AN EFFICIENT METHOD OF VOLUME RENDERING FOR MEDICAL SLICES

Wanqing Li † *Hong Xie* ‡ *Yianni Attikiouzel* †

† CIIPS, Dept of Electrical & Electronic Engineering,
The University of Western Australia, WA 6009, AUSTRALIA
wql@ee.uwa.edu.au

‡ Computer Science Department, Murdoch University,
Murdoch, WA 6150, AUSTRALIA
xie@cs.murdoch.edu.au

ABSTRACT

We present an efficient method for volume rendering. This method uses a special light source – slit-light source to illuminate the volume and simulates the interactions of light with objects, such as light absorption, scattering and reflection.

In comparison with previous methods, our method has several advantages. This includes flexibility for implementation, efficiency for computation, high reality for display and easiness of parallelization. The algorithm can be parallelized easily either using the Data Space Subdivision method or the Image Space Subdivision method.

1. INTRODUCTION

Volume visualization [1, 2, 3, 4] provides direct approaches to display 3-D information in 3-D digital scenes, or volumes. Two main categories of techniques are commonly used in volume visualization: surface rendering and volume rendering [1, 2, 3, 4].

Surface rendering is used for visualizing the shapes of 3D objects and the spatial relationships among these objects. In this method, the surfaces of objects are first defined and modelled by polygons or other geometric primitives through segmentation or surface detection. These geometrical primitives are then rendered using the conventional computer graphics techniques for geometrical objects. This method tends to lose valuable information about objects when applied to many natural objects, such as trees and human tissues, which are not easily modelled by simple geometric primitives.

In contrast to surface rendering, volume rendering transforms the density information of a volume to the gray or color intensity in the 2-D viewing space directly without detecting, formatting or modelling object surfaces. It avoids the difficult segmentation process and the possible artifacts that can be introduced by the segmentation process. It provides a better mechanism for displaying weak or fuzzy surfaces and internal structures.

In this paper, we will discuss some of the major issues with the volume rendering technique first. This is followed by a brief review of the previous work in this area. Then we will present our new method for volume rendering which is aimed at solving some of the major problems that have been discussed. Finally we describe some experimental results of applying our method to several sets of MRI and CT slices of human heads, which show the effectiveness of our method.

2. PROBLEMS OF VOLUME RENDERING

In general, A volume rendering system [2, 3, 4] can be divided into three components, i.e., preprocessing, illumination modeling and a ray tracing algorithm. Preprecessing converts the original density into device and application independent values, such as the absorption coefficiency, color and opacity. Illumination modeling defines the geometry of the model and interactions between the light and objects. Ray tracing algorithm traces rays into the volume and calculates their contributions to the viewing image according to the defined model.

There are three main problems in volume rendering: huge data volume, intensive computation, and requirement of high display quality

Previous efforts to deal with these problems, especially the first two, include the use of simple illumination models with parallel light sources, the incorporation of optimization strategies into ray tracing al-

0-8186-6950-0/94 $4.00

gorithms [5], the use of specialized architectures, like Fuchs/Poulton Pixel-Planes Machine, Kaufman's Cube Architecture, Pixar/Vicom II, Reynolds and Goldwasser's Voxel Processor Architecture [1, 6].

Recently, parallelization of classical volume rendering algorithms and their implementation on general purpose SIMD and MIMD machines, and on distributed systems or networked computer systems show some prospect of performance improvements [7]. However, such improvements depend heavyly on the amount of parallelism inherent in the original rendering algorithms. Unfortunately, the parallelization was not a major consideration when those classical volume rendering methods were first proposed.

As to the requirement of high display quality, it is often desired, in many applications, that the image should provide details of objects with high reality, e.g., shadow, shading and depth cue. In surface rendering, objects are well defined and modelled. Highly qualitative display can be normally obtained. In volume rendering, however, objects are implied in the volume data set. There is no underlying mathematical models that can be used to describe them. Consequently, the quality of display mainly relies on the illumination model used for rendering.

A number of illumination models were proposed in the past [8]. Most of them are solid object surface based illumination models. They do not consider the emission and absorption of light, which are very important to visualize volume data set.

In 1986, Kijiya [9] proposed a global illumination model without limitations, restrictions or simplifying assumption. Unfortunately, no analytic mathematical solution exists for his illumination equation. Kajiya suggested to solve the equation using Monte Carlo method. Later, several simplified models were proposed based on the Kajiya model. Some of the typical examples are Meinzer's Heidelberg Ray Tracing Model [4], Blinn's Light Reflection Function [10], and Drebin's and Levoy's Ray Casting model [2, 3].

Blinn's model requires that objects consist of homogeneous material. It used one parallel light source and assumed the low albedo case. In Drebin's and Levoy's ray casting model, there were no emission and absorption components, hence no real shadow. In Heidelberg model, although the emission and absorption of light were simulated for shadow and shading, the display lacked of perspective effect and depth cues because of using parallel light and also extra-computation was needed for the second light source.

In this paper, we have developed an efficient method of volume rendering based on the Heidelberg ray tracing model but using one slit-light source. It incorporates provisions for parallelization and it minimizes the data swap between the main memory and disks. The algorithm is flexible enough to be implemented on micro-computers, general workstations, parallel machines, and networked computers. Following sections describe our methods in details.

3. RAY TRACING MODEL

We assume that the original volume is already isotropic. Otherwise, interpolation may be applied to convert the data set into an isotropic one. Figure 1 gives an outline of the processes involved in our ray tracing model. It consists of a preprocessing component, a component for modeling illumination, and a ray tracing algorithm associated with the illumination model.

3.1. Preprocessing

Before rendering is taking place, the volume data values must be converted into a predefined uniform format. This is done in the preprocessing component. Of special consideration here is the transformation of modality-dependent values into the standard ones. This is accomplished by a linear transformation of voxel density into the range between 0 and 1.

Other preprocessing, like adjustment or enhancement of the density values of certain image regions, selection of interesting objects, could also be carried out in this component.

3.2. Illumination Model

It is composed of two parts. One is its geometry, which defines the type of light source, the geometric relationships among the light source, the volume, and the observer. The other is the interactions between the light and objects in volumes.

For the former, we use the slit-light source located vertical to the slice plane, as shown in Figure 2. The rendering of each slice is projected to one row on the display image. This setting brings us a number of benefits. With this model, it is obvious that the volume data can be loaded and rendered slice by slice. The model also implies the rendering process can be parallelized by data space subdivision (DSS) or by image space subdivision (ISS).

For the latter, Kajiya [9] provided a global description of the interactions between light and objects. His rendering equation states that the transport intensity of light from one surface point to another is simply the sum of the emitted light and the total light intensity which is scattered towards the second point from all other surface points. In his equation, each surface

point is thought as a new light source, the reflection and refraction of light are represented by a "geometry" term.

As there is no analytical or mathematical solution to the equation, it must be simplified before it can be implemented on computers. In our model, several simplifying assumptions have been made as did in [4]. One important assumption is that the various interactions of the light with objects are independent of each other. Other simplifications include the assumption of low albedo case, the omitting of the reflected and scattered portions in calculating the absorption, and the assumption that there is no refraction existed when light passing through the surfaces of different objects. Thus, our model simulates the following interactions, as shown in Figure 3.

- *Absorption of light*: when light passing through a voxel, it will be attenuated by a certain percentage. The percentage is defined as a function of the voxel's density, as shown in Figure 3 (a) and (b).

$$I_{abs} = f_a(d)I_{in} \quad (1)$$
$$I_{out} = (1 - f_a(d))I_{in} \quad (2)$$

  where, d is the voxel density after preprocessing, I_{in}, I_{out}, and I_{abs} are the intensities of the incident light, the transmitted light and the absorbed light respectively. $f_a(d)$ models the absorption of light, called absorption function, with value range of $[0, 1]$. It provides us ways of setting different absorption nonlinearly for different tissues. This setting will produce a variety of visual effects, such as transparency and boundaries between different types of tissues on the display.

  Absorption of light in our model produces real shadows on the final display.

- *Scattering*: when light hits a voxel, a portion of it will be scattered towards the observer. We assume that the light is scattered uniformly in all directions and the intensity of the scattered light is proportional to the intensity of the incident light at the voxel and to the function of the voxel density d, i.e.,

$$I_{scat} = I_{in} f_s(d), \quad (3)$$

  where $f_s(d)$ is a scattering function with value range of $[0, 1]$.

- *Reflection*: analogous to scattering, when light hits a voxel which is just on an object surface, some of it will be reflected to the observer. In contrast to scattering, the intensity of the reflected light depends on the position of the observer, or the viewing direction. This interaction give us a real shading on the display.

  Phong shading is used to model the reflection of the light. Just as scattering, the portion of the light reflected to the observer is proportional to the incident light.

$$I_{ref} = I_a + I_{dif} + I_{spe} \quad (4)$$
$$I_{dif} = I_{in}(\vec{n} \cdot \vec{L}) \quad (5)$$
$$I_{spe} = I_{in}((2(\vec{n} \cdot \vec{L})\vec{n} - \vec{L}) \cdot \vec{V})^p \quad (6)$$
$$I_a = 0 \quad (7)$$

  where I_{ref} is the reflected intensity, I_a is the ambient intensity, I_{dif} is the diffusely reflected intensity, I_{spe} is the specularly reflected intensity, and I_{in} is incident intensity. $\vec{n}$ is the normalized vector of the object's surface, $\vec{L}$ is the direction of the incident light, $\vec{V}$ is the viewing direction, p is the exponent that varies with the glossiness of the surface.

  To avoid an extra description or segmentation of the volume, we estimate the object's surface normal based on the intensities of neighbouring voxels, i.e., the intensity gradient is used as the normal of the surface at a given point [11].

3.3. Ray Tracing Algorithm

The ray tracing algorithm specifies the way of calculating the light reflected, scattered or emitted from each voxel to the observer. It could be designed either from the ray-tracing's point of view (i.e., in the image space), or from the voxel-traversal's point of view (i.e., in the object space). In the former, each ray's path from the light source to the volume and then to the observer is considered as a basic element of the algorithm. In the latter, light interactions at each voxel become the basic element of the algorithm. Our algorithm is based on the latter using a FTB (Front To Back) traversal method mostly (as shown in Figure 3 (a)). Each voxel is traversed in the algorithm only once.

Because of our specific illumination geometry, ray-tracing process for each slice is identical. The result of the ray-tracing for each slice corresponds to a row in the displayed image. The aggregate of the results for all slices forms a view of the volume.

In general, all rays into the volume are traced row by row in FTB. For each row, the intensity of each incident ray, and the contribution of the ray to the

final image row are recorded. In order to avoid unnecessary repeated computation of the intensity of each incident ray and the attenuation of the scattered and reflected light in each row, two array data structures are used to store the intermediate results of the computation: $ABSORP\_PATH1[i]$ (where i means the ith ray into the volume) keeps the absorption of the incident light up to the current ray-tracing row, and $ABSORP\_PATH2[j]$ (where j means the jth pixel on the image row) keeps the reflected and scattered light up to the current ray-tracing row.

According to the observation that once a ray has gone into the volume for a sufficient distance, it is attenuated quickly and its further contribution to the final image can be ignored, an adaptive termination of ray-tracing could be introduced into the above algorithm. Also obviously, the use of look-up tables will speed up the computation.

4. IMPLEMENTATION AND DISCUSSION

4.1. Implementation

We have implemented the rendering method using C++ and XView on Sun workstations. The rendering results on both X-ray CT and MRI slices have proven our initial expectation on the model.

Figure 4 shows two different views of both the soft-tissues and the bones of a human head from a 128x128x113 X-ray CT data set, where $f_a(d) = d$ and $f_s(d) =$ *a constant* are used. Figure 5 is the rendering result on a 128x128x109 MRI data set with $f_a(d) = d$ and $f_s(d) =$ *a constant*.

4.2. Discussion

With an aim to tackling some of the major problems currently existed in volume rendering, we proposed a new illumination model for volume rendering. In comparison with previous methods, the new method has several advantages. It is more flexible – the model can be easily implemented on various computer systems. It is very efficient – no data swapping is needed, voxel traversal was minimized to only once for each voxel, and adaptive termination of ray-tracing can be easily incorporated. It is also of higher reality – as our preliminary results show, the method produces real shadows and shadings. Finally, the method can be implemented on parallel computer systems to maximally take advantage of the multiple processors available – the algorithm can be easily parallelized either by the Data Space Subdivision or by the Image Space Subdivision method.

The main disadvantage of our model is the limitation of the viewing directions. We limit the viewing directions to those perpendicular to Z-axis (as shown in Figure 2). Views from other directions may be obtained by rotating the volume first and applying our method to the rotated volume.

5. REFERENCES

[1] M.R. Stytz, G. Frieder, O. Frieder, *Three-Dimensional Medical Imaging: Algorithms and Computer Systems,* ACM Computing Surveys, Vol.23, No.4, December 1991, pp.421-99

[2] Robert A. Drebin, Loren Carpenter, Pat Hanrahan, *Volume Rendering,* Computer Graphics (SIGGRAPH'88), Vol.22, No.4, Aug. 1988, pp.65-74

[3] Marc Levoy, *Display of Surfaces from Volume Data,* IEEE Computer Graphics and Appl cations, May 1988, pp.29-37

[4] Hans-Peter Meinzer, Kirsten Meetz, Dinu Scheppelmann, *The Heidelberg Ray Tracing Model,* IEEE Computer Graphics and Applications, Nov. 1991, pp.34-43

[5] Marc Levoy, *Efficient Ray Tracing of Volume Data,* ACM Trans Computer Graphics, Vol.9, No.3, July 1990, pp.245-61

[6] A. Kaufman, R. Bakalash, D. Cohen and R. Yagel, *A Survey of Architectures for Volume Rendering,* IEEE Engineering in Medicine and Biology, Vol.9, No.4, Dec 1990, pp.18-23

[7] Kwan-Liu Ma and James Painter, *Parallel Volume Visualization on Workstations,* Computer & Graphics, Vol.17, No.1, 1993, pp.31-37

[8] Donald P. Greenberg, *Light Reflection Models for Computer Graphics,* Science, Vol.244, No.4, April 1989, pp.166-73

[9] James T. Kajiya, *The Rendering Equation,* Computer Graphics (SIGGRAPH'86), Vol.20, No.4, 1986, pp.143-49

[10] J.F. Blinn, *Light Reflection for Simulation of Clounds and Dusty Surfaces,* Computer Graphics (SIGGRAPH'82), Vol.16, No.3, July 1982, pp.21-29

[11] A. Pommert, U. ticdc, G. Wiebecke, K. H Hohne, *Surface Shading in Tomographic Volume Visualization: A Comparative Study,* Proceedings of the first conference on visualization in biomedical computing, IEEE Computer Soc. Press, Los Alamitos, CA, USA, pp.19-26

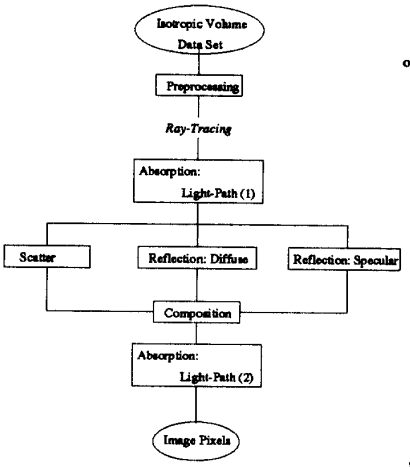


Figure 1. ***Overview of the Ray Tracing Model***

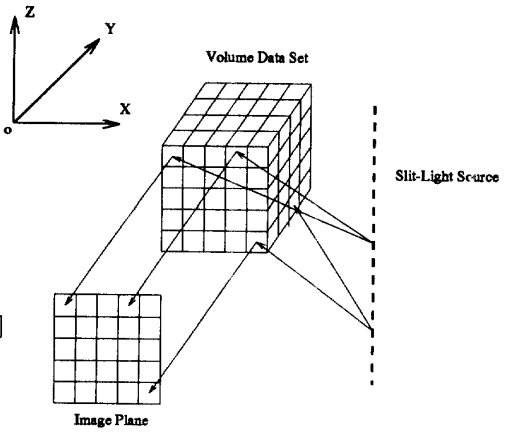
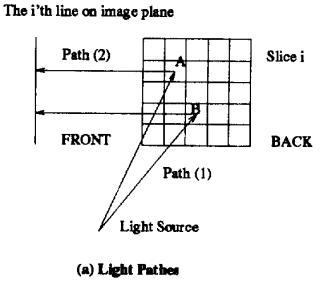
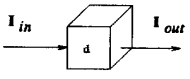


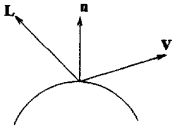
Figure 2. Geometry of the Illumination Model



(a) Light Pathes



(b) Absorption of Light



(c) Reflection

Figure 3. ***Interaction between light and substances (Rays passing through a voxel)***

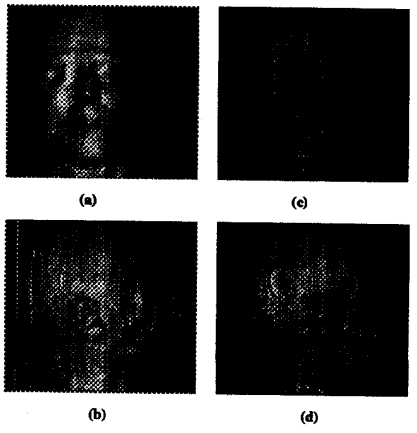


Figure 4. ***Rendering results on 128x128x113 X-Ray CT slices. (a), (b) Soft-tissue; (c), (d) Bones***

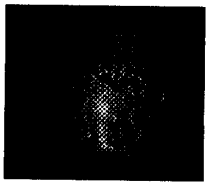


Figure 5. ***Rendering image on 128x128x109 MRI slices***