Texture Planes: Real-Time Volume Rendering Using Hardware Texture Mapping and Alpha Blending

Issac J. Trotts*
trotts@cs.ucdavis.edu
Deparment of Computer Science
University of California at Davis

Bernd Hamann†
hamann@cs.ucdavis.edu
Deparment of Computer Science
University of California at Davis

1 Algorithm Description

We present an efficient algorithm, called *Texture Planes*, for directly rendering 3D volumetric data on workstations with hardware for alpha blending (transparency) and 2D texture mapping. Our implementation runs about two orders of magnitude faster than a naive conventional volume renderer and generates images of comparable quality.

Conventional volume rendering techniques proceed by casting a ray through each pixel of the display, integrating several lighting calculations along each ray to obtain the color for the pixel. Our algorithm approximates this computation by rendering a stack of rectangles which slice through the volume. We apply a texture map to each of the rectangles based on the scalar values it touches in the volume. We assign each texel of the texture a (possibly) different color and transparency. We use the density of the volume at the location of of texel to find the its lighting properties in a look-up table and use the gradient to determine the opacity as well as the normal for the lighting computation.

To ensure the quality of the renderings from all viewing directions, we compute a stack of slices for each of the x, y, and z axes. To render, we select the stack whose slices are most nearly perpendicular to the vector from the camera point to the center of the volume.

2 Discussion

2.1 Hybrid Images with Volume Rendering and Z-Buffer Rendering

For some applications it is useful to render scenes containing both volumetric data and polygonal geometry. Levoy describes a way to do this with a conventional volume renderer in [Lev88], either by ray tracing the polygons or by sampling the polygons to incorporate them into the volume. However, ray tracing is usually a time-consuming process and sampling the polygons is likely to introduce spurious bumps on their surfaces (from voxel aliasing). Sampling the polygons into the volume might also complicate the classification of tissue types.

In contrast, volume renderings produced with our method can easily and accurately be displayed in a scene with polygon-based geometry, using a standard 3D graphics library such as *Open GL* or *Open Inventor* . Polygonal models passing through the volume are rendered correctly.

*Web: http://graphics.cs.ucdavis.edu/~trotts

†Web: http://graphics.cs.ucdavis.edu/people/hamann

‡This work was supported by various grants and contracts awarded to the University of California, Davis, including the National Science Foundation under contract ACI 9624034 (CAREER Award), the Office of Naval Research under contract N00014-97-1-0222, the Army Research Office under contract ARO 36598-MA-RIP, the NASA Ames Research Center under the NRA2-36832(TLL) Program, and the Lawrence Livermore National Laboratory under contract W-7405-ENG-48 (B335358). We thank all members of the Visualization Thrust at the Center for Image Processing and Integrated Computing (CIPIC) at the University of California, Davis, for their help.

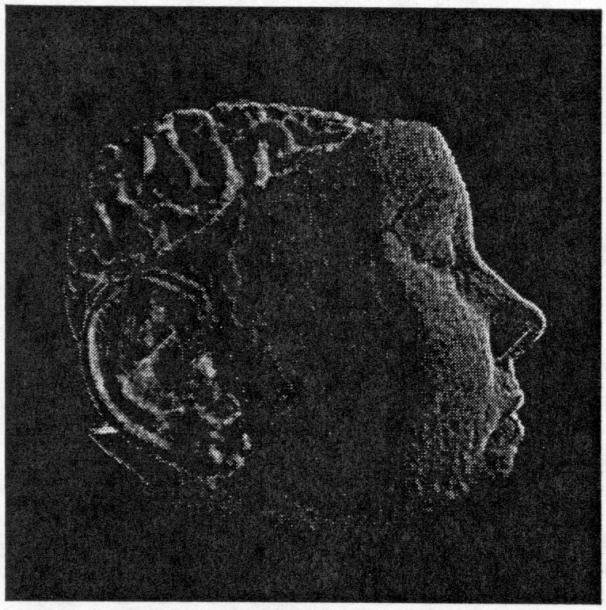

Figure 1: A rendering generated by *Texture Planes* in 1.136 seconds on an SGI Onyx 2 with 512 MB RAM, an Infinite Reality graphics engine and four R10000 processors. Only one processor was used. The computation of the slices and generation of an Open GL display list took 120 seconds.

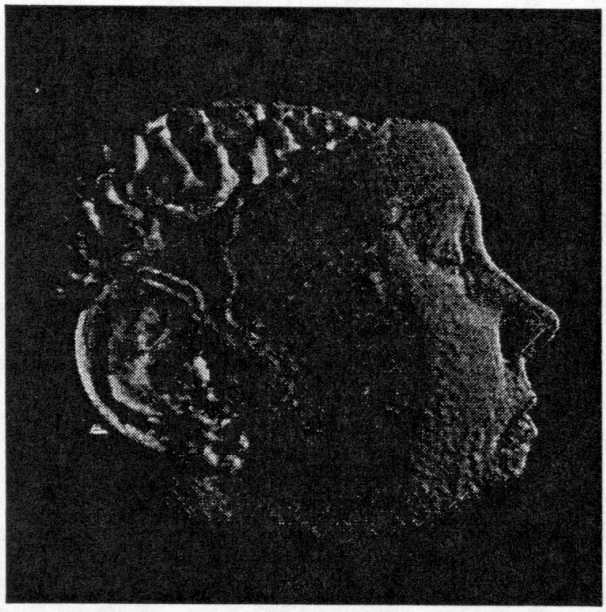

Figure 2: A rendering generated with a naive conventional volume renderer in 302 seconds on the same Onyx as in Figure 1.

2.2 VRML Implementation

Another advantage of our technique over conventional volume rendering is that it can be easily be implemented in the Virtual Reality Modeling Language, which makes possible the publication of interactive volume renderings to the web. It should also be able to seamlessly integrate these volume renderings into existing VRML worlds without any additional coding.

2.3 Non-Cartesian Grids

Our implementation currently handles only Cartesian grids (rectangular volumes with rectangular voxels). However, our algorithm could also be used for interactive rendering of volumes containing tetrahedra, hexahedra, and other voxel types. We would do this by resampling the irregular grid into a Cartesian grid, and then applying the Texture Planes algorithm as usual.

3 Conclusion

We have presented a simple algorithm that exploits hardware texture mapping and alpha blending to produce renderings of scalar fields at interactive frame rates in scenes with arbitrary polygon-based geometry. Our algorithm generates images of quality nearly equal to that of conventional volume rendering in a small fraction of the time. An investigation of methods for reducing the time spent preparing slices is slated for future research.

References

[Lev88] Marc Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, pages 29–37, July 1988.

[Max95] Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, pages 99–108, June 1995.