이중 도약을 이용한 효율적인 공간 도약법
(An efficient space-leaping method using double leaping)

이 정 진 † 신 병 석 †† 신 영 길 †††
(Jeong-Jin Lee) (Byeong-Seok Shin) (Yeong-Gil Shin)

요 약 공간 도약법(space leaping)은 가속화된 영상순서 볼륨 렌더링(image-order volume rendering) 방법의 하나로서 광선 추적 시 빈 공간을 식별하여 도약하도록 함으로써 렌더링 속도를 향상시킨다. 이 방법은 렌더링 속도는 빠르지만 공간 도약을 위한 자료구조를 만들기 위한 전처리 시간이 오래 걸리는 문제가 있다. 본 논문에서는 공간 도약법에서 도약 거리를 기존의 방법보다 두 배로 하는 미리 보기 샘플링 방법을 제안한다. 이 방법은 렌더링 속도의 큰 변화없이 전처리 과정인 거리 맵 생성의 시간을 단축시킬 수 있으며, 기존의 방법보다 렌더링 속도를 빠르게 하는 효과도 있다.

키워드 : 공간 도약, 볼륨 렌더링, 거리 변환

Abstract Space leaping is one of accelerated image-order volume rendering. This method accelerates rendering speed by finding and leaping the empty space. Although its rendering speed is very fast, it takes long pre-processing time to make the data structure to leap the space. In this paper we propose the look-ahead sampling algorithm to double the leaping distance comparing with previous approaches. This algorithm reduces the preprocessing time to make the distance map without significant changes of rendering time. Also, it accelerates the rendering time.

Key words : space leaping, volume rendering, distance transform

1. 서 론

영상 순서 볼륨 렌더링의 주된 가속화 방법으로는 공간 도약법[1,2], 조기 광선 중단법(early ray termination)[3], 원형 기반 렌더링(template-based rendering)[4,5]이 있다. 공간 도약법은 투명한 복셀들이 대응하는 픽셀 값 계산에 기여를 하지 않는 사실을 이용하여, 투명한 복셀들을 처리하지 않도록 함으로써 렌더링 속도를 향상시키는 방법이다. 조기 광선 중단법은 광선이 전진하면서 복셀들의 색상 값을 누적해 나갈 때 누적된 값이 임계치를 초과하면 광선의 진행을 멈추는 방법이다. 이 방법을 사용하면, 불필요한 광선의 진행을 막아 렌더링 시간을 줄일 수 있다. 원형 기반 렌더링은 광선의 진행 경로에 대한 원형(template)을 미리 만들어 두어서 렌더링 시간의 계산량을 줄임으로써 렌더링 속도를 빠르게 한다.

공간 도약법은 전처리 과정에서 각 복셀들에서 가시화 하려는 경계 복셀(boundary voxel)까지의 최단 거리를 계산하여 거리맵(distance map)이라는 공간 자료구조를 만들고, 광선 진행 시에 이 정보를 바탕으로 투명한 복셀들을 방문하지 않고 비약하도록 함으로써 렌더링의 속도를 향상시키게 된다. 그러나 공간 해상도가 N^3인 3차원 볼륨 데이타의 거리 변환(distance transformation)은 $O(N^4)$ 또는 $O(N^3 \log N)$의 복잡도를 가지기 때문에 상당한 시간이 소요되는 문제가 있다[6].

본 논문에서는 거리 맵의 값을 이용하여 투명한 복셀들을 도약할 때, 기존의 방법보다 두 배의 거리를 도약하도록 함으로써 렌더링 속도를 향상시킬 수 있는 방법을 제안한다. 또한, 거리 맵 생성 시에 최대 도약 거리를 반으로 줄여도 한 번에 두 배의 거리를 도약하기 때

· 이 논문은 과학재단 특정기초과제(과제번호: R01-2001-000-00359-0)의 지원으로 쓰여졌습니다.
† 비 회 원 : 서울대학교 컴퓨터공학부
finkl@cglab.snu.ac.kr
†† 정 회 원 : 인하대학교 컴퓨터공학부 교수
bsshin@inha.ac.kr
††† 종신회원 : 서울대학교 컴퓨터공학부 교수
yshin@cglab.ac.kr
논문접수 : 2002년 1월 8일
심사완료 : 2002년 12월 3일

문에 렌더링 속도에 큰 변화 없이 거리 맵 생성 시간을 단축시킬 수 있다.

논문의 구성은 다음과 같다. 2장에서는 기본적인 공간 도약법을 설명하고 문제점을 알아본다. 3장에서 이 문제를 해결할 수 있는 미리 보기 샘플링 방법을 자세히 설명한다. 4장에서는 실험을 통해 본 연구에서 제안된 방법의 속도 향상 효과와 화질을 살펴보고, 5장에서 결론을 맺는다.

2. 공간 도약법

공간 도약법은 투명한 복셀들을 식별하고 도약거리를 저장하기 위해 거리맵을 만드는 전처리 과정과 거리맵을 이용하여 최종 영상을 만드는 렌더링 과정으로 이루어진다.

전처리 과정에서 생성되는, 볼륨 데이타 V(x, y, z)에 대한 거리맵 D(x, y, z)은 다음과 같이 정의할 수 있다.

v(x, y, z)가 경계 복셀이면,

$$D(x, y, z) = 0$$

v(x, y, z)가 투명한 복셀이면,

$$D(x, y, z) = \min\left(\bigcup_{i,j,k=0}^{L,M,N} \Psi(v(x,y,z), v(i,j,k))\right)$$

$\Psi(v_1, v_2)$ 는 두 복셀 간의 거리 함수

v(i, j, k) 는 경계 복셀

두 복셀 사이의 이상적인 값은 유클리드 거리 (Euclidean distance)이지만, 이것을 계산하기 위해서는 복잡한 실수 연산이 필요하므로 계산시간이 증가한다[7]. 따라서 본 논문에서는 유클리드 거리의 근사 값인 체스판 거리(chess-board distance)를 사용하였다[8]. 이차원 데이타에 대해 체스판 거리변환 함수를 이용하여 만든 거리 맵의 예를 그림 1에 제시하였다. 렌더링 시에는 광선을 진행할 때, 거리맵에 저장된 거리만큼 공간을 도약한다. 거리 값이 0인 경계 복셀을 만나면, 복셀의 밀도 값을 참조하여 밝기를 구해 해당 픽셀에 더해주면 된다. 그림 1(a)는 거리맵을 사용하지 않는 렌더링 방법을 나타낸 것으로 광선은 경계 복셀을 찾을 때까지 복셀을 4회 방문한다. 선형 보간을 사용한 색 결정은 네 복셀 모두에서 계산된다. 그림 1(b)에서처럼 거리맵을 사용하면 광선은 2개의 복셀만 방문하게 된다. 이 때, 첫 번째 복셀은 단순히 거리값 참조를 위해서 방문하게 되고, 선형 보간을 사용한 색 결정은 두 번째 복셀에서만 계산된다. 결국 복셀을 방문하는 횟수가 줄게 되어 렌더링 속도가 빨라지게 된다.

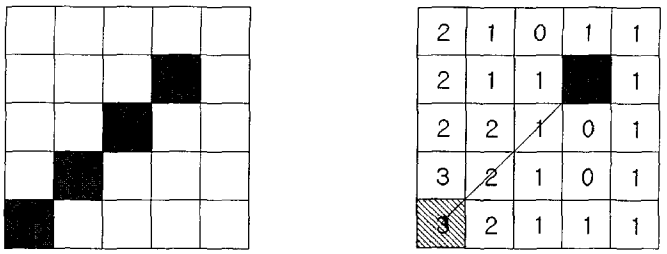
공간 도약법의 문제는 단순한 거리계산 함수를 이용하더라도 3차원 볼륨 데이타의 거리변환에는 상당한 시간이 소요된다는 것이다. 공간 해상도가 N^3인 3차원 볼륨 데이타의 거리 변환(distance transformation)은 $O(N^4)$의 비용이 소요된다. 이 값은 최악의 경우 현재 복셀에서 경계 복셀까지의 거리가 볼륨 데이타의 최대 크기와 같아질 수 있기 때문이다. 따라서 효율적인 렌더링을 위해서는 전처리 시간을 단축시키는 방법이 필요하다.

3. 이중 도약법(double leaping method)

3.1 이중 도약법의 기본원리

일반적인 경우 특정 복셀에서 경계 복셀까지의 평균 거리 d_{min}은 N보다 작기 때문에 실제 응용에서는 일회 최대 도약거리를 d_{max}(<<N)로 제한한다. 즉 d_{max}보다 큰 값은 모두 d_{max}로 설정된다. 이때 전처리 시간은 d_{max}값에 선형 비례하게 되므로 d_{max}값을 감소시키면 전처리 시간도 감소된다.

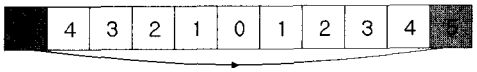
이중 도약법은 d_{max}값을 기존 값의 절반으로 하면서도 동일한 렌더링 효과를 낼 수 있도록 하는 방법이다. 이중 도약법에서 공간을 도약할 때, 거리 값이 d_{max}인



(a) 거리맵을 사용하지 않는 경우의 샘플링 패턴 (b) 거리맵을 사용하는 경우의 샘플링 패턴

그림 1 체스판 거리를 이용한 이차원 거리맵과 렌더링 방법 비교

복셀에서는 $2d_{max}-1$만큼 도약하도록 한다. 이 때, $2d_{max}$만큼 도약할 수 있으나 그림 2(a)와 같은 경우에는 렌더링이 되어야 할 거리값이 0인 surface voxel을 지나치게 된다. 따라서, 그림 2(b)와 같이 $2d_{max}-1$만큼 도약해야 한다. 최대거리 값을 가지는 복셀들이 연속된 경우에는 대부분 한번 도약한 후에 재도약하는 경우가 많다. 이 경우에 한번에 최대거리의 두 배를 도약하도록 함으로써 도약 횟수를 줄여서 렌더링 시간을 빠르게 할 수 있다. 따라서 이중 도약법을 이용하면 기존 방법과 동일한 거리변환을 하는 경우 거의 두 배의 렌더링 속도 향상을 기대할 수 있으며, 동일한 렌더링 속도를 유지할 경우 기존 방법에 비해 최대 도약거리를 절반으로 줄일 수 있다.



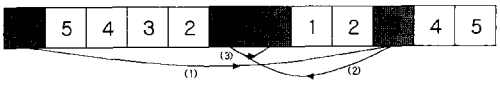
(a) $2d_{max}$만큼 도약할 경우

(b) $2d_{max}-1$만큼 도약할 경우

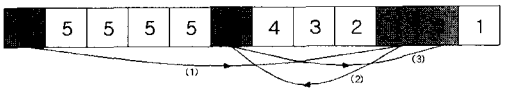
그림 2 이중 도약 거리의 비교

이 방법을 사용할 경우 물체의 경계 복셀이 아니라 내부 복셀로 도약할 수 있다는 것이 문제이다. 이 문제를 해결하기 위해서 내부 복셀로 도약한 경우는 $d_{max}-1$만큼 후퇴한 후에 일반적인 공간 도약법으로 전환한다.

내부 복셀로 도약할 때, 도약한 후의 위치는 그림 3과 같은 두가지 형태를 갖는다. 그림 3(a)와 같이 surface voxel을 지나친 경우에는 $d_{max}-1$만큼 후퇴하여 일반적인 공간 도약법으로 전환하게 되면, surface voxel을 놓치지 않게 된다. 그림 3(b)의 경우도 이와 같다.



(a) surface 복셀을 지나친 경우

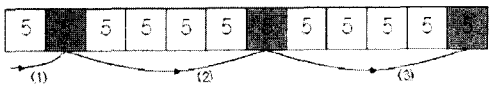


(b) surface 복셀을 지나치지 않은 경우

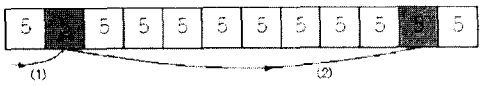
그림 3 내부 복셀 도약의 형태

3.2 이중 도약법의 속도 향상 효과

이 절에서는 공간 도약의 기존의 방법과의 비교를 통해서 이중 도약법의 속도 향상을 기술한다. $d_{max}=5$이고, 최대거리 값을 갖는 복셀들이 연속되어 있다고 하자. 기존의 방법은 그림 4(a)와 같이 d_{max}만큼의 공간을 두 번 도약하게 되어 거리맵을 3회 참조하게 된다. 이중 도약법의 경우에는 그림 4(b)와 같이 $2d_{max}-1$만큼의 공간을 한 번 도약하게 되어 거리맵을 2회 참조하게 된다. 중간에 있는 복셀이 모두 투명하므로 공간 도약은 렌더링된 이미지에 영향을 주지 않는다. 그림 4(a), (b)를 비교해 보면, 이중 도약법을 사용하면 광선이 비슷한 거리를 진행할 때 거리맵 참조는 1회 줄어서 기존의 방법보다 렌더링 속도가 빨라지게 됨을 알 수 있다. 즉 최대 거리값을 가지는 복셀열 들이 반복되는 경우 렌더링 속도는 기존 방법의 약 두 배가 된다.



(a) 기존의 방법을 이용한 공간 도약 형태



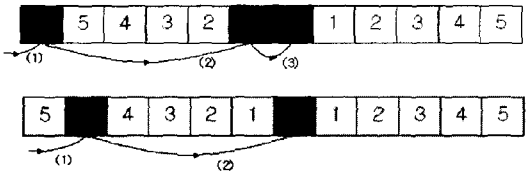
(b) 이중 도약법을 이용한 공간 도약 형태

그림 4 투명한 복셀열

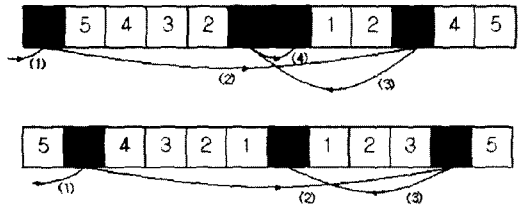
다음으로 경계 복셀을 포함한 복셀열을 생각해보자. 기존의 방법은 그림 5(a)와 같이 거리맵에 저장된 값만큼의 공간을 도약하여 경계 복셀을 찾게 된다. 이중 도약법의 경우에는 그림 5(b)와 같이 $2d_{max}-1$만큼의 공간을 도약하기 때문에 경계 복셀을 지나치는 경우가 발생할 수 있다. 이 경우에 진행한 광선을 $d_{max}-1$만큼 뒤로 이동시킨 후 기존의 방법을 이용하여 경계 복셀을 찾게 된다. 그림 5(a), (b)에서 보는 바와 같이 이중 도약법을 사용하면 경계 복셀을 찾을 때 거리맵 참조 횟수는 기존의 방법보다 1회 증가하는 경우가 발생한다.

실제 응용에서는 경계 복셀보다 경계가 아닌 복셀이 많으므로 그림 5의 경우보다 그림 4의 경우가 많게 된다. 따라서, 이중 도약법을 적용하면 렌더링 속도가 빨라진다. 이중 도약의 횟수를 n_{double}, 뒤로 복귀하는 횟수를 n_{back}이라고 하자. 같은 시점의 방향에 대해서 기존의 방법의 공간 도약의 횟수가 $n_{previous}$라고 하면, 이중 도약법을 사용하게 되면, 도약 횟수는 근사적으로 $n_{previous}$

$-(n_{double}-n_{back})$이 된다. 렌더링 시간은 도약의 횟수에 비례하므로, 렌더링 속도는 이중 도약의 횟수에서 뒤로 복귀하는 횟수를 뺀 것에 비례하여 증가한다.



(a) 기존의 방법을 이용한 공간 도약 형태



(b) 이중 도약법을 이용한 공간 도약 형태

그림 5 경계 복셀을 포함한 복셀열

또한, 최대거리를 줄일수록 최대거리 값을 갖는 복셀의 수가 늘어나게 된다. 이에 따라 그림 4와 같은 복셀열이 증가하므로, 이중 도약의 횟수가 늘어나고, 뒤로 복귀하는 횟수는 줄어든다. 따라서, $(n_{double}-n_{back})$이 증가하여 이중 도약법을 통해서 얻는 속도 향상은 증가하게 된다. 아래의 알고리즘 1은 이중 도약법을 의사 코드 형태로 나타낸다.

최대거리를 d_{max}로 한 경우와 최대거리를 $2d_{max}$로 한 경우에 거리맵을 만드는 시간은 최대 거리에 비례하여 증가하게 된다. 하지만, 이중 도약법을 사용하면, 최대 거리가 d_{max}인 경우에도 최대 거리가 $2d_{max}$인 경우와 비슷한 정도의 공간 도약 효과를 낼 수 있다. 광선 역진행에 따라 발생하는 약간의 렌더링 시간 증가를 감수한다면, 이중 도약법을 사용하여 거리맵을 만드는 시간을 줄일 수 있다.

```
Double_leaping(Distance_Volume D)
{
        d = GetDistanceValue(CurrentVoxel) ; //현재
            복셀의 거리 값
        if (d == dmax)
        {
                ray=ray+(2*dmax-1);
                d=GetDistanceValue(NewVoxel);//새로
                    운 복셀의 거리 값
                if (d != dmax)
                  ray=ray-(dmax-1); //경계 복셀을 지
                      나쳤을 경우 뒤로 복귀
        } else
                ray = ray + d;
}
```

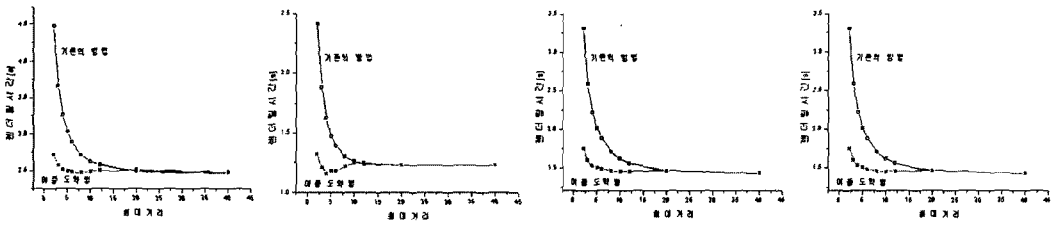
알고리즘 1 이중 도약법의 알고리즘

4. 실험 결과

실험은 Pentium III 466MHz CPU와 512MB의 메모리를 장착한 PC에서 이루어졌고 프로그램은 Visual C++을 사용하여 작성하였다. 실험 데이타는 256*256*225 해상도로 인간의 두부를 CT촬영한 bighead 데이타를 사용하였다.

4.1 이중 도약법의 속도 비교

최대 거리가 40, 20, 12, 10, 8, 6, 5, 4인 경우에 대해서 네 개의 시점의 방향(viewing direction)에 대해서 기존의 방법과 이중 도약법을 사용한 경우에 렌더링 시간을 비교하여 그림 6에 결과를 나타내었다. 결과에 제시한 네 개의 경우는 속도 향상이 적은 경우부터 최대 속도 향상을 얻을 수 있는 경우까지의 범위를 대표할 수 있는 것을 선택한 것이다. 최대거리가 줄어들면, 공



(a) case 1 (b) case 2 (c) case 3 (d) case 4

그림 6 최대거리에 따른 기존의 방법과 이중 도약법의 속도 비교

간 도약의 효과가 감소하여 기존의 방법은 렌더링 시간이 많이 증가한다. 하지만, 이중 도약법의 경우는 렌더링 시간이 기존의 방법에 비해서 조금 증가한다. 이 결과로 최대거리를 줄여도 이중 도약법을 사용하면, 적절한 렌더링 속도를 얻을 수 있다는 것을 알 수 있다.

최대거리에 따른 이중 도약법을 통해서 얻을 수 있는 속도 향상이 그림 7에 그래프로 나타나있다. 그림의 Y축에 표시된 속도 향상은 다음과 같이 정의된다.

$$\text{속도 향상} = \frac{\text{렌더링 시간}_{\text{기존의 방법}} - \text{렌더링 시간}_{\text{이중 도약법}}}{\text{렌더링 시간}_{\text{이중 도약법}}} \times 100[\%]$$

최대거리가 줄어들수록 속도 향상이 커지는 것을 알 수 있고, 이것은 최대 거리를 갖는 연속된 복셀 열의 수가 증가하기 때문이다. 특히, 최대거리가 5이하의 경우에는 속도 향상이 가장 적은 경우에도 기존의 방법에 비해서 10%이상이 되고, 최대 90%의 향상도 확인할 수 있다.

또한, 이중 도약법을 적용했을 경우에 이중 도약의 횟수와 뒤로 복귀하는 횟수를 비교한 결과가 그림 8에 나타나있다. 그림의 Y축에 표시된 도약 성공률은 다음과 같이 정의된다.

$$\text{도약 성공률} = \frac{\text{이중 도약 횟수} - \text{뒤로 복귀 횟수}}{\text{이중 도약 횟수}} \times 100[\%]$$

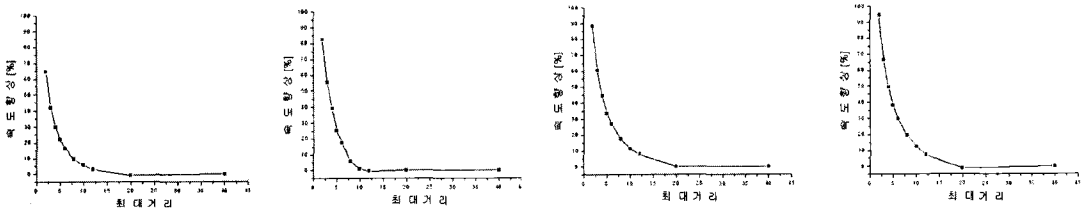
이것은 기존의 방법에 비해 이중 도약법을 적용할 경우에 얻게 되는 거리맵 참조 횟수의 감소를 의미한다. 이 비율과 그림 7에 나타낸 속도 향상을 비교해 보면, 두 값이 대체적으로 비례함을 알 수 있다.

4.2 이중 도약법의 적용과 거리맵 생성

최대 거리가 d_{max}인 거리 맵에서 이중 도약법을 사용하면, 최대 거리가 $2d_{max}$인 경우에 기존의 방법과 비슷한 공간 도약 효과를 얻을 수 있다. 이 경우에 대해서 최대거리가 각각 (20, 10), (12, 6), (10, 5), (8, 4), (6, 3), (4, 2)일 때, 두 방법의 렌더링 시간을 비교하여 그림 10에 나타냈다. 이 때 이중 도약법을 통해 얻을 수 있는 렌더링 속도 향상을 그림 11에 나타냈다. 최대 거리가 줄어들면, 최대 거리가 d_{max}에서 이중 도약법을 사용하는 경우가 최대거리가 $2d_{max}$에서 기존의 방법을 사용하는 경우보다 렌더링 시간이 빨라진다. 그림 9에서 최대 거리가 $d_{max}=2$인 거리 맵과 최대 거리가 $2d_{max}=4$인 거리 맵을 살펴보자. 그림 9(a)를 보면, 이중 도약법의 경우 $2d_{max}-1$만큼 도약하므로, 2회 도약하게 된다. 그림 9(b)를 보면, 기존의 방법의 경우 d_{max}만큼 도약하므로, 3회 도약하게 된다. 이러한 경우가 뒤로 복귀하는 횟수보다 많게 되면, 이중 도약법을 사용하는 경우가 렌더링 시간이 빨라지게 된다.

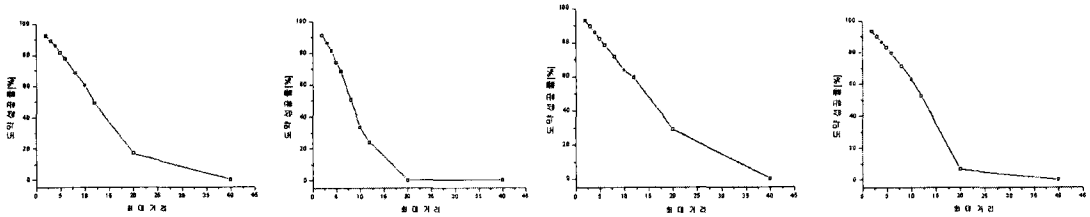
(a) 이중 도약법 (b) 기존의 방법

그림 9 최대거리에 따른 이중 도약법(최대 거리=d_{max})과 기존의 방법(최대 거리=$2d_{max}$)에 대한 도약 횟수 비교



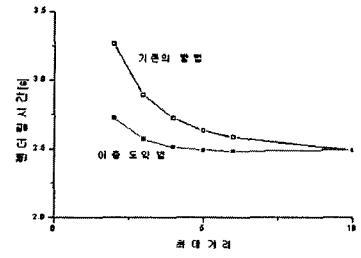
(a) case 1 (b) case 2 (c) case 3 (d) case 4

그림 7 최대거리에 따른 렌더링 속도 향상의 변화

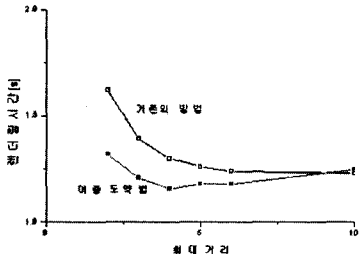


(a) case 1 (b) case 2 (c) case 3 (d) case 4

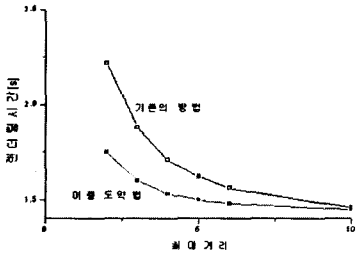
그림 8 최대거리에 따른 도약 성공률의 변화



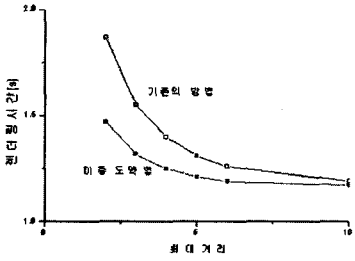
(a) case 1



(b) case 2

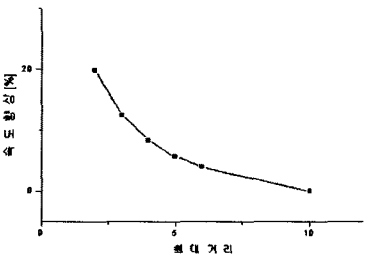


(c) case 3

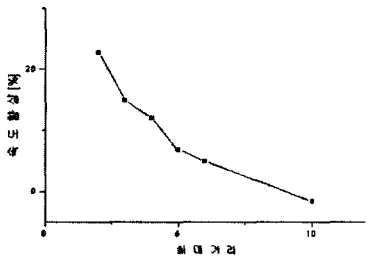


(d) case 4

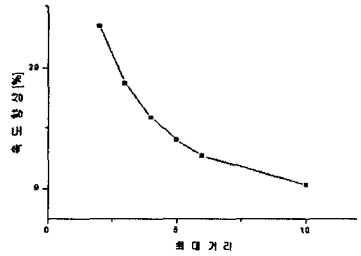
그림 10 최대거리에 따른 이중 도약법(최대 거리=d_{max})과 기존의 방법 (최대 거리=$2d_{max}$)에 대한 렌더링 시간 비교



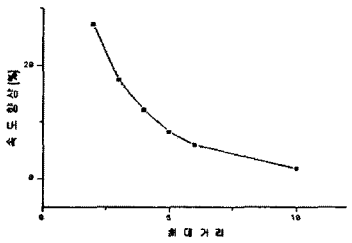
(a) case 1



(b) case 2



(c) case 3



(d) case 4

그림 11 최대거리에 따른 이중 도약법(최대 거리=d_{max})과 기존의 방법(최대 거리=$2d_{max}$)에 대한 렌더링 속도 향상의 변화

표 1 거리맵 생성 시간 비교

d_{max}	거리맵 생성 시간 (최대 거리=$2d_{max}$) [s]	거리맵 생성 시간 (최대 거리 =d_{max}) [s]	거리맵 생성 속도 향상 [%]
10	56.78	45.68	24.2
6	48.48	39.48	22.7
5	45.68	37.69	21.1
4	43.02	35.67	20.6
3	39.48	33.41	18.1
2	35.67	32.09	11.1

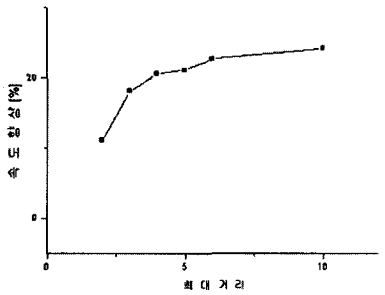


그림 12 최대거리에 따른 최대 거리=d_{max}인 경우와 최대 거리=$2d_{max}$인 경우의 거리맵 생성 속도 향상의 변화

또한, 이 때 얻을 수 있는 거리맵 생성 시간의 이득이 표 1에 주어졌다. 이 때 거리맵 생성 속도의 향상은 그림 12에 나타냈다. 10~20% 정도의 거리 맵 생성 시간의 단축을 확인할 수 있다.

4.3. 화 질

기존의 방법과 이중 도약법의 경우에 화질을 비교하기 위해서 그림 13의 왼편과 가운데에 각각 기존의 방법으로 얻어진 이미지와 이중 도약법을 사용하여 얻어진 이미지를 나타내었다. 두 이미지의 픽셀간 밝기 차이를 오른편 이미지에 밝기로 나타내었다. 이 그림에서 모든 픽셀의 밝기가 0임을 알 수 있고, 이것은 두 이미지가 똑같은 영상임을 말해 준다. 즉, 이중 도약법이 surface voxel을 빠뜨리지 않고 방문한다는 것을 실험을 통해서 확인할 수 있다.

5. 결 론

본 논문에서는 이미지-기반 볼륨 렌더링에서 거리 맵을 이용한 공간 도약 시 최대 거리의 두 배를 도약하는 방법을 제안하였다. 기존의 방법에 비해서 한 번 도약 시 더 많은 거리의 공간을 도약하게 되고, 도약하는 공간에 있는 복셀들은 투명하기 때문에 렌더링에 영향을 미치지 않는다. 따라서, 화질의 저하 없이 도약 횟수가 줄게 되어 렌더링 속도가 빨라진다.

(a) image 1

(b) image 2

그림 13 화질 비교

실험 결과에 의하면, 이중 도약법을 사용할 경우에 렌더링 속도 향상은 관찰자의 시점 방향에 따라 달라지지만, 최대거리가 5이하인 경우에는 최대 90% 정도의 속도 향상을 얻을 수 있다. 이 때, 화질의 변화는 없었다. 이중 도약법의 경우 최대 거리를 감소하여도 렌더링 속도는 기존의 방법과 같이 많이 증가하지 않았다. 이 실험 결과를 바탕으로 최대 거리를 2dmax로 한 경우 기존의 방법으로 렌더링을 하는 것과 최대거리가 dmax인 경우 이중 도약법을 사용한 경우의 렌더링 시간이 차이가 거의 없음을 보였다. 또한, 최대거리가 작을 경우에는 이중 도약법을 사용한 경우의 렌더링 속도가 최대거리가 반임에도 빨라지는 것을 알 수 있었다. 결국, 이 방법을 이용하여 최대 거리에 비례하는 거리 맵 생성 시간을 단축할 수 있음을 보였다.

참 고 문 헌

[1] Yagel, R. and Shi, Z., "Accelerating volume animation by space-leaping," *Proceedings of IEEE Visualization '93,* pp.62-69. 1993.

[2] Sramek, M. and Kaufman, A., "Fast ray-tracing of rectilinear volume data using distance trans forms," *IEEE Transactions on visualization and computer graphics*, Vol.6, No.3, pp. 236-252, 2000.

[3] Danskin, J. and Hanrahan, P., "Fast algorithms for volume ray tracing," *Workshop on Volume Visuali zation,* pp. 91-98, 1992.

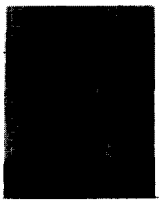
[4] Cheol-Hi Lee, Yun-Mo Koo, and Yeong Gil Shin, "Template-based rendering of run-length encoded volumes," *Proceedings of the Fifth Pacific Conference on Computer Graphics and Applica tions,* pp.138-147, 1997.

[5] Roni Yagel, and Arie Kaufman, "Template-based volume rendering," *Eurographics '92 Computer Graphics Forum 11 (3)*, pp.153-167, 1992.

[6] Datta, A. and Soundaralakshmi, S., "Fast parallel algorithm for distance transforms," *Proceedings of 15th International Parallel and Distributed Processing Symposium*, pp.1130-1134, 2001.

[7] Sharghi, M. and Ricketts, I.W., "A novel method for accelerating the visualisation process used in virtual colonoscopy," *Proceedings of Fifth International Conference on Information Visualisa tion,* pp.167-172, 2001.

[8] De Assis Zampirolli, F. and De Alencar Lotufo, R., "Classification of the distance transformation algorithms under the mathematical morphology approach," *Proceedings XIII Brazilian Symposi um on Computer Graphics and Image Processing*, pp. 292-299, 2000.



이 정 진

2000년 2월 서울대학교 기계공학부 학사. 2002년 2월 서울대학교 컴퓨터공학부 석사. 2002년 3월~현재 서울대학교 컴퓨터공학부 박사과정. 관심분야는 그래픽스 하드웨어, 가상 내시경, 볼륨 렌더링 등

신 병 석

1990년 서울대학교 공과대학 컴퓨터공학과. 1992년 서울대학교 대학원 컴퓨터공학과 석사. 1997년 서울대학교 대학원 컴퓨터공학과 박사. 2000년~현재 인하대학교 컴퓨터공학부 조교수. 관심분야는 실시간 렌더링, 볼륨그래픽스, 의료영상

신 영 길

1982년 2월 서울대학교 계산통계학과 학사. 1984년 2월 서울대학교 계산통계학과 석사. 1990년 2월 미국 USC 전산학과 박사. 1990년 2월~1992년 2월 경북대학교 전자계산학과 전임강사. 1992년 3월~현재 서울대학교 컴퓨터공학부 교수 관심분야는 볼륨 렌더링, 의료 영상 처리, 하드웨어 기반 렌더링