Exploring Volume Rendering with Path Tracing

Scott Davis, Xiaoqian Jiang, Greg Nichols, James Cremer*
Department of Computer Science, University of Iowa

1 Introduction

Advanced computer graphics rendering techniques have the potential to greatly enhance the understanding of complex medical datasets, such as MRI and CT scan data.

Traditional volume rendering techniques relies on simple lighting and illumination techniques including: 1) isosurface extraction and 2) direct volume rendering by ray casting. The first technique works poorly when surfaces aren't the primary element of interest. Basic ray casting doesn't suffer from this limitation but also doesn't include information such as shadows that enhance spatial understanding. Ray tracing techniques handle shadows but still lack proper consideration of global illumination. So, we want to do better.

Our aim is to reveal subtle details of volume data by introducing global illumination techniques that account for indirect illumination as well as shadows and direct illumination.

2 Exposure

Our technique approximates the rendering by Monte Carlo Path Tracing at the voxel level. See Figure 1 for an overview of our technique.

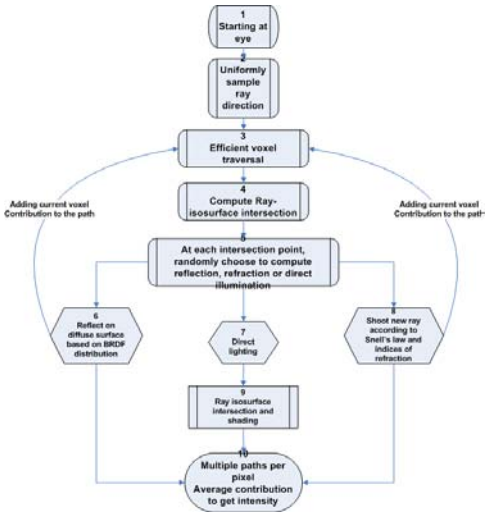

Figure 1: Overview of our method

Some major components of our method are:

2.1 Diffuse surface reflection

Ray tracing produces photorealism for shiny objects, but it does not handle more diffuse objects where color bleeding occurs. This happens mostly on diffuse surfaces from light bouncing off nearby surfaces. In step 6 of our pipeline, we approximate diffuse object reflection using Monte Carlo Path Tracing by sending out rays randomly about the hemisphere around the normal

2.2 Efficient voxel traversal

Monte Carlo Path Tracing elegantly models reflection, refraction and shadows with one major drawback: computation expense. The bottleneck here is the ray object intersection, which takes up to to 95% of the rendering time. We adopted an efficient voxel traversal algorithm to speed up the process. Our 3D Bresenham algorithm is a straightforward extension of the DDA line algorithm. The traversal algorithm breaks the ray into intervals of t, each of which spans one voxel. A ray starts from the origin and visits each of these voxels in interval order.

2.3 Ray isosurface intersection for trilinear boxes

Step 10 of our pipeline consists of two phases. We analytically compute the normal at each intersection and shade the resulting intersection point where an isosurface exists. Computation only involves ray-isosurface intersection, no explicit surface is computed.

3 Results

We implemented our technique in C++ on a Pentium 4, 1.3 GHz CPU machine with 768 MB of RAM. Our dataset is the 64x64x64 fuel injection simulation from *http://www.volren.org/*. We implemented several techniques, including marching cubes, ray casting and path tracing to compare the results and the rendering times.

Marching Cube Ray casting Path tracing

Figure 2: Comparison of different techniques

To our ray tracer, we added path tracing and in Figure 2 is an example of our output. We were able to render our path traced image in 92 seconds using 1 sample per pixel and 651 seconds using 9 samples per pixel to a 200x200 image.

4 Conclusion

Our results provide compelling initial evidence that advanced rendering and illumination techniques can add greatly to medical visualization, and our technique provides one promising direction for future research.

*e-mail: {scodavis|xjia|gbnichol|cremer}@cs.uiowa.edu