X-Ray Casting: Fast Volume Visualization Using 2D Texture Mapping Techniques

Youngser Park, Robert W. Lindeman, and James K. Hahn*

The Laboratory for Advanced Computer Applications in Medicine
The George Washington University†

ABSTRACT

An approach is described for approximating 3D volumetric rendering of medical data for use in surgical simulation. Our approach uses a two-step method we call *X-Ray Casting*. In the first step, rays are cast through the volume, and projected onto a 2D projection surface, producing a 2D texture map. In the second step, the resulting texture is rendered by mapping it onto semi-opaque geometry of the same shape as the projection surface. The first step is performed off-line and is view-independent, while the second step is done at interactive frame rates and is view-dependent. The rendering step takes less time than more traditional volumetric rendering techniques, while still providing reasonable realism for surgical simulation environments. In addition, the approach makes good use of the limited amount of fast texture memory present in many relatively low-cost, high-performance graphics workstations, compared with the large texture memory requirements of high-cost, real-time volumetric rendering systems. This paper reports on work in progress.

1 Introduction

We have developed a prototype system for simulating catheter insertion for surgical procedures, such as those used in the treatment of brain aneurysms. Producing an effective and accurate rendering for vasculature simulation requires several parts. We must:

1. acquire medical data, typically Computed Tomography (CT) or Magnetic Resonance (MR) data,
2. extract the vasculature from this data, and store it in a way that allows for efficient rendering and simulation,
3. represent the surrounding tissue and bone to provide the surgeon with the cues typically found in the actual procedure, and
4. render the vasculature on a workstation in a manner that provides adequate realism for the simulation.

This paper will focus on a new approach to parts 3 and 4 above. The motivation for our approach comes from studying catheter insertion procedures, both live and on video tape. The nature of the images viewed by the surgeons during the procedure suggests a number of simplifications that make them amenable to real-time manipulation on the computer screen. Movement of the fluoroscope camera is limited to three degrees of freedom, namely longitudinal axial translation and rotation, and zooming. This allows us to pre-compute display information based on the limited views possible using these degrees of freedom. Also, because we are concentrating on catheter insertion procedures, we can assume that the surgeon's focus will be on the vasculature. This makes the use of approximations more plausible; the objects located at the focus of the procedure are rendered at a high level of detail, using surface-based approaches, and the surrounding area is approximated, while still maintaining reasonable realism.

CT data is used to recover the original X-Ray value for a particular ray path. The surrounding material is of great importance to the visualization, as it provides the necessary cues that enable the surgeon to successfully relate the current location of the catheter to landmarks in the body. Because there is a certain level of commonality from patient to patient, surgeons learn where certain major vessel bifurcations are in relation to bodily landmarks.

The remainder of this paper is organized as follows. In Section 2 we provide a survey of pertinent previous work done in the field of real-time volumetric rendering. Section 3 describes the application we are developing using our approach. Section 4 gives a detailed description of our X-Ray Casting algorithm. Some results of our implementation are shown in Section 5, followed by conclusions in Section 6.

2 Previous Work

Many algorithms for rendering volume data have been developed [5, 11, 12, 15]. Though volume rendering is a very useful visualization tool, it is often quite slow. Two approaches have been used to speed up volume rendering: special purpose hardware [4], and parallel implementations [13]. Interactive frame rates have recently been achieved in some implementations [10]. Generally, however, the cost of the machines or their lack of general availability make these solutions less attractive.

Many approaches have used graphics hardware to produce volumetric representations of 3D data [2, 7, 8, 9, 16]. In particular, Guan and Lipes [8] implemented a volume renderer on the Kubota Denali system that made use of 3D texture hardware. Fraser's technique [7] uses the graphics hardware found in the SGI RealityEngine to perform fast volume rendering. The approach places the entire volume into the RE's texture memory as a 3D texture. Rendering is accomplished by making several planes that are parallel to the image plane and letting the graphics hardware apply the texture to those planes. The values at the planes are then composited using the graphics hardware alpha channels. Using this technique, the author claims image generation rates of 0.1 seconds for a 256^3 volume into a 256^2 image.

This is a successful technique, however, there appear to be three problems with it. The process requires a large amount of texture memory. The largest amount that can be put into an Infinite Reality graphics system is 64MB, making volumes larger than 400^3 untenable (assuming only one byte per voxel). It appears that if the user wants to change the opacity function, the volume needs to be reprocessed and then reloaded into the texture memory. It also appears that the process may be significantly undersampling the volume when it creates the sample planes. We have attempted to address some of these problems in our approach.

* [parky|gogo|hahn]@seas.gwu.edu

†The George Washington University, Department of EE&CS, 801 22nd St., NW, Washington, DC 20052.

3 Application Description

Figure 1 shows the system structure for our simulation. The CT images are used for two purposes. First, a segmentation process is used to extract the vasculature. The spatial and connectivity information is stored in the blood vessel data structure. The simulation module uses these data, along with data from the force feedback device, to perform the simulation.

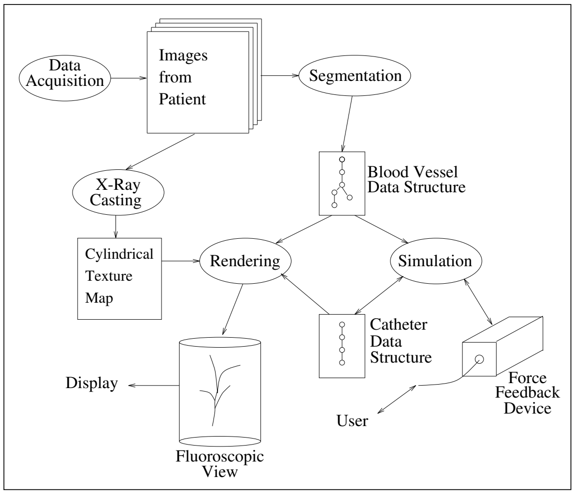

Figure 1: The catheter insertion simulation system structure

The force feedback device provides us with real-time updates of the catheter's position. In addition, it is designed to provide realistic haptic feedback to the surgeon during the simulated procedure. The simulation module computes the momentary forces and torques of the end of catheter, and outputs them to the haptic device. The user receives active feedback of the interaction between the catheter and the vasculature.

Second, the CT images are used to perform X-Ray Casting to produce a cylindrical texture map, as described in detail in Section 4. The rendering process uses the blood vessel data structure to render the vasculature using a surface-based approach. The vasculature is embedded in a cylinder, which is wrapped with the cylindrical texture map. Finally, the catheter data structure is used to render the catheter as a NURBS curve.

4 X-Ray Casting

A unique method for representing the large amounts of volumetric data in a compact form is currently being developed. X-Ray Casting is a two-step process. The first step is performed off-line and is view-independent, while the second step is done in real-time and is view-dependent. The first step involves creating a body map, while the second step involves texturing geometry in the output scene with the computed body map, based on viewer location. This is similar to the way environment mapping wraps a texture around geometry [6].

One of the major issues we needed to solve in refining our approach is selecting the geometry onto which we project the cast rays. At first thought, this seems irrelevant, since the orthographic projection we are using would mean we could use just a rectangle as projection geometry. However, this approach might not be effective as it does not take into account the final direction of view. We need to cast rays from a direction, or point, onto a geometry which can then be viewed from different points of view without re-casting.

Anatomically, some landmarks are in front of the vasculature, and some are behind it. Camera movement during the procedure is constrained, allowing only rotation about, and translation along, the longitudinal axis of the body. For these reasons, we use a cylindrical projection about the position of the vasculature in the CT data. The algorithm for constructing a cylindrical body map from sequential CT images follows, and is shown graphically in Figure 2.

Algorithm 4.1

X-Ray Casting Algorithm:
Set a circle with CylCenter $(0,0)$ and CylRadius r
for $slices$(sequential CT images)
    Load the slice into memory
    for $angle$(0-360)
        Cast a ray from CylCenter to edge of circle, i.e.
        for $radius$(0-CylRadius)
            Get a sample (pixel value)
            Calculate the attenuation factor (see below)
            Set the cylinder surface pixel value
        end
    end
end
Create the output image

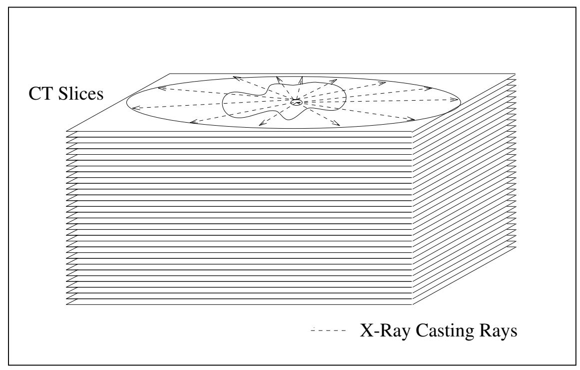

Figure 2: CT stack for X-Ray Casting

The samples from one slice can be thought of as making up one circle of data. If we stack the circles from all the slices, we produce a cylindrical representation of the volume, with the radius of the cylinder equal to CylRadius and the height of the cylinder equal to the number of slices. This map can then be "unwrapped" to form a rectangular (distorted) texture map of the sampled volumetric data, as shown in Figure 3. The output image size for our data set is 360×720 ($SweepAngle \times NumberOfSlices$). It is in RGB format with a depth of 1. Our tests were conducted using CT images from the National Library of Medicine's Visible Man data set [1]. The input size of each slice in our case is $512 \times 512 \times 12$ in RAW format.

Each CT slice has a header file with a defined constant, called the *Hounsfield number offset* [3], the image size in pixels, and a pixel thickness for each pixel in the image. By using these, we apply the formula:

$$\mu_i = \left[\frac{(I_i - 1024)}{1024} * \mu_w \right] + \mu_w \tag{1}$$

$$\frac{E_f}{E_i} = \exp\left[-\sum_{i=1}^{n} (\mu_i * t_{slice}) \right] \tag{2}$$

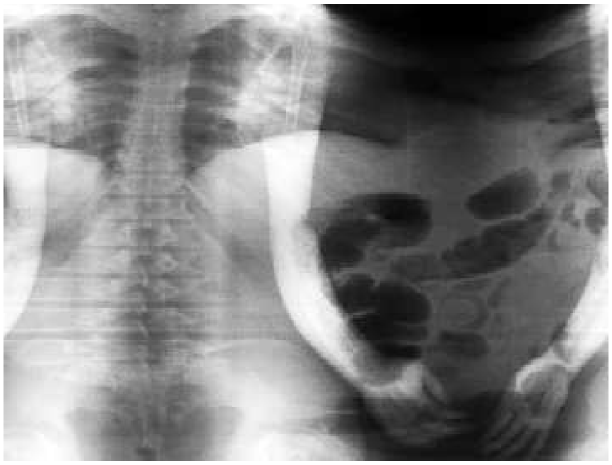

Figure 3: The 2D unwrapped body image

where, μ_i is the attenuation number, μ_w is the attenuation number of water (we use 0.02/mm), I_i is the intensity of the pixel, t_{slice} is the thickness of a pixel for the current slice, and E_f/E_i is the output/input energy, which constitutes the overall attenuation factor. The $(I_i - 1024)$ in Equation (1) is the Hounsfield number. Since this is an exponential function, the final projection pixel value can be computed by multiplying each term.

After the image is created, we wrap (texture map) this 2D image onto the surface of a cylinder with the proper transparency value. We choose the size of the cylinder to be sufficient to provide a parallax effect when rotated about the blood vessel. In other words, we can see through the front side of the body, represented by the textured cylinder, to the blood vessel inside the body, which partially occludes the rear side of the body.

As shown in Figure 4, we are approximating a ray cast through the volume with sample points on the surface of the cylinder. We approximate the value at point A by sampling the values along $\overrightarrow{OA}$, and at point B by sampling along $\overrightarrow{OB}$. A correct orthographic computation would sample the values along $\overrightarrow{AB}$. For $\theta = 180°$, the sample points accurately represent the data along the cast rays, as these rays are normal to the projection plane. The amount of distortion of the data increases as θ diverges from $180°$ in either direction. The impact of this distortion is minimized because the surgeon's attention is focused on the vasculature where θ will be close to $180°$.

5 Results

Once embedded in the body map, the resulting combination of pseudo-volumetric cues and the vasculature produces a rendered image very similar to that seen on the operating-room fluoroscope by the surgeon. Figure 5 shows one frame taken from an actual surgical procedure (patient's head). Figure 6 shows a snapshot of our visualization (patient's thoracic region with descending aorta), using the X-Ray Casting technique.

The fact that our approach uses texture maps to approximate volumetric data means that any hardware support for texture mapping provides a significant performance boost. We are testing on a 250MHZ R4400 SGI $Indigo^2$ Maximum Impact with 64MB main memory and 1MB texture memory. The pre-processing time takes approximately 93 seconds for $512 \times 512 \times 720$ data. We have achieved interactive rendering speeds of 16 frames/second for a scene with 12,264 triangles.

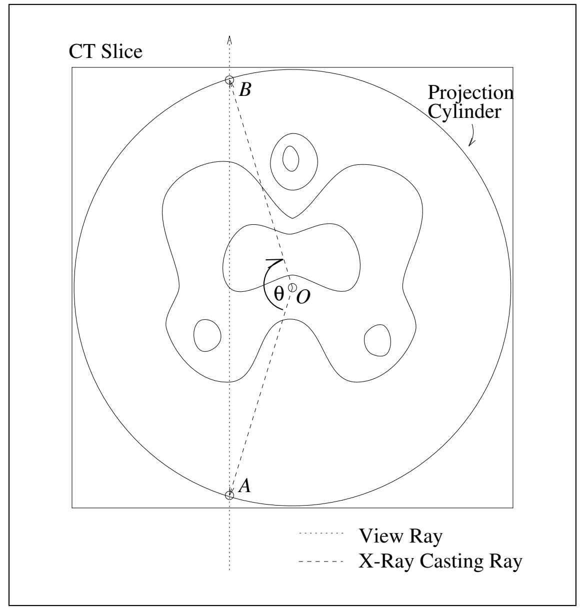

Figure 4: CT slice with approximation rays

The X-Ray Casting software used to create the body map is written in C++, and outputs a texture map. We are using Open Inventor [14] for rendering the environment.

6 Conclusion

A method for approximating volumetric medical data for surgical simulation was presented. The approach takes advantage of available texture memory, and provides realism approaching the actual display used in the operating room. We have shown that it is possible to achieve interactive frame rates using a limited amount of texture memory. This makes our approach attractive for systems using low-cost graphics workstations.

This is a simple approximation of 3D volume rendering. To get a better image, we are experimenting with putting multiple concentric cylinders inside the volume. This should give us more accurate results, because as the number of cylinders increases, distortion decreases, and we approach a full volumetric representation.

We also plan to implement the rectangular projection technique mentioned above, in order to compare the results to our cylindrical approach. This will require more disk space, will add to the amount of texture memory required to represent the whole environment, and increase the time required to change from one view to another, as a new texture map must be applied each time the view direction changes, but might provide more accurate results.

In addition, we are also examining the use of a spherical projection for visualizing CT head data. In this part of the surgical procedure, there is a different set of degrees of freedom for camera movement, so a cylindrical projection approach cannot capture all the necessary views of the volume data.

Further work will help us determine the extent to which our approximations and assumptions are valid within this domain, as well as shed some light on the applicability of our approach to more general visualization applications.

7 Acknowledgement

We would like to thank Dr. Murray Loew for his generous assistance with the image processing aspects of our work. We also thank the other members of The Laboratory for Advanced Computer Applications in Medicine for their support, specially for Rob Page who gave us many useful information on Section 2.

References

[1] Information on the visible human project can be found at http://www.nlm.nih.gov/research/visible/visible_human.html.

[2] B. Cabral, N. Cam, and J. Foran. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In *Proceedings 1994 Symposium on Volume Visualization*. IEEE Computer Society Press, October 1994.

[3] Z.H. Cho, J.P. Jones, and M. Singh. *Fundamentals of Medical Imaging*. Wiley, 1993.

[4] T.J. Cullip and U. Newman. Accelerating volume reconstruction with 3d texture hardware. Technical Report TR93-027, University of North Carolina, Department of Computer Science, Chapel Hill, NC., 1993.

[5] T.T. Elvins. A survey of algorithms for volume visualization. *Computer Graphics*, 26(3):194–201, August 1992.

[6] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practices (2nd Edition)*. Addison Wesley, 1990.

[7] R Fraser. Interactive volume rendering using advanced graphics architectures at http://www.sgi.com/technology/volume/volumerendering.html. 1995.

[8] S.Y. Guan and R. Lipes. Innovative volume rendering using 3d texture mapping. In *Proceedings Medical Imaging 1994–Image Capture, Formatting, and Display*, volume 2164, pages 382–392. SPIE, February 1994.

[9] B.M. Hemminger, T.J. Cullip, and M.J. North. Interactive visualization of 3d medical image data. Technical Report TR94-032, University of North Carolina, Department of Computer Science, Chapel Hill, NC., 1994.

[10] P. Lacroute. Real-time volume rendering on shared memory multiprocessors using the shear-warp factorization. In *1995 Parallel Rendering Symposiun*, pages 15–22. ACM SIGGRAPH, October 1995.

[11] M. Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, pages 29–37, May 1988.

[12] W.E. Lorensen and H.E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Computer Graphics*, volume 21, pages 163–170. ACM SIGGRAPH, July 1987.

[13] J.P. Singh, A. Gupta, and M Levoy. Parallel visualization algorithns: Performance and architectural implications. *IEEE Computer*, 7(27):45–55, July 1994.

[14] Josie Wernecke. *The Inventor Mentor*. Addison Wesley, 1994.

[15] L. Westover. Footprint evaluation for volume rendering. In *Computer Graphics*, volume 24, pages 367–376. ACM SIGGRAPH, August 1990.

[16] O. Wilson, A.V. Gelder, and J. Wilhelms. Direct volume rendering via 3d textures. Technical Report UCSC-CRL-94-19, University of California, Santa Cruz, Department of Computer and Information Science, Santa Cruz, CA, 1994.

Figure 5: A frame taken from the actual surgical video

Figure 6: A snapshot of our visualization