Volume Rendering를 위한 향상된 Sampling 방법

박재영 °, 이병일, 최흥국
인제대학교 전산학과

Improved Sampling Method For Volume Rendering

Jae-Young Park °, Byong-Il Lee, Heung-Kook Choi
Dept, of Computer Science Inje University

요 약

본 논문에서는 volume rendering 기법을 이용하여 2차원 MRI 영상들을 합성하여 3차원 영상 만들 때 보다 해상도를 높이기 위한 개선된 sampling방법을 소개한다. 2차원 슬라이스 영상들이 3차원으로 재구성할 때 voxel 기반으로 렌더링을 하기 때문에 오브젝트의 내부 영역까지도 볼 수 있는 것이 volume rendering의 가장 큰 장점이다. 따라서 영상을 재구성하는 과정에서 보다 향상된 interpolation을 적용시켜서 공간 해상도를 향상시키면 보다 명확하게 오브젝트 내부 정보를 살펴 볼 수 있다. 본 논문에서는 nearest neighbor 이나 linear 같은 interpolation으로 sampling한 방법보다 cubic interpolation을 3차원 공간에서 적용 시켜서 보다 resampling이 잘 되도록 하여 해상도를 향상시켜 보았다. 이렇게 향상된 interpolation 적용시켜서 렌더링할 때 얼마나 오브젝트 내부 영역이 잘 가시화가 되었는지 transfer function을 적용시켜서 오브젝트 내부 정보를 렌더링 해보았고, 임의의 축으로 오브젝트을 잘라서 2D 단면 영상으로 출력해 보았다. 보다 향상된 interpolation을 적용시켜서 resampling을하면 영상 해상도가 개선되었음을 볼 수 있었다.

1. 서론

Volume rendering은 2차원 영상들을 재구성하여 3차원 영상으로 만드는 것인데, 일반적으로 CT나 MRI 같은 슬라이스 영상들을 3차원으로 재구성할 때 많이 쓰인다. 오브젝트 직접적으로 파괴하지 않고 내부 모습을 볼 수 있기 때문에 오늘날 의료, 지형, 기계 분야 등에 많이 쓰인다. 특히 요즘 많은 병원들이 PACS system을 도입하고 있는데, 예전과는 달리 슬라이스 영상들을 필름으로 현상해서 일일이 보관 하는 것이 아니고 PACS 시스템에서는 모든 영상들이 디지털화된 이미지 파일 형태로 저장이 되고 네트웍을 통해서 빠르게 전송이 될 수 있다. 따라서 원하는 영상들을 언제든지 빠른 시간에 찾아 볼 수가 있고, 연속적인 슬라이스 영상들이 많이 필요한 MRI 같은 영상들도 하나의 단위로 묶어서 편하게 볼 수 있다. 이러한 환경에서 2차원 슬라이스 영상들을 쉽고 빠르게 모을 수가 있기 때문에 3차원 영상으로 렌더링 해서 진단에 쓰는 경우가 많이 늘어나고 있다. 2차원 영상들을 3차원으로 재구성을 하면 2차원 영상일 때 보이지 않았던 부분도 볼 수 있고, 인체 내부 조직의 정확한 형태를 보다 객관적으로 살펴 볼 수가 있다. 따라서 volume rendering할 때에는 공간 해상도가 매우 중요한데, 본 논문에서는 volume rendering 전처리 단계에서 interpolation을 통해서 공간해상

도를 향상시킬 수 있는 보다 나은 방법을 제시하고, 오브젝트 내부 모습을 transfer function으로 렌더링 했을 때 얼마나 좋은 결과가 나왔는지 실험 해보았다.

2. Volume rendering과 공간 해상도

2.1 Volume rendering이란?

Volume Rendering은 연속적인 2차원 슬라이스들을 기본으로 하는 3-dimensional datasets들을 z축으로쌓아서 3차원 영상으로 재구성하는 것이다[1]. 일반적으로 Volume Rendering은 다음 그림 1과 같은 과정을 거치는데 이중에서 resampling단계에서 공간 해상도가 결정이 된다.

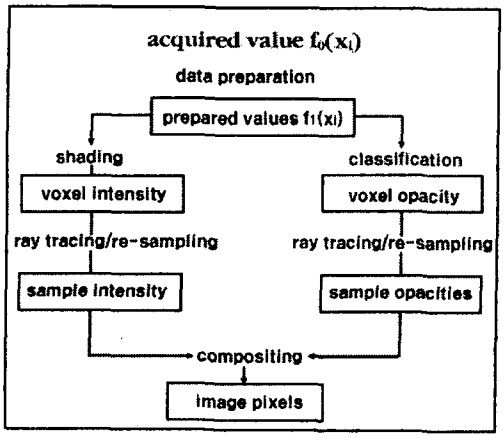


그림 1. Overview of volume rendering pipeline

먼저 sampling할 때에는 그림 2와 같이 ray tracing 할 때와 같이 일어나는데, 주변에 있는 픽셀의 위치와 값을 고려해서 voxel을 데이터를 얻고나서 이 voxel 기반으로 적당한 interpolation을 적용해서 resampling을 하게 된다[2].

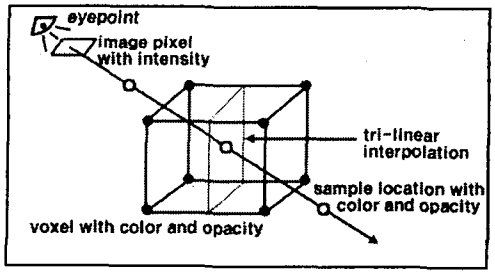


그림 2. Ray tracing/resampling steps

2.2 Resampling을 위한 interpolation

2차원 영상에서 이미지 크기를 변형하면 원래 픽셀과 결과 픽셀 사이에 새로운 픽셀을 생성해 주어야 한다. 이때 주변의 픽셀을 고려해서 새로운 픽셀을 생성하면 더욱 이미지 해상도가 좋아진다. 이때 새로운 결과 픽셀 값을 보다 정확하게 결정하도록 해주는 것이 바로 interpolation이다. 마찬가지로 3차원에서도 다음 그림 3과 같이 각각의 ray가 주변의 sampling된 포인터를 지날 때 정확하게 직진하는 ray와 교차하는 점이 일치하지 않고 조금씩 오차가 생긴다.

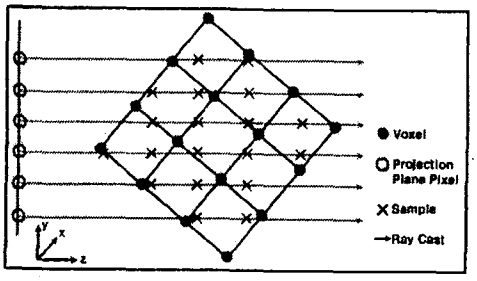


그림 3. Transformed image space and mapping

이런 값들이 resampling 함으로서 ray의 일직전상에 제대로 놓이도록 하면 렌더링 할 때 보다 정확한 값을 가진 최종 픽셀을 결정할 수가 있다. 이때 nearest neighbor나 linear같은 interpolation 보다 cubic 같은 고차수 interpolation을 쓰면 보다 공간해상도가 좋은 결과를 얻을 수 가 있다.

2.3 3차원 bicubic interpolation

cubic convolution interpolation 커널은 다음 그림과 같이 하나의 좌표에 대해 4개의 단위로 구성된다. 따라서 한 차원에 대해서 4번의 곱셈에 필요하며 sampling하고자 하는 주변의 포인터를 보다 많이 고려해서 계산을 하기 때문에 좀더 정확한 값을 얻을 수가 있다[4].

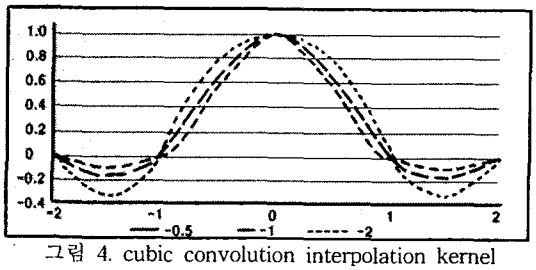


그림 4. cubic convolution interpolation kernel

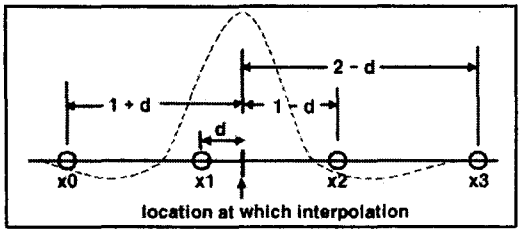


그림 5. Distance from known points to interpolated point

만약 x0, x1, x2, x3 4개의 sampling된 포인터가 있을 때, 이들 값 사이에 resampling된 새로운 값을 얻기 위해서 다음 그림 5와 같이 주변의 포인터를 고려해서 새로운 값 d를 얻기 위해서 그림 4와 같은 커널을 적용시키면 되며 함수는 다음과 같다.

f(d) = c0 * x0 + c1 * x1 +c2 * x2 +c3 * x3
이때 c0,c1,c2,c3은 커널 계수이며 식 (1)과 같이 계산된다.

$$C_0 = a(1+d)^3 - 5a(1+d)^3 + 8a(1+d) - 4a$$
$$C_1 = (a+2)^3 - (a+3)d^2 + 1$$
$$C_2 = (a+2)(1-d)^3 - (a+3)(1-d)^3 + 1$$
$$C_3 = a(2-d)^3 - 5a(2-d)^2 + 8a(2-a) - 4a \quad (1)$$

이런 interpolation이 3차원 공간에서 이루어져야 하는데, 먼저 x축으로 interpolation 커널을 적용시켜서 결과를 얻고 그 결과 값을 가지고 그림 6처럼 다시 y축으로 interpolation 하면 된다. 마지막으로 이렇게 2차원으로 얻어진 값을 이용해서 그림 6처럼 z축으로 적용시키면 3차원 tricubic interpolation이 된다.

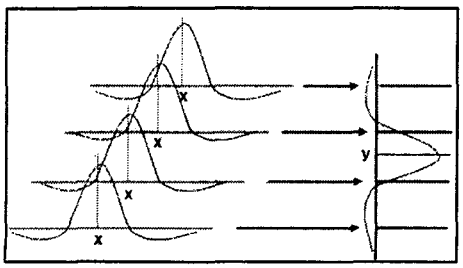


그림 6. Extending cubic convolution interpolation to two dimensions.

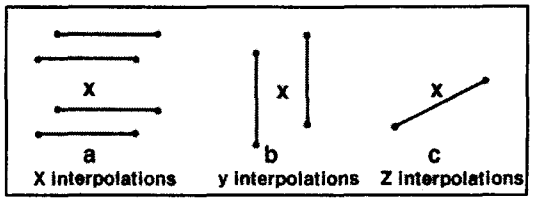


그림 7 (a-c) Interpolation in three dimension

이때 3차원으로 interpolation이 일어나기 때문에 ray casting할 때 현재 어느 위치에서 resampling을 할지 z 값을 알아야 한다. 즉 3차원 공간 속에서 현재 ray가 어느 지점을 지나고 있는지 알아야만 주변의 포인터 값을 고려해서 interpolation을 적용시킬 수가 있고 커널 계수를 식 (1)와 같이 구할 때 변수 d를 결정할 수 있다. 따라서 sampling된 좌표의 위치를 알기 위해서 다음 식 (2)같은 transformation matrix를 정의해서 ray가 지나가는 위치를 결정할 때 사용하였다[3][5].

$$\begin{bmatrix} a & e & i & m \\ b & f & j & n \\ c & g & k & o \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} \quad (2)$$

x-vector z-vector

3. 결과 및 구현

오브젝트 내부 모습을 가시화 하기 위해서 transfer function을 적용시켜서 렌더링 하였다. transfer function은 classification단계에서 이루어지는데, 픽셀의 opacity속성을 할당하는 함수이다. ray casting하면서 쌓아둔 픽셀의 정보를 최종 화면에 투영할 때 다음 식 (3)과 같이 intensity와 gradient를 같이 고려해서 transfer function에 적용시키면 오브젝트 내부의 모습을 보다 다양하게 관찰할 수 있다.

$$a_1 = O(I_i, |\nabla_i|, \ldots, \ldots, \ldots) \quad (3)$$

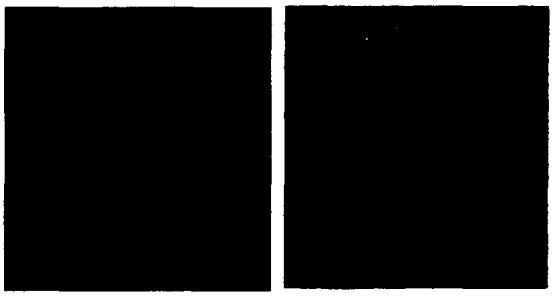
위의 식에서 는 $(O\ldots)$ opacity transfer function, I_i = intensity value, $|\nabla_i|$ = gradient magnitude을 뜻한다.

volume rendering할 때 cubic interpolation이 얼마나 많이 공간해상도 향상에 영향을 주었는지 linear interpolation과 같이 구현해서 비교해 보았

다. 다음 그림 8은 3d-head dataset을 가지고 average Transfer Function을 적용시켜 렌더링한 결과이다.

(a) Linear (b) Cubic

그림 8. Average composition



(a) Linear (b) Cubic

그림 9. Average composition 50-70번째 슬라이스

좀더 오브젝트 내부의 원하는 부분을 자세하게 관찰하기 위해서 그림 9처럼 슬라이스 번호 중에서 50-70번째 사이의 20개 영상만 z 축으로 잘라서 다시 재구성해서 렌더링한 결과이다. 그림에서 보듯이 linear 보다 cubic interpolation방법이 좀더 공간 해상도가 낮은 결과를 볼 수가 있다.
linear은 주변의 8개의 포인터를 고려해서 새로운 값을 생성해서 interpolation하지만 cubic은 주변의 32개의 포인터를 고려해서 결과 값을 생성하기 때문에 각각의 ray와 교차하는 sampling 포인터를 생성할 때 좀더 정확하게 포인터를 잡을 수가 있어서 linear interpolation 방법보다 전체적으로 이미지가 밝고, 선명한 것을 볼 수 있다.
보다 정확한 고등 차수 interpolation을 사용할수록 이미지 해상도를 향상이 되나 그만큼 계산 량이 많아지기 때문에 렌더링 시간이 많이 걸린다. 따라서 계산 량을 줄이기 위해서 Horner의 규칙을 사용해서 계산을 하면 다항식을 계산할 때 곱셈과 덧셈의 횟수가 적어지므로 시간을 줄일 수가 있다. 본 논문에서는 cubic interpolation의 계수를 구할 때 이런 방법을 적용시켜서 계산 횟수 줄였으며, 주변의 픽셀의 위치를 구할 때도 식 (2)를 이용하여 ray casting할 때 쓰이는 vector정보를 이용하여 주변의 픽셀이 쉽고 정확하게 결정하도록 하였다.

5. 결론

volume rendering에서는 2차원 데이터를 3차원 영상으로 재구성하여 오브젝트 내부 모습을 3차원으로 볼 수 있게 해주는 것이 가장 큰 장점이다. 따라서 본 논문에서는 좀 더 향상된 interpolation을 이용하여 공간 해상도를 개선할 수 있는 방법을 제시하였으며, 구현에 필요한 전처리 과정들을 알아보았다. neighbor, linear interpolation같은 방법 보다 해상도가 향상된 영상을 얻을 수 있었으며, transformation matrix를 효율적으로 이용하여 cubic interpolation에 적용시켜서 구현하여 보았다. volume rendering은 아주 많은 데이터를 필요로 하기 때문에 렌더링 시간을 줄일 수 있는 보다 향상된 방법이 요구되며, mmory i/o나 multi processor 면에서 좀더 나은 성능을 볼 수 있도록 프로그램 하면 더욱 정확하고 빠른 결과를 보일 것이다.

[참고문헌]

[1] R.A Drebin L. Carpenter, and P.Hanrahan. "Volume Rendering. Computer Graphics", 22(4):65-74, August 1008.

[2] Marc Levoy, Display of Surfaces from Volume Data, University of North Caronlina, IEEE May1998

[3] OpenGL Architecture Review Board. OpenGL reference Manual. Addison-Wesley Press, 2nd.. 1997

[4] R.G Keys. Cubic Convolution Interpolation forDigital Image Processing. IEEE Transactions on Acoustics, Speech and Processing, ASSP-29(6):1153-1160, December 1981

[5] Donald Hearn, M.Pauline Baker. "Computer Graphics". Prentice-Hall Inc. 1997