 슬라이스와 평행하게 샘플을 얻는다 [3,5,7]. 이러한 방식으로 샘플을 선택하는 것은 객체순서에 기반을 둔 알고리즘에서 사용하고 있는 방법이다.

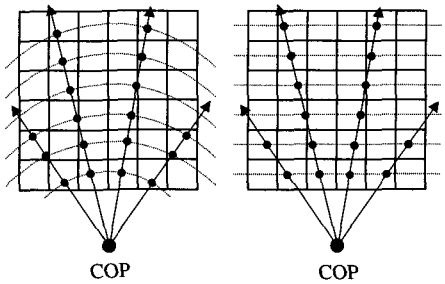

그림 1 샘플을 정하는 방법 (a) 일반적인 광선 발사법 (b) 본 알고리즘

여기서, COP = $(x_{COP}, y_{COP}, z_{COP})$는 투영중심이다.

중간 화상의 수직 스캔라인의 화소는 동일한 i 좌표 값을 가지며, 수평 스캔라인의 화소는 동일한 j 좌표 값을 갖는다. 수직 스캔라인의 화소에서 나오는 모든 광선들은 x-z 평면 상에서 동일한 직선으로 투영되며, 수평 스캔라인의 화소에서 나오는 모든 광선들은 y-z 평면 상에서 동일한 직선으로 투영된다 (그림 2). 여기서 동일 수직 스캔라인의 광선들 간에 존재하는 규칙성을 수직 일관성이라고 하며, 동일 수평 스캔라인의 광선들 간에 존재하는 규칙성을 수평 일관성이라고 한다. 위의 두 가지 일관성을 이용하여 수직, 수평의 두 종류의 원형 집합을 만든다. 수직 원형 집합의 각 원형은 각각의 수직 스캔라인의 광선을 대표하며, 수평 원형 집합의 각 원형은 각각의 수평 스캔라인의 광선을 대표한다. 따라서, 중간 화상의 크기가 $n_i \times n_j$이라면 n_i개의 수직 원형들과 n_j개의 수평 원형들이 필요하다.

이제 각 원형의 구성 요소를 정의하고, 이러한 원형을 사용하여 각 샘플에서 재샘플링 하는 방법에 대해 설명한다. 한 화소를 $\mathbf{I}_{i,j} = (i, j)$ 라고 정의하며, 화소 $\mathbf{I}_{i,j}$의 광선 위의 샘플 $\mathbf{P}_k = (x_{P_k}, y_{P_k}, z_{P_k})$에서 재샘플링을 한다고 하자. 수직 원형의 각 요소는 순서쌍 $\mathbf{V}_k^i = \langle \mathbf{X}_k^i, \mathbf{Wx}_k^i \rangle$로 정의된다. 여기서 $\mathbf{X}_k^i = (\lfloor x_{P_k} \rfloor, \lfloor z_{P_k} \rfloor)$는 x-z 평면상에서의 x 좌표와 z좌표이며, $\mathbf{Wx}_k^i = \{wx_k^{rt} \mid r,t \in \{0,1\}\}$는 x축과 z축 방향으로의 가중치 집합이다. 수평 원형의 각 요소는 $\mathbf{H}_k^j = \langle \mathbf{Y}_k^j, \mathbf{Wy}_k^j \rangle$로서 정의되며 여기서 $\mathbf{Y}_k^j = (\lfloor y_{P_k} \rfloor)$는 y-z평면 상의 y좌표이다. $\mathbf{Wy}_k^j = \{wy_k^s \mid s \in \{0,1\}\}$는 y축 방향으로의 가중치 집합이다.

본 알고리즘에서는 재샘플링을 위해 삼선형(trilinear) 보간법을 사용하므로 각 샘플마다 8개의 주변 복셀들에

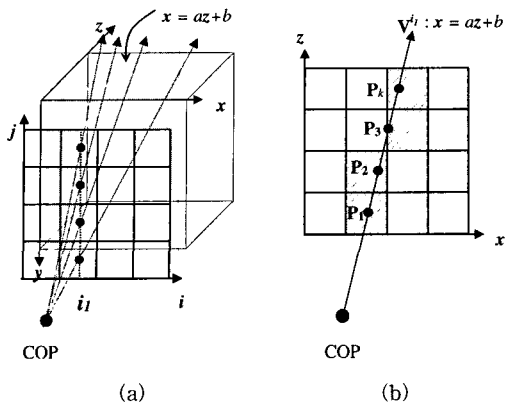

그림 2 (a) 동일한 수직 스캔라인에서 나온 광선들 간에 존재하는 규칙성과 (b) 이에 대응되는 원형의 예

대한 가중치 집합 $\mathbf{W}_k = \{w_k^{rst} \mid r,s,t \in \{0,1\}\}$이 필요하며 각 요소는 다음과 같이 정의된다.

$$w_k^{000} = (1-r_k)(1-s_k)(1-t_k)$$
$$w_k^{001} = (1-r_k)(1-s_k)t_k$$
$$w_k^{010} = (1-r_k)s_k(1-t_k)$$
$$w_k^{011} = (1-r_k)s_kt_k$$
$$w_k^{100} = r_k(1-s_k)(1-t_k)$$
$$w_k^{101} = r_k(1-s_k)t_k$$
$$w_k^{110} = r_ks_k(1-t_k)$$
$$w_k^{111} = r_ks_kt_k$$

여기서, $r_k = x_{P_k} - \lfloor x_{P_k} \rfloor$, $s_k = y_{P_k} - \lfloor y_{P_k} \rfloor$, $t_k = z_{P_k} - \lfloor z_{P_k} \rfloor$이다.

그러나, 수직 스캔라인과 수평 스캔라인으로 나누어 원형을 만들므로 위의 가중치 집합 역시 수평, 수직 방향으로 나누어 진다. x축 방향으로의 가중치를 x-가중치 집합이라고 하고, y축 방향으로의 가중치를 y-가중치 집합이라고 하자. 수직 원형에 있는 x-가중치 집합 $\mathbf{Wx}_k$는 다음과 같이 정의된다.

$$wx_k^{00} = (1-r_k)(1-t_k)$$
$$wx_k^{01} = (1-r_k)t_k$$
$$wx_k^{10} = r_k(1-t_k)$$
$$wx_k^{11} = r_kt_k$$

또한, 수평 원형에 있는 y-가중치 집합 $\mathbf{Wy}_k$은 다음과 같이 정의된다.

$$wy_k^0 = (1-s_k)$$
$$wy_k^1 = s_k$$

따라서, 화소 $\mathbf{I}_{i,j}$ 에서 나온 광선 위의 임의의 샘플 $\mathbf{P}_k$는 x-가중치와 y-가중치를 조합함으로써 얻어질 수 있다. 즉, $\mathbf{Wx}_k^i \times \mathbf{Wy}_k^j$로써 얻어지며, 각 요소는 다음과 같다.

$$w_k^{000} = wx_k^{00} \times wy_k^0$$
$$w_k^{001} = wx_k^{01} \times wy_k^0$$
$$w_k^{010} = wx_k^{00} \times wy_k^1$$
$$w_k^{011} = wx_k^{01} \times wy_k^1$$
$$w_k^{100} = wx_k^{10} \times wy_k^0$$
$$w_k^{101} = wx_k^{11} \times wy_k^0$$
$$w_k^{110} = wx_k^{10} \times wy_k^1$$
$$w_k^{111} = wx_k^{11} \times wy_k^1$$

위의 식을 통해, 각 샘플 점에서 재샘플링을 하기 위해 가중치를 구하는 연산에서 곱셈의 수가 반으로 줄어드는 것을 볼 수 있다. 볼륨 데이타의 크기가 $n \times n \times n$ 일 때 샘플의 수는 n^3가 되는데, 본 알고리즘에서는 가중치를 구하는 연산에서 곱셈의 연산을 $8*2*n^3$에서 $8*n^3$로 줄일 수 있다. 따라서 원형의 사용은 재샘플링의 시간을 단축시켜 렌더링 시간을 줄일 수 있음을 보여준다.

2.2 적응 샘플링

본 알고리즘은 시점으로부터의 거리에 따라 볼륨을 여러 구역으로 나누어 각 구역마다 광선의 간격을 재조정함으로써 샘플링 비율을 일정하게 유지할 수 있게 한다. 이러한 기법은 원근 투영 볼륨 렌더링에서 많이 사용되어 온 방법이며, 본 알고리즘에서 구역을 나누는 방법은 Lee의 방법과 유사하다 [8-9]. 즉, COP로부터 볼륨의 첫번째 슬라이스까지의 거리를 d라고 할 때, 그림 3과 같이 볼륨은 COP로부터 거리가 $d2^l$ ($l \geq 0$)인 지점에서 나누어 진다. 이렇게 볼륨을 분할할 때 각 구역

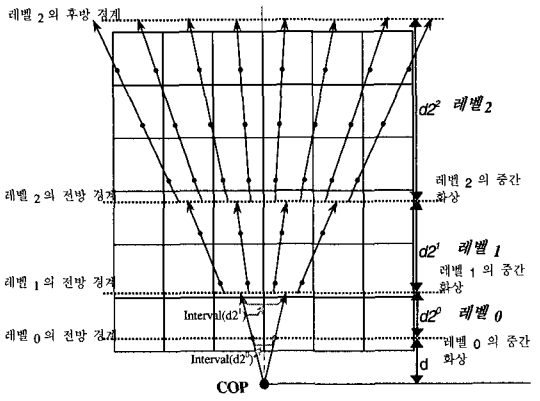

그림 3 COP로부터의 거리에 따른 구역 분할 및 적응 샘플링

의 깊이는 바로 앞 레벨의 깊이보다 두 배 깊어지며, 한 구역 내에서 인접한 광선 간격도 전방 경계에서 보다 후방 경계에서 두 배로 증가하게 된다. 한 구역에서 전방 경계에서의 인접 광선과의 간격을 $interval(d2^l)$라고 하고, 후방 경계에서의 간격을 $interval(d2^{l+1})$라고 할 때, COP로부터의 임의의 거리 d_z에서의 광선과의 간격 $interval(d_z)$는 다음과 같은 특성을 갖는다.

$$interval(d2^l) \le interval(d_z) \le interval(d2^{l+1}) = 2\ interval(d2^l)$$

단, $d2^l \le d_z \le d2^{l+1}$

위의 식은 $interval(d2^l)$를 0.5로 맞출 때, $interval(d2^{l+1})$는 1.0이 되며, $interval(d2^l)$는 0.5와 1.0 사이의 값을 갖게 됨을 의미한다. 즉, 한 구역 내에서 전방경계에서의 광선간의 간격을 0.5로 할 때 그 구역 내에서는 광선이 지나가지 않는 복셀이 존재하지 않음을 알 수 있다. 따라서, 본 알고리즘에서는 구역을 COP로부터 처음 슬라이스까지의 거리인 d를 2의 제곱 승 배로 증가시키면서 나누고, 한 구역 내에서 광선이 모든 복셀을 지나가도록 하기 위해 전방경계에서 복셀 간격의 반 만큼의 간격으로 원형을 만든다. 또한, 다음 레벨의 구역으로 넘어갈 때에 각 원형을 수직, 수평방향으로 2개씩, 총 4개로 분할하여 다음 구역에서의 광선의 간격을 재조정한다. 다음 구역의 원형을 만들 때에는 볼륨의 바깥 쪽으로 나간 부분의 원형은 만들지 않으므로 각 구역마다 만들어지는 원형의 수는 거의 같게 된다.

또한, 다음 구역으로 넘어갈 때 중간화상의 정보도 전달되어야 한다. 즉, 중간화상의 크기를 4배 증가시키고, 각 화소를 4개의 화소에 복사한다. 이때, 화소의 색상 뿐 아니라, 화소 의 불투명도까지 전이시킨다. 이것은 다음 레벨의 구역을 렌더링할 때 아직 렌더링이 완료되지 않은 화소만을 처리할 수 있게 해 주므로 뒤에 있는 구역들을 처리하는 시간을 단축시킬 수 있다. 다시 말하면, 볼륨 렌더링에서 가속화 기법의 하나인 광선 조기 종료를 전역적으로 할 수 있다. 반면, Brady의 알고리즘에서는 각 구역을 새로 처리하므로 광선 조기 종료를 한 구역 내에서만 이용하게 되어 광선 조기 종료의 장점을 거의 이용할 수 없다.

3. 객체순서식 처리

본 논문의 원근 투영 원형기반 렌더링 알고리즘은 객체순서로 수행된다. 이것은 볼륨을 데이타의 저장 순서대로 처리해 나갈 수 있고, 런-길이 부호화와 같은 자료구조를 사용함으로써 데이타의 응집성 등의 특성을 이용할 수 있기에 효율적이다. 본 절에서는 먼저 객체순서로 실행되는 기본 알고리즘을 설명하고, 다음에 광선 조기 종료와 런-길이 부호화된 데이타를 사용하여 알고리즘을 최적화하는 방법을 설명한다.

3.1 기본 방법

본 알고리즘은 슬라이스를 순서대로 가져와 처리한다. 삼선형 보간법을 사용하여 재샘플링을 하므로, 동시에 두개의 슬라이스를 가져와야 한다. 각 슬라이스에서 두 줄 씩의 스캔라인을 가져와 인접한 8개의 복셀들로 이루어진 공간 안에 있는 모든 샘플들을 동시에 계산한다. 즉, 8개의 이웃 복셀에 대한 색상 및 불투명도를 한 번만 계산한 후에 원형에 있는 샘플링 가중치를 사용하여 각 샘플 점에서 재샘플링을 수행한다. 여기서, 인접한 8개의 복셀로 이루어진 구역을 셀(cell)이라고 하자. 한 셀 안에 있는 샘플들을 계산하기 위해서는 그 셀을 통과하는 원형들을 찾아내야 한다. 각 셀을 통과하는 원형을 효율적으로 찾기 위해 우리는 다음과 같은 특성을 이용한다.

특성 1: 두 쌍의 슬라이스 상의 모든 수평 셀 스캔라인들은 동일한 묶음의 수직 원형들이 지나간다. (그림 4(a))

특성 2: 한 수평 셀 스캔라인의 모든 셀은 동일한 수평 원형들이 지나간다. (그림 4(b))

위의 특성 중 첫번째는 수직 스캔라인 상의 모든 셀

에서 동일한 x값을 갖는 샘플들에서는 동일한 수직 원형을 적용할 수 있음을 의미한다. 두 번째 특성은 한 수평 스캔라인에서 동일한 y값을 갖는 샘플에서는 동일한 수평 원형을 적용할 수 있음을 의미한다. 따라서, 한 수평 스캔라인을 지나가는 한 묶음의 수직 원형 집합을 찾은 후에는 현재 처리하고 있는 슬라이스에 있는 모든 셀 스캔라인의 샘플들을 계산할 때 한번 찾아 둔 수직 원형들을 동일하게 사용할 수 있다. 또한, 현재의 수평 스캔라인을 지나가는 수평 원형들을 찾은 후에는 현 스캔라인의 각 셀 안에 있는 샘플들을 계산할 때 이 수평 원형들을 사용할 수 있다.

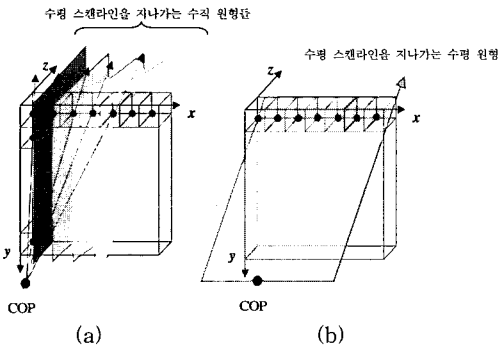

그림 4 특성 1과 2에 대한 예: (a) 모든 셀 스캔라인은 동일한 묶음의 수직 원형이 지나간다. (b) 한 스캔라인의 모든 셀은 동일한 수평 원형이 지나간다.

위의 특성들을 이용하기 위해, 2차원 배열의 자료구조를 두 개 사용한다. 수직 원형에 대한 테이블을 **Mv** 라고 할 때, 각 원소는 x-z 평면상에서 바라 본 셀에 해당한다. 또한, 수평 원형에 대한 테이블은 **Mh**이며, 각 원소는 y-z 평면상에서 바라 본 셀과 같다. 배열의 원소는 각 셀을 지나가는 원형에 대한 인덱스를 가지고 있다. **Mv**의 크기는 $n_z \times n_x$이며, 각 원소는 다음과 같은 순서쌍으로 정의된다.

Mv(z,x) = <*index*, *offset*>

단, z와 x는 (x, y, z) 위치에 있는 셀의 z, x좌표 *index*는 셀을 통과하는 원형들 중에서 처음 원형의 인덱스 *offset*은 현 스캔라인에서 가장 앞에 있는 셀과 현재 셀이 투영되는 화소사이의 상대거리 또한, **Mh**의 크기는 $n_z \times n_y$이며, 각 원소는 다음과 같이 정의된다.

Mh(z,y) = <*index*>

Mh의 원소는 **Mv**와 비슷하게 정의되나 *offset*은 포함하지 않는다. *offset*은 광선 조기 종료와 런-길이 부호화 데이타를 사용할 때, 한 스캔라인에서 건너뛸 복셀 런에 대응되는 화소 런의 길이를 알기 위해 필요한 정보이기에 **Mh**에서는 필요 없다. 여기서, 한 셀을 통과하는 원형은 **Mv**(z,x)과 **Mv**(z,x+1), **Mh**(z,y)과 **Mh**(z,y+1) 사이의 인덱스로써 찾을 수 있다.

이와 같이, 셀을 통과하는 원형을 찾은 후에 셀 안에 있는 샘플들을 계산하여 각각을 해당하는 화소로 합성시킨다 [14]. 예를 들어, 셀을 통과하는 원형이 그림 5와 같이 $V_k^i, V_k^{i+1}, H_k^j, H_k^{j+1}$일 때 각 샘플들은 $\{wx_k^i, wx_k^{i+1}\} \times \{wy_k^j, wy_k^{j+1}\}$ 로써 계산된 후, 화소 $I_{i,j}$, $I_{i+1,j}$, $I_{i,j+1}$, $I_{i+1,j+1}$로 합성된다.

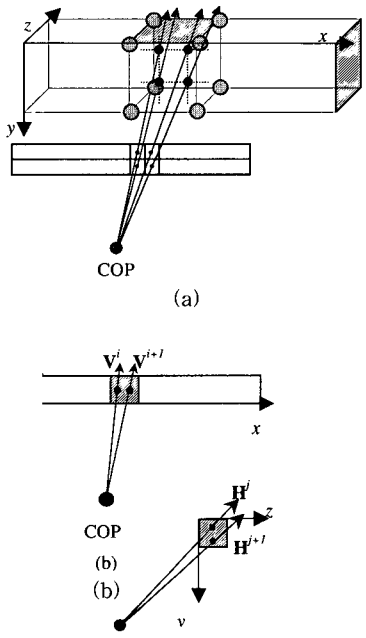

그림 5 (a) 한 셀을 통과하는 광선과 그 위의 샘플들과 대응되는 (b) 수직 원형 (c) 수평 원형

3.2 최적화

원근 투영 렌더링을 가속화하기 위해 런-길이 부호화 볼륨을 사용한다. 이는 빈 공간은 버리고 정보가 들어있는 복셀 들만을 묶은 것으로서, 평행 투영 볼륨 렌더링 기법에서 가속화를 위해 많이 사용되어 온 기법이

다[5-7,12-13]. 본 알고리즘은 삼선형 보간 필터를 사용하여 재샘플링을 하므로 네 줄의 복셀 스캔라인을 동시에 따라가야 하며, 이에 대응되는 화소 스캔라인을 가리키는 포인터를 함께 증가시키면서 정보가 없는 투명한 복셀 런을 건너뛴다. 원근 투영에서는 화소 스캔라인과 복셀 스캔라인의 대응 비율이 다르므로 건너뛰어야 할 화소 런의 길이가 투명한 런의 길이와 일치하지 않는다. 우리는 이것을 **Mv**에 있는 *offset*정보를 이용하여 효율적으로 계산할 수 있다. 예를 들어, 현재 가리키고 있는 복셀 포인터가 (x_1,y,z) 에서 (x_2,y,z)로 건너뛰어야 할 경우에 화소 포인터는 그림 6과 같이 (i,j)에서 $(i + \mathbf{Mv}(z,x_2).\text{offset} - \mathbf{Mv}(z,x_1).\text{offset},\ j)$으로 건너뛰게된다.

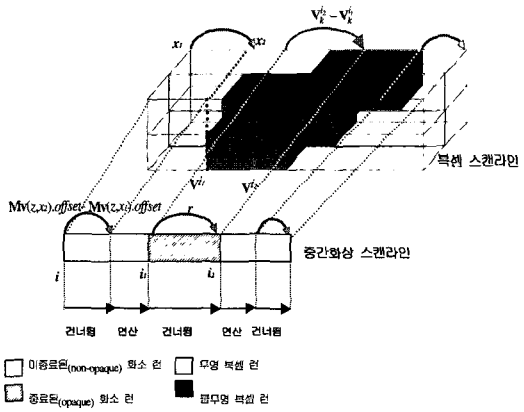

그림 6 광선 조기 종료 기법과 런-길이 부호화 볼륨의 예

또한, 본 알고리즘은 광선 조기 종료 기법을 사용한다. 이는 광선을 따라가면서 샘플들을 합성하다가 일정한 임계 값에 도달했을 때 다음에 있는 샘플들은 합성한 결과에 영향을 주지 않기에 더 이상의 합성을 하지 않고 종료하는 기법이다 [11]. 이 기법을 객체순서 볼륨 렌더링에서 적용하기 위해서는 화소 스캔라인 상에서 종료가 끝난 화소들을 런으로 묶어 이미 종료된 런에 대해 건너뛰고, 복셀 스캔라인에서 동일한 길이를 건너뛰게 함으로써 구현할 수 있다 [5,7,12,17]. 그러나, 원근 투영 알고리즘에서는 종료된 런의 길이와 이에 대응되는 복셀 런의 길이가 다르므로 이것을 계산해야 한다. 현재 가리키고 있는 화소의 좌표가 (i_1, j)이고 종료된 런의 길이가 r일 때, 화소를 가리키고 있는 포인터는 (i_1+r, j)로 변경되고, 이에 대해 복셀 포인터는 $\mathbf{V}_k^{i_1+r}.x$ - $\mathbf{V}_k^{i_1}.x$ 만큼 전진하게 된다. 본 알고리즘에서는 위의 두 기법을 효율적으로 적용할 수 있으므로 알고리즘을 더욱 빠르게 수행할 수 있다.

3.3 알고리즘

본 알고리즘은 크게 렌더링 단계와 2차원 와핑 단계로 나누어진다. 렌더링 단계에서는 우선 주어진 시각 변환에 따라 두 종류의 원형들을 만든다. 원형은 본격적인 렌더링 단계에 앞서 만들어지나 전체 렌더링 시간에 포함되는 것이다. 즉, 새로운 프레임을 생성할 때마다 원형은 그때 주어진 시각 변환에 따라 새로 만들어진다. 원형을 만든 후, 렌더링 단계에서는 볼륨을 슬라이스를 순서대로 가져와서 재샘플링을 하며 각 샘플을 해당하는 화소로 합성한다. 와핑 단계에서는 효율적인 2차원 와핑 기법을 사용하여 최종이미지로 변환시킨다 [14].

```
// 렌더링 단계
COP로부터의 거리에 따라 구역을 나눈다.
for 각 구역에 대해
   원형 생성;
      for 원형의 k번째 요소에 대하여
        이에 대응되는 앞 뒤 2개의 슬라이스를 가져온다.;
        for 슬라이스의 각 복셀 스캔라인에 대하여
          2개의 슬라이스에서 2줄 씩의 복셀 스캔라인을 가져
          온다.;
          for 복셀 스캔라인의 각 복셀에 대하여
               이웃한 8개의 복셀의 불투명도와 색상 값을 계
               산한다;
               원형에 있는 정보를 사용하여 셀안에 있는 모
               든 샘플을 계산한다. ;
               각 샘플을 해당하는 화소로 합성한다.;
            end for
        end for
      end for
  if 다음 구역이 남아있으면
  then 현재의 중간 화상을 4배 확대;
       각 화소를 해당되는 4개의 화소로 복사;
end for

// 와핑 단계
중간 화상을 와핑하여 최종 화상을 만든다.;
```

원형기반 원형 투영 볼륨 렌더링 알고리즘

4. 실험 결과

본 절에서는 원형기반 객체순서 원근 투영 알고리즘의 성능을 분석한다. 또한, 일반적인 광선 발사법과 화상순서 원형기반 알고리즘과의 비교를 통해, 원형으로 얻는 속도 향상과 객체순서 처리로 인한 속도 향상을

각각 분석한다. 화상순서 원형기반 알고리즘은 단지 비교를 위하여 구현된 것이며, 본 논문에서 제안된 원형을 이용하고 기존의 광선 발사법과 같이 광선을 따라가면서 화상 순서로 렌더링 한다. 화상순서 원형 알고리즘과 광선 발사법의 비교를 통해서는 원형으로 인한 이점을 분석할 수 있으며, 화상순서 원형기반 알고리즘과 본 논문의 객체순서 알고리즘과의 비교를 통해서는 객체순서 처리로 인한 이점을 분석할 수 있다. 본 실험은 333MHz 펜티엄-II 프로세서과 128메가 램을 장착한 PC에서 이루어졌다. 실험 데이타로는 VolPack에서 제공하는 256×256×225의 CT데이타 Head, 256×256×167의 MRI데이타 Brain, 128×128×84의 MRI데이타 Brainsmall, 256×256×110의 CT데이타 Engine 등 4개를 사용했다 [6]. 본 실험에서의 모든 알고리즘은 재샘플링 필터로 삼선형 보간법을 이용하며, 모두 광선 조기 종료 기법을 사용한다. 이때 사용된 임계 값은 0.95이다.

그림 7은 본 알고리즘과 광선 발사법의 결과를 보여준다. 그림 7(a)은 90° 시야(field of view)로 렌더링한 결과이며, 그림 7(b)는 120° 시야로 렌더링한 것이다. 그림 7(c)은 그림 7(b)의 네모영역을 5배 확대한 것이다. 그림 7을 통해 우리의 알고리즘의 화질이 광선 발사법보다 좋음을 알 수 있다. 우리의 알고리즘은 적응 샘플링을 하므로 샘플링 비율이 일정하게 유지되는 반면, 광선 발사법은 광선이 벌어짐에 따라 샘플링 비율에 떨어지기 때문에 에얼리어싱(aliasing)이 많이 생긴다. 시야의 각도가 커질수록 에얼리어싱 현상은 더욱 커지게 된다.

표 1은 렌더링 시간의 비교와 원형과 객체순서 처리로 인한 속도 향상에 대한 분석을 보여주고 있다. 여기서 속도 향상은 향상된 렌더링 시간 분의 기존 방법으로 인한 렌더링 시간으로 계산 한다. 원형으로 인한 속도 향상은 평균 2.3이다. 이것은 원형을 사용할 때 샘플의 위치를 구하는 연산이 줄고, 재샘플링 가중치를 구하는 연산이 반으로 줄기 때문이다. 객체순서 처리로 인한 속도 향상은 평균 15.9이다. 원형과 객체순서 처리를 사용하여 본 알고리즘은 대화식 속도에 거의 근접한 속도를 원형 투영 렌더링을 할 수 있음을 보여준다.

표 2는 구역의 수에 따른 렌더링 시간의 변화를 보여주고 있다. 구역의 개수는 시야에 따라 달라진다. 시야

표 1 렌더링 시간의 비교 (단위 : 초)

볼륨데이타	광선의 수 (Width×Height)	객체순서 원형기반	속도향상 객체순서 원형기반 대 화상순서 원형기반	화상순서 원형기반	광선발사법	속도향상 화상순서 원형기반 대 광선발사법
Brainsmall(128×128×84)	256×256	0.370	9.5	3.518	8.165	2.32
Brain(256×256×167)	512×512	1.250	22.8	28.550	66.284	2.32
Head(256×256×225)	512×512	1.639	15.9	26.162	59.605	2.27
Engine(256×256×110)	512×512	2.009	15.4	30.840	70.125	2.27

표 2 구역에 수에 따른 렌더링 시간의 변화 (단위: ms) :사용된 데이타는 Brainsmall, MT : 원형을 만드는 시간, RC: 재샘플링 및 합성 시간, SP: 다음구역 화상으로 확장시키는 시간

구역의 수 (시야)	렌더링 단계									와핑 단계 (ms)	총 렌더링 시간 (ms)
	1번째 구역			2번째 구역			3번째 구역				
	MT	RC	SP	MT	RC	SP	MT	RC	SP		
1 (60°)	219			-			-			141	360
	110	109	-								
2 (90°)	192			103			-			141	436
	58	60	74	69	34	-					
3 (120°)	108			444			48			141	741
	30	9	69	65	79	300	44	4	-		

가 60, 90, 120° 일 때, 구역의 개수는 각각 한 개, 두 개, 세 개로 변한다. 즉, 시점이 볼륨에 가까워 질수록 볼륨은 더 많은 수로 분할된다. 렌더링 단계는 원형을 만들고 재샘플링 및 합성을 하는 모든 과정을 포함한 것이다. 표 2를 통해 구역이 많이 나누어 질수록 렌더링 시간이 증가됨을 볼 수 있다. 각 구역을 렌더링하는 시간을 단계별로 나누어 시간을 측정해 볼 때, 원형을 만

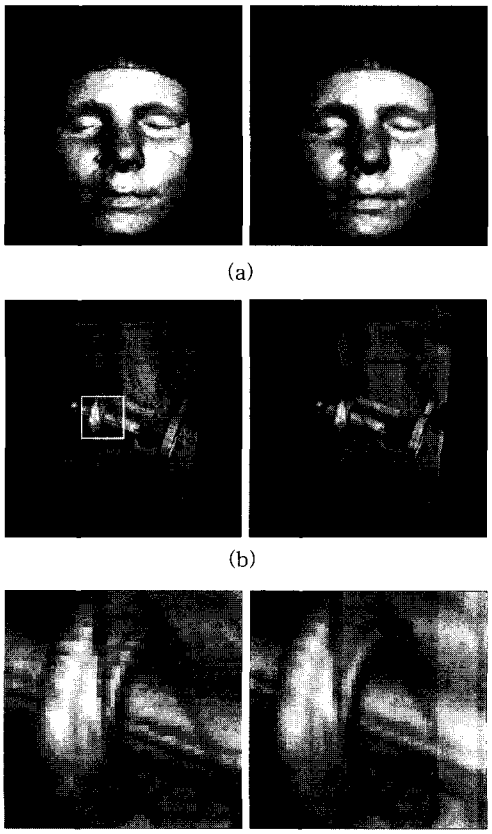

그림 7 (a) Brain 과 (b) Engine 의 원근 영상: 왼쪽에 있는 영상들은 광선 발사법에 의해 생성된 것이며, 오른쪽에 있는 영상들은 본 원형 알고리즘에 의한 것. (c)는 (b)의 네모 영역을 5배 확대시킨 것. (a)의 시야는 90° 이며 (b)의 시야는 120°

드는 시간(MT)과 재샘플링하고 합성하는 시간(RC)은 구역이 몇 개로 나누어 지는 것과 상관이 없이 각 구역에서의 시간을 모두 합하면 거의 일정한 것을 볼 수 있다. 이것은 본 알고리즘이 적응 샘플링을 하여 샘플링 밀도가 각 복셀마다 일정한 수를 유지하기 때문이다. 그러나, 한 구역에서 다음 구역으로 넘어갈 때 중간화상을 전이시키는 과정(SP)에서 많은 시간이 소모됨을 알 수 있다. 본 알고리즘은 이 부분에 있어서의 최적화할 수 있는 기술이 필요하다. 그러나, 구역이 많이 나누인 때에도 전체적인 렌더링 속도는 여전히 다른 방법보다 빠름을 볼 수 있다.

최근, Kreeger외 4인은 슬라이스 순서에 바탕을 둔 원형 광선 발사법 알고리즘을 발표했다 [3]. 그들은 195MHz R10000에서 80×80×127의 볼륨 데이타를 0.91초에 렌더링 했다고 보고했다. 그들의 알고리즘은 샘플을 구할 때 삼선형 보간법보다 연산이 반이나 적고 화질이 떨어지는 쌍선형(bilinear) 보간법을 사용했으며, 또 광선의 수도 우리의 실험인 256×256보다 훨씬 적음에도 불구하고, 우리의 방법보다 렌더링 시간이 더 길게 걸림을 알 수 있다. 또한, Brady의 알고리즘 역시 80×80×127의 볼륨 데이타를 3D그래픽 보조프로세서가 장착된 330MHz 펜티엄II에서 0.696초에 렌더링함을 보고했다 [1-2].

5. 결 론

본 논문에서는 원형과 객체순서 처리를 이용한 효율적인 원근 투영 알고리즘을 제안하였다. 동일한 수평 또는 수직 스캔라인에서 나온 광선들간에 존재하는 일관성을 제시하고 이것을 이용하여 원형을 만드는 기법을 소개하였다. 원형은 샘플을 계산할 때 사용되는 가중치를 가지고 있으므로 샘플을 얻는 연산을 단축시킨다. 또한, 객체순서로 볼륨 데이타를 처리해 갈 때 인접한 복셀들을 가져와 그 안에 있는 샘플들을 동시에 계산할 수 있기 때문에 복셀들을 한번 씩 만 처리할 수 있다. 또한, 본 알고리즘은 원근 광선의 발산으로 인한 샘플링 밀도의 감소 문제를 해결하여, 일정한 샘플링 밀도를 유지할 수 있다. 볼륨을 먼저 주어진 시점으로부터 여러 구역으로 분할한 후에, 다음 구역으로 넘어 갈 때 원형의 간격을 재조정한다.

실험 결과는 본 알고리즘은 일반적인 광선 발사법과 비교해 볼 때, 불규칙한 샘플링으로 인한 에얼리어싱이 줄어듬을 보여주고 있다. 무엇보다도 렌더링 속도가 다른 방법에 비해 매우 향상되었음을 보여준다. 본 알고리즘에서 각 구역을 여러 프로세서에서 병렬적으로 나누어 처리한다면 속도면에서 더욱 향상될 수 있다. 저자들은 현재 본 알고리즘을 이용한 실시간 볼륨 네비게이션을 연구하는 중에 있다. 이는 가상 내시경과 같은 시스

템에서 많이 이용되며, 의료 분야에서 매우 유용하다.

참 고 문 헌

[1] M. Brady, K. Jung, H. Nguyen, and T. Nguyen. Two-Phase Perspective Ray Casting for Interactive Volume Navigation. In Proceedings of Visualization '97, pages 183-189, October 1997. IEEE.

[2] M. Brady, K. Jung, H. Nguyen, and T. Nguyen. Interactive Volume Navigation. IEEE Transactions on Visualization and Computer Graphics, 4(3):243-256, July-September 1998. IEEE.

[3] K. Kreeger, I. Bitter, F. Dachille, B. Chen, and A. Kaufmann. Adaptive Perspective Ray Casting. In Proceedings of Symposium on Volume Visualization, pages 55-62, October 1998. ACM.

[4] K. L. Novins, F. X. Sillion, and D.P. Greenberg. An Efficient Method for Volume Rendering using Perspective Projection. Computer Graphics, 24(5):95-100, November 1990.

[5] P. G. Lacroute. Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation. PhD thesis, Stanford University, 1995.

[6] Lacroute, P. G., VolPack Software Distribution URL http://www-graphics.stanford.edu/software/volpack.

[7] C. Lee, Y. Koo, and Y. Shin, Template-Based Rendering of Run-Length Encoded Volumes, In Proceedings of Pacific Graphics '97, pages 139-147, 1997 and the Journal of Visualization and Computer Animation, Volume 9, Number 3, pages 145-161, 1998.

[8] C. Lee and Y. Shin, A Terrain Rendering Method Using Vertical Ray Coherence, The Journal of Visualization and Computer Animation, 8(2) : 97-114, April-June 1997.

[9] C. Lee and Y. Shin, An Efficient Ray Tracing Method for Terrain Rendering, Proceedings of Pacific Graphics '95, pages 180-193, 1995.

[10] M. Levoy, Display of Surfaces from Volume Data, IEEE Computer Graphics & Applications, Volume 8, Number 5, pages 29-37, May 1988.

[11] M. Levoy, Efficient ray tracing of volume data, ACM Transaction on Graphics, 9(3) : 245-261, July 1990.

[12] Y. Koo, C. Lee, and Y. Shin, Stereoscopic Volume Rendering Using Templates, In Proceedings of IEEE Visualization '98 Late Breaking Hot Topics, North Carolina, October 1998.

[13] Y. Koo, C. Lee, and Y. Shin, Object-Order Template-Based Approach for Stereoscopic Volume Rendering, to appear in the Journal of Visualization and Computer Animation.

[14] T. Porter, and T. Duff, Compositing Digital Images, Computer Graphics Volume 18 Number 3, pages 253-259, July 1984.

[15] R. Yagel and A. Kaufman, Template-Based Volume Viewing, Computer Graphics Forum, Volume 11, Number 3, pages 153-167, September 1992.

[16] R. Yagel and K. Ciula, High Quality Template-Based Volume Rendering, Ohio State University Technical Report, OSU-CISRC-10/93-TR35, 1993.

[17] 구윤모, 이철희, 신영길, 복수 원형을 이용한 볼륨 렌더링에서의 효율적인 수퍼샘플링, 한국 정보과학회 논문지(A), 26권 1호, 1999년 1월.

구 윤 모

1992년 서울대학교 공과대학 컴퓨터공학과 졸업(학사). 1994년 서울대학교 공과대학원 컴퓨터공학과 졸업(석사). 2000년 서울대학교 공과대학원 컴퓨터공학과 졸업(박사). 현재 서울대학교 박사후 연구원. 관심분야는 컴퓨터 그래픽스, 볼륨 렌더링 등

이 철 희

1991년 서울대학교 공과대학 컴퓨터공학과 졸업(학사). 1993년 서울대학교 공과대학원 컴퓨터공학과 졸업(석사). 1998년 서울대학교 자연과학대학원 전산과학과 졸업(박사). 1998년 ~ 2000년 단국대학교 컴퓨터공학과 전임강사(겸임). 현재 (주) 엑스폼닷컴 대표이사. 관심분야는 지형렌더링, 영상기반 렌더링, 볼륨 렌더링 등

신 영 길

정보과학회논문지 : 시스템 및 이론 제 27 권 제 1 호 참조