Interactive Splatting of Nonrectilinear Volumes

Peter L. Williams
(williams@csrd.uiuc.edu)
Center for Supercomputing Research & Development
and
National Center for Supercomputing Applications
University of Illinois at Urbana

Abstract

This paper describes various techniques for achieving interactive direct volume rendering of nonrectilinear data sets using fast projection (splatting) methods. The use of graphics hardware, rendering approximations, parallelization and reduced resolution meshes are discussed. Results from the use of these techniques are presented in the form of color photos and comparative timings.

1 Introduction

Volume rendering (also known as volumetric rendering) is used to show the characteristics of the interior of a solid region in a 2D image [3]. Volume data is typically defined over a mesh. Meshes can be classified by the structure of their cells as rectilinear, curvilinear or irregular (unstructured). These terms are defined in [7, 25, 26]. Curvilinear and irregular meshes are also referred to as nonrectilinear meshes.

A good deal of attention has been given to volume rendering rectilinear data sets [2, 4, 9, 10, 11, 12, 13, 16, 17, 19, 20, 23]. However, much less has been published dealing with nonrectilinear data [1, 5, 6, 13, 17, 18, 22, 24]. Nonrectilinear data can be rendered volumetrically by first interpolating it to a rectilinear mesh; efficient methods for doing this are being investigated by Wilhelms et al [22]. The focus of this paper is on volume rendering nonrectilinear data without interpolating it to a rectilinear mesh.

There are three basic techniques for displaying volume data [12], the use of 2D cross sections or slices of the data, (cross section rendering); 3D level surfaces (isosurfaces) or threshold rendering, which can either be opaque or semitransparent; or direct volume rendering where a 2D projection of a colored semitransparent 3D volume or cloud is displayed. This paper concentrates on direct volume rendering.

Direct volume rendering is a method for rendering 3D scalar fields by directly displaying the data without first extracting intermediate representations, such as isosurfaces. The volume of data as a whole is rendered. One way to do this is to display a 2D projection of a colored semitransparent 3D volume or cloud, where the color and opacity are functions of the scalar field. These functions can be used to highlight desired features in the data, such as the extrema or hot-spots, or other regions of interest such as shock waves. Thus a holistic view of the entire field can be given with the brightly colored extrema gleaming through the cloud. A feeling for the spatial orientation of the field and the relative locations of areas of interest is given by rotating the image, hence the importance of interactivity.

There are three classes of techniques for direct volume rendering: ray tracing or ray casting [1, 4, 9, 11, 12, 16, 19, 22], projection methods [2, 10, 13, 17, 19, 21, 20, 23, 24, 26], and hybrid methods [5, 6].

We focus on the use of projection methods for nonrectilinear data where the data is rendered without interpolating it to a rectilinear mesh [13, 17, 24, 26]. In these methods, each cell of the mesh is projected onto the screen in back-to-front order. To do this, an algorithm is required to visibility order the cells of the mesh. The cell's color and opacity contribution to each pixel is calculated and then blended with the pixel's existing color and opacity.

A very accurate, but computationally intensive, algorithm is given by Max, Haarrahan and Crawfis [13]. A fast approximation to this process, sometimes called a splatting algorithm, is given by Shirley and Tuchman.
A modified version of the Shirley and Tuchman Projected Tetrahedra (PT) splatting algorithm is used as the basis for the work herein. Wilhelms and Van Gelder [23] discuss a number of relevant issues.

The PT algorithm requires the cells to be visibility ordered. Algorithms and techniques for visibility ordering the cells of meshes of any shape and cell structure are given by Williams in [25]. The Meshed Polyhedra Visibility Ordering (MPVO) algorithms described in that paper are used herein.

We investigate techniques for achieving interactive performance with the PT splatting algorithm along with the MPVO algorithm, even for very large data sets. The techniques investigated are parallelization, graphics hardware support, rendering approximations and mesh filtration. Using the methods described herein, images of data sets with over 1,000,000 cells have been generated interactively (in less than 15–30 seconds). Using the filtering methods described herein, this performance is possible for even larger data sets.

In Section 2, we discuss our investigation of fast approximations to the PT splatting process. These approximations along with the PT algorithm and the highly accurate methods for direct volume rendering form a hierarchy of rendering methods that trade-off image accuracy/quality and generation time. We briefly describe the role of high performance graphics hardware and parallelization in achieving the goal of interactive volume rendering. In depth coverage of these topics is given in [26].

In Section 3, reduced resolution meshes are discussed. The goal is to reduce the number of cells rendered so the rendering time is improved with the minimum possible degradation in image value.

2 Rendering approximations

In this Section, we investigate a number of fast rendering approximations that support the goal of interactive volume rendering. The Shirley and Tuchman PT algorithm, which has been modified and optimized for a high performance graphics workstation, is used as the basis for these approximations and also as a standard of comparison, both for image information content and rendering speed.

When considering the validity of these approximated images, it is important to keep in mind that interactive volume rendering is intended to give the scientist a working tool to provide a general idea of the spatial distribution of the scalar field and roughly identify areas of interest, e.g. extrema or hot-spots, and not necessarily to create a highly realistic or precise image for publication purposes.

In other words, this environment is intended to be a data previewer for the scientist. He or she might use it in an analogous way to a professional photographer who uses a Polaroid camera to quickly check several different shots before using a 4x5 view camera for the final photo. These fast approximations can also be useful for setting initial viewing parameters.

There is a high degree of abstraction inherent in direct volume rendering using projection methods; every cell in the mesh contributes to the image. Therefore approximation errors if randomly distributed over the image may not be significant. However, if the mesh is highly regular, then these errors can be magnified and be highly noticeable.

For this paper, only tetrahedral cells are considered. The MPVO algorithm can deal with any cells. However, the PT splatting algorithm deals only with tetrahedra. Wilhelms and Van Gelder [23] discuss the splatting of right-angled parallelepiped (bricks). Alternatively, nontetrahedral cells can be tetrahedralized so the PT algorithm can be used.

To highlight areas of interest in the scalar field $S(x, y, z)$ and de-emphasize other areas, user-specified color and density transfer functions $\kappa_c(S(x, y, z))$, $\kappa_d(S(x, y, z))$, $\kappa_s(S(x, y, z))$, and $\rho(S(x, y, z))$ map a scalar field value to a color $\kappa_c$, $\kappa_d$, $\kappa_s$ and density $\rho$. The density is used to calculate an opacity value.

We start with an overview of the splatting process, then describe Shirley and Tuchman's PT algorithm, and then describe our fast splatting approximation algorithms, and finally present comparative timings and discuss image quality.

2.1 Overview of the splatting process

In splatting, as in splatting a snowball against a wall [21], each cell is projected onto the screen in visibility order from back to front to build up a semitransparent image. The contribution of each cell to the image is proportional to the thickness of the splat.
is rendered as a set of up to four triangles which have a common vertex at the point of maximum thickness of the splat. At this common vertex, the opacity is nonzero; at all other vertices the opacity is zero. The opacity and color at the vertices of each triangle are interpolated over the splat. Wilhelms and Van Gelder [23] describe three possible interpolation methods.

2.2 Projected Tetrahedra algorithm

The PT algorithm works by compositing into an image the polygonal projection of each cell, called the cell's footprint, onto the viewing plane. Each footprint is subdivided into from one to four triangles, depending on the projection, and then rendered as a set of triangles.

For tetrahedra, there are only two combinatorially different projections; they are shown in Figure 1 as cases 1 and 2. Cases 3 and 4 are degenerate instances. My experiments show that, for irregular meshes, case 1 occurs approximately 40% of the time, on average, and case 2 approximately 60%. Cases 3 and 4 occur less than 5% of the time. In what follows, only cases 1 and 2 are discussed, the extension to the degenerate cases is obvious.

Each 2D footprint has a “nonzero thickness” (NZT) vertex as shown in Figure 1. For each cell, the opacity at the NZT vertex is computed as described below. The opacity at the remaining vertices is set to zero. The color and density is calculated using Equations 1 and 2 as explained below.

Using the optical model for a volume density described in [17] or [27], the PT method assumes the density and color are constant along a ray through a cell. This leads to Equations 1, 2, and 3, where \( t_1 \) and \( t_2 \) are the cell entry and exit points of a ray through the NZT vertex. \( S \) is the scalar field. \( I_{\text{red}}(t) \) is the cumulative intensity of red light at point \( t \) on a ray traveling towards the eye. \( I_{\text{red}}(t_1) \) is the cumulative result of a cell by cell ray integration up to the point \( t_1 \). Similar equations apply for the green and blue channels.

\[
\kappa_{\text{avg}} = \frac{\kappa_r(S(t_2)) + \kappa_r(S(t_1))}{2} \tag{1}
\]

\[
\rho_{\text{avg}} = \frac{\rho(S(t_2)) + \rho(S(t_1))}{2} \tag{2}
\]

\[
I_{\text{red}}(t_2) = \kappa_{\text{avg}}(1 - e^{-\rho_{\text{avg}}(t_2-t_1)}) + I_{\text{red}}(t_1)e^{-\rho_{\text{avg}}(t_2-t_1)} \tag{3}
\]

The opacity \( \alpha \) is then: \( \alpha = 1 - e^{-\rho_{\text{avg}}(t_2-t_1)} \)

As suggested in [13, 17, 24], the projection of a polyhedron can be approximated by a collection of Gouraud-shaded polygons. The graphics hardware linearly interpolates the color and opacity specified at each vertex over the interior of the polygon. Also, the hardware is used to blend the new and old color for each pixel. The remainder of the splatting process, cell visibility ordering, decomposition of the polyhedral cell into polygons, and calculation of color and opacity for each polygonal vertex is done in software.

Figure 2 and Color Figure 1 show volumetrically rendered images using the modified PT algorithm on a Silicon Graphics 4D/360VGX workstation. The MPVO algorithm for nonconvex meshes was used for visibility ordering. The image in Figure 2 is of a simulated temperature field defined on a nonconvex irregular mesh of 15,499 tetrahedra comprising a MBB-Gehäuse solid modeling benchmark. The mesh was generated by a conformed Delaunay triangulation [14, 15]. Color Figure 1 is a density field defined on a curvilinear mesh of 187,395 tetrahedra from a simulation of a blunt-fin induced shock wave and turbulent boundary layer separation [8].

The PT and MPVO algorithms and the rendering approximations were parallelized for a MIMD architecture with graphics hardware. All timings reported herein, both serial and parallel, are from a Silicon Graphics 4D/360VGX graphics workstation. The parallelization of the splatting phase is straightforward. Each CPU takes a cell and splats it in parallel.

(See color plates, p. CP-5.)
A spin-lock provides mutually exclusive access to the graphics hardware when a graphics call is made. The parallelization of the visibility ordering algorithms is discussed by Williams in [26].

2.3 The rendering approximations

To determine the feasibility of even faster methods than the PT algorithm, several rendering approximation methods were investigated. Each method involves a trade-off of image accuracy/quality for faster generation time.

2.3.1 Voxel approximation

The voxel (VOX) approximation precomputes an average color and opacity for each cell based on the average of the scalar data at the four vertices of the cell. This value is stored as a single packed 32-bit integer in the MPVO data structure along with the cell's vertex and adjacency information. The color $\kappa_c$ for a cell $c$ is calculated as:

$$\kappa_c = \kappa_c(\text{avgScalarData}(c))$$

where $\alpha$ is an empirically determined constant called the attenuation factor. Typically, $0.05 \leq \alpha \leq 5.0$ for the VOX method. The average scalar data is calculated by averaging the data values at the four vertices of the cell.

The attenuation factor $\alpha$ for a given data set is calculated by adjusting $\alpha$ until the approximated image matches as closely as possible the corresponding image produced by the PT method. The attenuation factor need be calculated only once in a preprocessing step.

For case 1, only one triangle is output, the single front-facing or back-facing triangle, for case 2, the two front-facing triangles. Each triangle is rendered with a single color and opacity, the precomputed average value for the cell described above.

Images generated using the VOX approximation are shown in Figure 3 and Color Figure 3; attenuation factors of 2.25 and 0.1, respectively, were used.

(See color plates, p. CP-5.)
2.3.2 Uniform Thickness Slab approximation

The next approximation, the uniform thickness slab (UTS1) approximation, treats each cell as a slab of uniform thickness. The opacity for each vertex $v$ is calculated as $a = a_p(S(v))$, where $a$ is an empirically determined constant. Interesting and useful visual effects can be achieved by varying $a$ between 0.1 and 2.5. See Figures 4 and 5 where $a$ is 1.8 and 0.4, respectively. The color for each vertex is simply $c_r(S(v))$, $c_g(S(v))$, and $c_b(S(v))$. The vertex colors and opacities are calculated in a preprocessing step and stored as a single packed 32-bit integer for each vertex.

Rather than perform a case analysis to determine the splat profile, all front-facing faces are rendered. This technique is fast since the MPVO data structure contains information about which faces are front-facing.

A variation of this method, called UTS2, is to render the single front-facing or back-facing face for case 1 projections. This requires extra calculation to determine the footprint; but it results in about 25% fewer polygons being rendered.

Images generated with the UTS1 method are shown in Figures 4 and 5 and Color Figure 2. An image from the UTS2 method is shown in Color Figure 4. An attenuation factor of 0.15 was used for both color images. The UTS1 method was somewhat faster than the UTS2 method for serial use. Both UTS methods were twice as fast as the PT method in serial mode.

2.3.3 Wedge approximation

The final method considered here, the wedge (WED) approximation, decomposes each splat into triangles as shown in Figure 1. Initially, the centroid was used as the NZT vertex for case 2 because it was faster to calculate than the NZT vertex used by the PT method. However, the image quality was not good; therefore the NZT vertex, as calculated by the PT method, had to be used.

The color and opacity for each vertex is calculated in the same way as in the UTS1 method. For case 1, the color and opacity for the NZT vertex is just the color and opacity for that vertex. For case 2, the color and opacity for the NZT vertex is the average of the color and opacity at the four cell vertices. The opacity is set to zero at the remaining vertices of the footprint. The color and opacity for each vertex is calculated and stored in a preprocessing step. A typical value for the attenuation factor $a$ used in the WED method is 0.6.

The WED approximation was not significantly faster than the PT algorithm; and its image quality was noticeably inferior. Therefore this approximation method was not considered useful.

2.3.4 Summary

Table 1 shows comparative timings for these five methods, for 71,680 and for 593,920 tetrahedra, both for serial and for parallel execution using 6 CPUS.

The execution time for the PT splatting method, for data sets up to 600,000 cells, was brought into the interactive range by the use of graphics hardware and parallelization. Through the use of the suite of approximations described above, in particular the UTS1 method, interactivity was achieved for serial rendering of data sets of over 1,000,000 cells. Using the UTS1 approximation, a data set of 1,003,520 cells was rendered serially in 34.0 seconds. That same data set was rendered in parallel using the VOX method in 16.3 seconds. In comparison, the time to render that data set using the PT method in serial was 79.6 seconds, and 30.2 seconds in parallel.

It was found that the size of the image generated had little effect on the time of generation. An image that filled the entire screen took less than 2% longer than one that was one inch square.

The UTS1 method was the fastest of the approximation methods for serial execution; and surprisingly, its images were the best approximation to the PT method's images, especially if the attenuation factor was carefully set. Little difference was noted between the images of the UTS1 and UTS2 methods.

(See color plates, p. CP-5.)
The VOX approximation also gave good images but usually the voxel nature of the approximation was clearly evident. The VOX approximation is more valid when the cells are relatively small in terms of pixel coverage. For parallel usage, this method is the fastest of all the approximations. An explanation of why the VOX method outperforms the other methods for parallel execution is given in [26].

3 Reduced resolution meshes

Even with parallelization, fast graphics hardware and the use of rendering approximations, for very large data sets, it still may not be possible to generate images interactively. For example the hardware rendering system may have an upper bound on the number of polygons per second that it can handle.

Therefore, to achieve interactivity, it may be necessary to reduce the number of cells rendered by techniques such as: filtering (rendering only selected cells), by coalescing adjacent cells, or by retriangulating a subset of the vertices of the mesh. I refer to this concept as reduced resolution meshes. How best to create such meshes is an open question.

Filtering generally requires visibility ordering the entire mesh and may result in holes or gaps in the image. The coalescing method can result in nonconvex cells not amenable to ordering by the MPVO algorithm.

The most promising approach seems to be to retriangulate a random subset of the original vertices of the mesh using a (conformed) Delaunay triangulation [14, 25]. This will decrease the time for visibility ordering since fewer cells need to be ordered; the MPVO algorithm can be used; and there will be no holes in the image. At this time, algorithms and implementations of conformed Delaunay triangulations are in the research stage.

One way to render only selected cells is by rendering the maximal independent set of the adjacency graph of the cells of the mesh. (An independent set is a set of vertices in a graph, no two of which are adjacent. A maximal independent set is an independent set which will no longer be one when any vertex is added to the set.) Such a set can be calculated as follows. For each cell c in the mesh, render c only if none of c's neighbors have been rendered. Regularity can be avoided in this procedure if the indices of the cells are randomly permuted. A disadvantage of this method is that the degree of the nodes imposes bounds on the level of filtration allowed. Typically, maximal independent set filtration of tetrahedral meshes results in approximately 20–25% of the cells being rendered. A volume rendered image using maximal independent set filtering is shown in Color Figure 5.

More flexibility in the level of filtration can be achieved by selecting cells at random in the mesh. In general random selection methods have a nice distribution property. A volume rendered image using 25% random filtering is shown in Color Figure 6. Comparative timings for these filtering methods are given in Table 2 for a 1,003,520 cell mesh.

Another technique is to discard every n-th cell that is output from the visibility ordering, possibly tagging adjacent cells to prevent their removal.

For any of these methods, it may be valuable to flag cells which have interesting or rare data values at their vertices so these cells are not filtered out. Such a determination can be made based on domain specific knowledge and/or on a histogram analysis of the data.

Low-reject, high-reject or band-reject filtering can be used to eliminate uninteresting data. Filtered cells can be flagged 'do-not-render' in a preprocessing step. Similar filtration could be achieved by the density transfer function; however, such filtration would then occur at run time.

Since the size of data sets continues to grow with the increase in hardware computing power, it seems that the concept of reduced resolution meshes may remain valuable for some time to come. It may be particularly useful when the data set is being previewed or the color maps are being selected.

4 Conclusion

We have examined several methods for achieving interactive volume rendering: approximations to the splatting process, the use of high performance graphics hardware, parallelization and reduced resolution meshes.

By the use of parallelization and splatting approximations, the goal of interactive volume rendering has been found to be feasible, even when the data sets are very large. Nonrectilinear data sets with up to 100,000 cells were rendered volumetrically in less than 2 seconds, data sets with over 1,000,000 cells in 16 seconds. Using the filtering methods described herein, this performance is possible for even larger data sets.

(See color plates, p. CP-5.)
Table 2: Comparative timings for two filtration methods using the PT Algorithm for 1,003,520 tetrahedra.

<table>
<thead>
<tr>
<th>Method</th>
<th>No Filtering</th>
<th>Max Indep Set</th>
<th>20% Random</th>
<th>50% Random</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial</td>
<td>79.6 sec.</td>
<td>35.1 sec.</td>
<td>33.4 sec.</td>
<td>55.9 sec.</td>
</tr>
<tr>
<td>6 CPUs</td>
<td>30.2 sec.</td>
<td>13.9 sec.</td>
<td>13.0 sec.</td>
<td>21.6 sec.</td>
</tr>
</tbody>
</table>

It will remain for computational scientists to validate the utility of these volume rendering techniques. They should be valuable for data previewing and for setting initial viewing parameters. The goodness of a particular approximation method depends on the specific data set being visualized, the viewpoint, the transfer functions used, etc. The ultimate test is whether the scientist finds the image useful for understanding his or her data, and is not misled.

Sharp isosurfaces do not seem to be feasible for the splatting methods described herein. If the density transfer function simulates a set of very narrow band pass filters, then the density map will have a set of very narrow rectangular pulses. Since the PT method averages the density at the front and back faces of a cell, the PT method could completely miss an isosurface passing through the cell.

Solids or polygons can be embedded in a volumetrically rendered image by generating a new mesh using a Delaunay triangulation that conforms to the vertices of the old mesh and the geometric description of the embedded objects.

It has been pointed out [17, 23] that because only 8 bits are used for the opacity channel on many high performance workstations this could lead to distortions in splatted images which are created using hardware opacity blending. This is due to roundoff error when a lot of cells have very small opacity. However, this was not observed to be a problem in the experiments that I performed. If an image was too transparent then the density map or attenuation factor was adjusted. However, it would be helpful to implement a volume renderer which did not rely on hardware blending in order to compare the the images.

When the MPVO algorithm is used, it outputs the cells in layers from front to back. Then the image unfolds as if a cutting plane perpendicular to the line of sight was sweeping over the image towards the viewer.

One of the most interesting discoveries in this work was the amount of information that could be gained by watching the image being rendered. The fine structure of the interior of the scalar field could be clearly discerned during the rendering process. It was known that watching this process could be useful; however, the degree of utility far surpassed my expectations.

The rendering process became an animation. It was quite fascinating to watch; sometimes it could be breath-taking. If the rendering time was significantly faster, or much slower, this effect would be lost. This way of experiencing data provides information that is not available by looking at a finished image, even with transparency.

5 Acknowledgement

Helpful conversations were had with Herbert Edelsbrunner, Peter Shirley and Allan Tuchman. This work was partially supported by the U. S. Department of Energy under Grant DE-FG02-85-ER25001, the Air Force Office of Scientific Research Grants AFOSR-90-0044, and Sun Microsystems Inc. I am grateful to Silicon Graphics Inc. for an equipment loan.

References


