Anisotropic Volume Rendering for Extremely Dense, Thin Line Data

Greg Schussman[*]
Stanford Linear Accelerator Center

Kwan-Liu Ma[†]
University of California at Davis

Abstract

Many large scale physics-based simulations which take place on PC clusters or supercomputers produce huge amounts of data including vector fields. While these vector data such as electromagnetic fields, fluid flow fields, or particle paths can be represented by lines, the sheer number of the lines overwhelms the memory and computation capability of a high-end PC used for visualization. Further, very dense or intertwined lines, rendered with traditional visualization techniques, can produce unintelligible results with unclear depth relationships between the lines and no sense of global structure. Our approach is to apply a lighting model to the lines and sample them into an anisotropic voxel representation based on spherical harmonics as a preprocessing step. Then we evaluate and render these voxels for a given view using traditional volume rendering. For extremely large line based datasets, conversion to anisotropic voxels reduces the overall storage and rendering for $O(n)$ lines to $O(1)$ with a large constant that is still small enough to allow meaningful visualization of the entire dataset at nearly interactive rates on a single commodity PC.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism; I.4.10 [Image Processing]: Image Representation—Volumetric.

Keywords: anisotropic lighting, line data, scientific visualization, vector field, volume rendering

1 Introduction

Many scientific investigations require the study of complex 3D or 4D vector fields such as fluid flow field, electric field, and magnetic field. Displaying the field lines directly is popular. However, when the field consists of a large number of dense, intertwined lines, it becomes very difficult to depict the spatial relationships between the lines and their intrinsic global and local structures. For example, one problem, known as the dark current problem, under study by the scientists at the Stanford Linear Accelerator Center (SLAC) for the design of National Linear Collider (NLC) requires visualization of very dense particle paths from the simulation of particle accelerator structures. Even in the 2D case, the paths become a tangled mess long before global patterns emerge. Another source of line data results from integrating a very large number of streamlines through CFD data to capture intricate structural details in velocity and vorticity fields.

The visualization challenge is to convey, clearly and interactively, the global structure of the dark current particle paths, or to show their relationship with electric and magnetic fields. This is a more difficult problem than simple vector field visualization because each point in a vector field maps to exactly one vector, but for path data, a point may map to any number of paths. For vector field visualization using streamlines, the line density can be adjusted by changing the number and placement of seed points. However, for measured or simulated path data, all paths are potentially significant so no path can be automatically discarded to reduce path density.

This paper presents Anisotropic Volume Rendering (AVR), a new technique making possible the depiction of extremely dense, fine field lines. Our basic approach is to subdivide the region of interest into cubes, which have their line data sampled and converted into voxels with anisotropic radiance and opacity. Voxels are represented efficiently by non-linear approximation where only significant spherical harmonics coefficients from a truncated Laplace series are stored. Direct volume rendering is then performed on traditional voxels computed from the anisotropic voxels, each evaluated according to a given viewing direction. Interactive scaling of line thickness is possible. The resulting voxel data size has a fixed upper bound, thereby enabling an overview of extremely large data on a commodity PC.

2 Previous Work

Spherical harmonics were first introduced into computer graphics in 1987 for representing bidirectional reflection functions from surface bump maps [Cabral et al. 1987]. In general, they are useful for sampling and reconstructing a scalar function on a sphere [Glassner 1995]. In particular, they have been used for representing BRDFs for material surfaces and directional intensity distributions for surfaces [Sillion et al. 1991] and, more recently, for real-time rendering using complex environment maps converted into the frequency domain [Ramamoorthi and Hanrahan 2001]. One approach to anisotropic function representation uses a non-linear (discarding insignificant coefficients) wavelet lighting approximation [Ng et al. 2003] to be faster than linear spherical harmonics. However, one can obtain speed increases by discarding insignificant spherical harmonic coefficients in a similar way, which is what we do. So in the same sense, our approach would be classified as using a non-linear Laplace Series approximation. The use of spherical harmonics for capturing anisotropy has so far been limited to one-sided surfaces [Ramamoorthi and Hanrahan 2001; Sloan et al. 2002; Ng et al. 2003].

The illuminated line work most closely related to ours is that by [Stalling et al. 1997]. Their work uses 2D textures of precomputed illumination based on the work of [Banks 1994], which treats lines as 1-manifolds in 3-space. A consequence

[*]schussman@slac.stanford.edu
[†]ma@cs.ucdavis.edu, http://www.cs.ucdavis.edu/~ma

IEEE Visualization 2004
October 10-15, Austin, Texas, USA
0-7803-8788-0/04/$20.00 ©2004 IEEE

of this is that, for any point on the curve, there is no clear surface and therefore no clear surface normal vector (rather, there are an infinite number of surface normals which are constrained to a plane). The diffuse term for the shading equation suffers from "excess brightness" which can be alleviated by exponentiating that term. Although values for this exponent are presented, we have found that some manual adjustment is needed in practice. Line primitives have some additional drawbacks. They do not provide the perspective depth cue. For extremely dense sets of lines, transparency is needed to prevent the rendering of an opaque mass. But transparency introduces its own set of difficulties. Limited precision in graphics hardware can cause severe quantization error for very transparent lines. Transparency also calls for back-to-front compositing of the line fragments, but this sorting can be prohibitively expensive for extremely large datasets. It is pointed out by [Stalling et al. 1997] that back-to-front compositing of approximately sorted segments (by an individual axis) yields only about 1% pixels with somewhat incorrect color, and that display lists of pre-sorted lines (along the major axes) speeds rendering. However line datasets too large to fit in memory preclude storing multiple sorted copies and the use of display lists. Furthermore, for very dense lines, approximate sorting can result in significant visual artifacts. Our approach is more closely related to what Banks refers to as promoting the manifold dimension; that is, we treat curves as tubes, so that normal vectors are cleanly defined, and there is a real surface with a visible side over which to integrate a shading equation. However our goal is to handle tubes projecting to have sub-pixel widths (consistent with averaging down a supersampling of shaded geometric tubes), so these integrated values are accumulated into floating point spherical harmonics coefficients for anisotropic voxels. This provides sub-pixel illumination consistent with multi-pixel illumination, correct depth ordering (except within an individual voxel), a fixed upper bound on memory requirements regardless of the line data size, and provides the perspective depth cue if the volume renderer renders in perspective. This also accumulates all line contributions to a voxel prior to entering graphics hardware, so that an individual voxel is not subject to accumulated quantization error in hardware. Volume haloing [Interrante and Grosch 1998], which requires more than one pixel of projected tube thickness, is not appropriate for what we intend to accomplish.

The work most closely related to the volumetric aspect of ours uses three dimensional textures to render fur [Kajiya and Kay July 1989], and mentions the desirability of the *Painter's Illusion* in which far more detail is suggested than is actually present on very close examination of a region of an image. Toward this end, they model teddy bear fur as a dense block, and tile this over the surface of the bear. The diffuse component of their lighting model is essentially the same as ours, but their specular model assumes uniformly dense fur and diffraction, which is not appropriate for the highly non-uniform density datasets which motivated AVR. They mention that the specular contribution can be derived using an approach similar to the diffuse contribution, but they admit that doing so is difficult and results in a model which is "quite complex". A contribution of this paper is an efficiently computable, nearly analytical solution to this specular problem. Their representation is based on storing a density, a frame, and a reflectance function, where the reflectance function is common to the whole dataset. In contrast, the AVR representation uses Laplace Series coefficients to store anisotropic density and radiance, with the frame assumed to be common to the whole dataset. Their approach makes numerous assumptions about the teddy bear fur, but these do not hold for scientific datasets. The AVR approach only assumes that the fiber thickness is smaller than the voxel dimensions. Finally, they use ray tracing to producing nice shadowing effects during rendering. If global illumination were to be used with AVR, it would occur during preprocessing, but could then be rendered as quickly as other lighting models.

Light transport and illumination of micro geometry has also been an area of graphics research. In the work of [Daubert et al. 2003] visibility and illumination are precomputed for tiled repetitive micro-geometry by raycasting a tile. This allows efficient rendering with global illumination, but the preprocessing method is prohibitively expensive because our datasets are much larger and do not contain tiled repetitive patterns. By restricting our micro-geometry to tubes and using a nearly analytical function for precomputation of local illumination and occlusion for tubes, we are able to preprocess very large datasets in a fraction of the time that they take to generate. The work of [Sillion et al. 1991] also considers micro-geometry, but assumes the overall reflection properties do not vary over a surface. Recent research has also been done on light scattering from human hair [Marschner et al. 2003] where the hair is not opaque and light undergoes multiple scattering within a fiber. Their goal is match the visual qualities of real hair. In contrast, our goal is to match opaque supersampled tube geometry illuminated with a given shading model. We want a visual match for tubes so thin that supersampling is intractable (e.g., for sets of fibers which are orders of magnitude thinner and denser than human hair).

3 Anisotropic Volume Rendering

The AVR process, illustrated in Figure 1, consists of two phases, one for preprocessing, and the other for rendering. A traditional voxel has only an opacity and color, which are isotropic (independent of viewing direction). An AVR voxel has opacity and color intensity, but these are anisotropic.

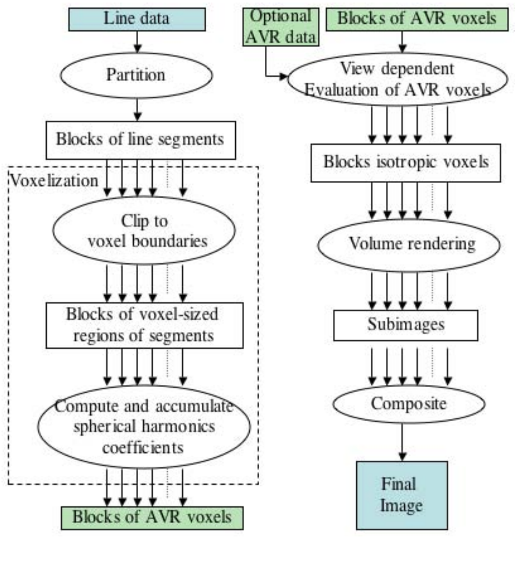

Figure 1: Flowchart of AVR preprocessing (left) and rendering (right).

The input for AVR consists of line data, where these lines or curves are represented as line segments. Each segment has a beginning and an ending point in 3D, and an associated radius and material properties. The first step in preprocessing is to partition the (presumably gigantic) line dataset into blocks which are of manageable size. Because the input data is too large to fit in memory, and the number of blocks to be saved can exceed the maximum allowable number of open files, we buffer the line segments in memory, and only open and append to a file when more than a certain number of segments have accumulated for the corresponding block. This allows all line data to be partitioned in a single streaming pass without incurring significant penalty for opening and closing files.

Once the data has been partitioned into blocks, the blocks can be voxelized in an embarrassingly parallel fashion. Voxelization consists of clipping line segments to voxel boundaries and accumulating spherical harmonic coefficients. Clipping is straightforward; each block corresponds to an n^3 volume of voxels, so every line segment is clipped to the cubical boundaries of any voxel region through which it passes. The Laplace Series coefficients are then computed for the lighting contribution of the clipped subsegment, and accumulated in the corresponding AVR voxel. Lighting can be from multiple light sources, which can be any mixture of ambient and directional light sources, and the directional light sources can be either fixed in world space (e.g., the sun) or fixed in eye space (e.g., a headlight). Once all segments have been processed, insignificant coefficients are discarded. The details of these calculations are presented in Section 4.

The final output of the preprocessed data is blocks of AVR voxels. In our implementation, each block is a 3D array of pointers which are null for empty (completely transparent) voxels, or point to an instance of an AVR voxel class. Each block stores the indices for its location within the entire volume, and thereby can "know" where it is located in 3D. An AVR voxel consists of a list of quantum numbers and an associated list of coefficients. Because these coefficients are accumulated in software, no quantization error is introduced prior to hardware rendering. If the addition were done in limited precision graphics hardware, a dense bundle of fine lines could have each individual line quantized to 0, even though collectively the bundle should be visible.

The rendering phase takes as input blocks of AVR voxels, the eye position and viewing direction of an observer, and other viewing parameters such as transfer functions. These are used to evaluate a block of AVR voxels. When an AVR voxel is evaluated, the input is a viewing direction, and the output is an rgb color intensity and an opacity value. In effect, evaluating a block of AVR voxels produces a block of traditional voxels corresponding to the current view. Traditional volume rendering can then be performed in parallel to produce sub images which are composited to produce a final image.

The opacity of AVR voxels is directly proportional to the corresponding segment radii, so the equivalent of interactively scaling the radii can be accomplished by scaling the opacity of an evaluated AVR voxel. If there are more than one input line type (for example electric field lines and magnetic field lines), these can be preprocessed independently, and can have their evaluated results scaled and combined interactively, also without further preprocessing. Opacity transfer functions can also be manipulated once the AVR voxel opacity has been evaluated.

4 Averaged Lighting Derivations

This section provides the mathematical details for implementation. There are two steps in execution. The first is to compute the averaged lighting for a single line segment from a single view with a single light source. This is repeated for any additional segments lying within the current voxel. This would also be repeated for additional light sources. For this specific view, the color and opacity are accumulated via a weighted average.

The organization for this section is as follows. First we state our assumptions and conventions, and describe our notation. Then we provide a starting point for deriving expressions for averaged lighting contribution of a generic light source which can be any shading function. This starting point is an integral for which an efficiently computable analytical or numerical solution is needed to make preprocessing of large datasets tractable. Next, special limits of integration for the specific case of a directional light source are given, followed by the weighted averaging equations for color and opacity. This section concludes with the solution of the averaged lighting integrals for the specific case of OpenGL's Phong style lighting model.

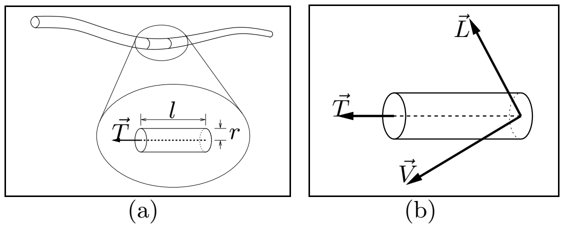

Figure 2: Each segment has a radius r, length l, and unit tangent vector $\vec{T}$ (a). Input unit vectors consist of $\vec{T}$, the light source $\vec{L}$, and view $\vec{V}$ (b).

4.1 Conventions and Notation

First, we make some simplifying assumptions. The mathematical formulation of intensity (as a function of view direction, curve tangent, and light direction) assumes orthographic projection during sampling. However, during visualization, the overall view uses perspective projection, and permits up-close viewing. The derivations in this section assume that the radius of any cylinder is less than the width of the surrounding voxel. The relative depth relationship of two line segments is not preserved within the same voxel but is preserved between separate voxels.

Lighting parameters consist of a light vector, $\vec{L}$, a view vector, $\vec{V}$, and light properties. Lines (curves) are composed of line segments which are modeled as cylinders. Each cylinder has radius r, and length l. A cylinder has material properties: ambient, diffuse, and specular coefficients for red, green, and blue color channels, and a shininess exponent. A unit vector $\vec{T}$, which lies along the axis of the cylinder, can be thought of as a normalized version of the average tangent vector for this cylinder's portion of the curve. Figure 2a shows these cylinder parameters in relation to one another. The unit vectors $\vec{V}$ and $\vec{L}$ points toward the eye and a distant light source, respectively, and are depicted in Figure 2b in relation to $\vec{T}$.

We construct a 3D *orthogonal basis* from $\vec{T}$ and $\vec{V}$. Disregarding the special case where a cylinder is viewed "end on" (because there would be no visual contribution), $\vec{T}$ and $\vec{V}$ are linearly independent, and the orthogonal projection of $\vec{V}$ into the plane perpendicular to $\vec{T}$ is the unique vector $\vec{V}_p$ where $0 < \left|\vec{V}_p\right| \leq 1$. The three basis vectors are $\vec{T}$, $\vec{N}$, and $\vec{B}$ where the latter two are computed according to $\vec{N} = \mathcal{N}\left(\vec{V}_p\right)$, $\vec{B} = \vec{T} \times \vec{N}$, $\vec{V}_p = \vec{V} - \left(\vec{V} \cdot \vec{T}\right)\vec{T}$, where $\mathcal{N}$ denotes vector normalization. The $\vec{N}\vec{B}$ plane is perpendicular to the $\vec{T}$ axis, effectively forming a cross-sectional slice of the cylinder from Figure 2b. Note that $\vec{N}$ is the cylinder surface normal which most closely points toward the eye. We define any other unit surface normal in terms of the angle θ, which is the rotation in the $\vec{N}\vec{B}$ plane, going clockwise about the $\vec{T}$ axis, so

$$\vec{n}(\theta) = \vec{N}\cos\theta + \vec{B}\sin\theta. \tag{1}$$

The projected visible side of the cylinder corresponds to $-\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2}$.

4.2 Generic Initial Derivation

The initial portion of derivations for averaged lighting over a cylinder surface have a common starting point independent of the expression for the light source. Using $\mathcal{F}$ as a generic placeholder for a light contribution from any shading model, the generic averaged intensity, $\bar{I}_{\mathcal{F}}$, is the shading intensity integrated over the projected visible cylinder surface divided by the projected area of the cylinder surface:

$$\bar{I}_{\mathcal{F}} = \int_{S_{VL}} \mathcal{F}\ \cos\phi\, dS_{VL} \Big/ Area_{vis}, \tag{2}$$

Here, S_{VL} denotes the projected area that is both visible and illuminated. Note that S_{VL} depends on $\mathcal{F}$, and that $S_{VL} \subseteq S$, because S denotes all projected visible area, regardless of whether that area is illuminated. The limits of integration for θ (corresponding to S_{VL}) are not necessarily $\frac{-\pi}{2}$ and $\frac{\pi}{2}$, and depend on what $\mathcal{F}$ is. The projected visible area of a cylinder surface is

$$Area_{vis} = 2\,rl\,\left(\vec{V}\cdot\vec{N}\right). \tag{3}$$

Equation 2 can be partially evaluated, taking Equation 3 into account and using $dS = r\,d\theta\,dz$ from cylindrical coordinates. The simplified resulting expression concludes the initial derivation, and is now ready to accept $\mathcal{F}$:

$$\bar{I}_{\mathcal{F}} \;=\; \tfrac{1}{2}\int_{\theta_0}^{\theta_1} \mathcal{F}\ \cos\theta\, d\theta. \tag{4}$$

The limits of integration for θ for the cylinder surface which is both visible to the eye and illuminated by a directional light source are illustrated in Figure 3, in which $\vec{V}_p$ denotes the projection of $\vec{V}$ into the $\vec{N}\vec{B}$ plane, $\vec{L}_p$ denotes the projection of $\vec{L}$ into the $\vec{N}\vec{B}$ plane, and α is the angle between the projected view and projected light vectors. From this figure, the limits of integration for a directional light source are

$$\theta_0 = \max\left(\tfrac{-\pi}{2},\ \alpha - \tfrac{\pi}{2}\right), \qquad \theta_1 = \min\left(\tfrac{\pi}{2},\ \alpha + \tfrac{\pi}{2}\right). \tag{5}$$

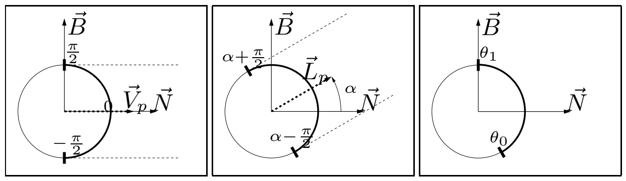

Figure 3: The portion of the cylinder surface visible to the eye (left) may not be the same as the portion of the surface illuminated (middle). Integrating only over the intersection of these two regions (right) simplifies the sampling equations for directional light sources.

4.3 Weighted Average within a Voxel

Averaged lighting expressions from Section 4.2 provide the lighting contribution of a single cylinder for a given view. This paper targets extremely dense line datasets where many line segments can lie within a single AVR voxel. We use a weighted average to combine the contributions of multiple segments for this view. For n cylinders numbered $i \in \{0, 1, \ldots, n-1\}$, we denote the rgb color triplet by C_i, the projected visible area (as computed in Equation 3) by A_i, and the final color and opacity for a voxel by $\mathcal{C}$ and $\mathcal{O}$, respectively. The weighted averages are

$$\begin{aligned}\mathcal{C} &= \sum_{i=0}^{n-1}(C_i)(A_i)\Big/\sum_{i=0}^{n-1}A_i, &(6)\\ \mathcal{O} &= 1 - \prod_{i=0}^{n-1}\left(1 - A_i\Big/\frac{S_c}{4}\right). &(7)\end{aligned}$$

where S_c is the surface area of the cube, and $S_c/4$ is the average visible surface area of a cube [Arvo and Kirk 1989].

4.4 Application to OpenGL Lighting

The mathematics of OpenGL's lighting are given in Chapter 5 of the OpenGL Programming Guide (the "Red Book") [Woo et al. 1999]. To summarize briefly, OpenGL uses the following (in slightly different notation) to produce the color $I = I_a + I_d + I_s$ where I is the total intensity rgbα vector, and the a, d, and s subscripts correspond to ambient, diffuse, and specular, respectively. Denoting componentwise vector multiplication by $*$,

$$\begin{aligned}I_a &= (A_m * A_l), &(8)\\ I_d &= (D_m * D_l)\ \max\left(\vec{L}\cdot\vec{n},\, 0\right), &(9)\\ I_s &= (S_m * S_l)\ \max\left(\vec{H}\cdot\vec{n},\, 0\right)^k, &(10)\end{aligned}$$

where $\vec{n}$ is the surface normal vector, A, D, and S correspond to ambient, diffuse, and specular, respectively. The subscripts m and l correspond to material and light, respectively. The "halfway" unit vector is $\vec{H} = \mathcal{N}\left(\vec{L} + \vec{V}\right)$.

In this case, the average intensity $\bar{I} = \bar{I}_a + \bar{I}_d + \bar{I}_s$. The individual sub-expressions come from substituting Equations 8, 9, and 10 in place of $\mathcal{F}$ in Equation 4. The ambient term is simply $\bar{I}_a = A_m A_l$, and the easily obtained simplified diffuse term is

$$\begin{aligned}\bar{I}_d = \tfrac{D_m D_l}{8}\Big[&\left(\vec{L}\cdot\vec{N}\right)\Big(\sin 2\theta_1 - \sin 2\theta_0 + 2(\theta_1 - \theta_0)\Big)\\ &- \left(\vec{L}\cdot\vec{B}\right)\Big(\cos 2\theta_1 - \cos 2\theta_0\Big)\Big].\end{aligned} \tag{11}$$

Space constraints preclude showing the complete derivations here, but they are available in [Schussman 2003]. The specular expression is more challenging, so we provide a summarized derivation here. The main difficulty surrounding integration of the specular expression is the k exponent, which causes analytical solutions (for known k) to expand out to so many trigonometric terms as to be computationally impractical (even for preprocessing large datasets).

Starting with Equation 4, and using $\mathcal{F} = I_s$ from Equation 10 gives

$$\bar{I}_s = \tfrac{1}{2}\int_{\theta_0}^{\theta_1} S_m S_l \, \left(\vec{H}\cdot\vec{n}(\theta)\right)^k \cos\theta \, d\theta. \tag{12}$$

Substituting Equation 1 into Equation 12, and distributing the dot product yields

$$\bar{I}_s = \tfrac{S_m S_l}{2}\int_{\theta_0}^{\theta_1} \left[\left(\vec{H}\cdot\vec{N}\right)\cos\theta + \left(\vec{H}\cdot\vec{B}\right)\sin\theta\right]^k \cos\theta \, d\theta. \tag{13}$$

We now rearrange and split up this expression so that the part which does not have an analytical solution will at least lend itself to a precomputed numerical solution. Toward this end, we introduce $\vec{H}_p$ and β; similar to $\vec{L}$ and $\vec{L}_p$ in Figure 3, $\vec{H}_p$ is the projection of $\vec{H}$ into the $\vec{N}\vec{B}$ plane, so

$$\vec{H}_p = \left(\vec{H}\cdot\vec{N}\right)\vec{N} + \left(\vec{H}\cdot\vec{B}\right)\vec{B}, \tag{14}$$

and β is the angle between $\vec{H}_p$ and the $\vec{N}$ axis, which can be computed as: $\beta = \texttt{atan2}\left(\mathcal{N}(\vec{H}_p)\cdot\vec{B},\ \mathcal{N}(\vec{H}_p)\cdot\vec{N}\right)$, where `atan2` is ISO 9899 compliant. The orthogonal projection of $\vec{H}_p$ onto $\vec{N}$ is $\left|\vec{H}_p\right|\cos\beta$. Because $\vec{H}_p$ and $\vec{H}$ both have the same $\vec{N}$ and $\vec{B}$ components, the following equations result:

$$\vec{H}\cdot\vec{N} = \left|\vec{H}_p\right|\cos\beta, \qquad \vec{H}\cdot\vec{B} = \left|\vec{H}_p\right|\sin\beta. \tag{15}$$

Substituting these into Equation 13, then substituting in a trigonometric angle addition identity, and using a change of variables where $u = \theta - \beta$, and using another angle addition identity gives

$$\begin{aligned}\bar{I}_s = \tfrac{S_m S_l}{2}\left|\vec{H}_p\right|^{k-1}\Big((\vec{H}\cdot\vec{N})\int_{\theta_0+\beta}^{\theta_1+\beta}\cos^{k+1}u\,du \\ -(\vec{H}\cdot\vec{B})\int_{\theta_0+\beta}^{\theta_1+\beta}\cos^{k}u\sin u\,du\Big).\end{aligned} \tag{16}$$

The second integral in Equation 16 has a clean, closed-form solution, but the first integral does not. The integral of $\cos^{k+1} u$ is an order k multivariate polynomial in $\cos u$ and $\sin u$ with $k+2$ terms for even k, or $k+1$ terms for odd k. If u is bounded below and above, (which we denote by u_{min} and u_{max}, respectively), the integral can be precomputed numerically and stored in a lookup table. From our change of variable and we obtain bounds for β, and then substitute back to get $u_{min} = -\pi \le u = (\theta - \beta) \le \pi = u_{max}$. Because these bounds exist, a function $F(u_0, u_1)$, implemented in terms of the lookup table, can be used to replace the problematic integral, providing our final expression for the averaged specular contribution:

$$\begin{aligned}\bar{I}_s = \tfrac{1}{2}S_m S_l\left|\vec{H}_p\right|^{k-1}\Bigg[\left(\vec{H}\cdot\vec{N}\right)F(\theta_0+\beta,\theta_1+\beta) + \\ \left(\vec{H}\cdot\vec{B}\right)\left(\frac{\cos^{k+1}(\theta_1-\beta)-\cos^{k+1}(\theta_0-\beta)}{k+1}\right)\Bigg],\end{aligned} \tag{17}$$

where θ_0 and θ_1 are given by Equations 5.

5 Voxelization

The previous section was about obtaining the averaged lighting color, $\mathcal{C}$, and opacity, $\mathcal{O}$, for a single view of multiple line segments located within a voxel. These results are functions of a viewing direction. This section is about sampling (evaluating) those functions for many different views, and converting those samples into a set of Laplace Series coefficients, which compactly represent an approximation of the functions for all viewing directions. Compactness is necessary for fitting a large amount of AVR voxel data in memory. This representation also allows $\mathcal{C}$ and $\mathcal{O}$ to be reconstructed for any specific viewing direction in a computationally efficient manor, to allow faster rendering. In the convention of [Porter and Duff 1984], $\mathcal{C}$ and $\mathcal{O}$ are non-premultiplied color and opacity.

The sampling function only takes a view, a cylinder and a light source. It can be called multiple times, once for each light source, and the results can be combined. Light sources can have any direction, either held fixed in world space, or held fixed in eye space, or defined as some other function of viewing direction. Samples from different cylinders within the same voxel are combined as well. Once the compact representation is obtained, sampling need not be repeated unless the combined lighting functions are changed (examples include light sources being added or removed, a world space light being moved in world space, or a light direction being redefined as a different function of viewing direction).

Spherical harmonic functions form the basis for the Laplace series in a way that is directly analogous to sine and cosine harmonics forming the basis for Fourier series. We adopt the spherical coordinates convention most commonly used in physics, as follows. The azimuthal angle, ϕ, lies in the XY plane, is measured from the X axis, is confined to the interval $[0, 2\pi]$, and corresponds to longitude. The polar angle, θ, is measured from the Z axis, is confined to the interval $[0, \pi]$, and corresponds to colatitude (where colatitude is $90^\circ-$latitude). Cartesian coordinates are obtained from spherical coordinates as $(x, y, z) = (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta)$. The *normalized spherical harmonics* are typically denoted by $Y_{l,m}$ and defined in real form by

$$Y_{l,m}(\theta,\phi) = \begin{cases} N_{l,m}\,P_{l,m}\cos\theta\ ,\cos m\phi, & m>0 \\ N_{l,0}\,P_{l,0}(\cos\theta)\,/\,\sqrt{2}, & m=0 \\ N_{l,m}\,P_{l,|m|}(\cos\theta)\,\sin|m|\,\phi & m<0 \end{cases}, \tag{18}$$

where l denotes *order*, m denotes *degree*, and l and m are quantum numbers subject to $0 \le l < \infty$ and $-l \le m \le l$. The normalizing constants are by

$$N_{l,m} = \sqrt{((2l+1)/2\pi)\,(l-|m|)!/\,(l+|m|)!}. \tag{19}$$

where $P_{l,m}$ are the associated Legendre polynomials which can be evaluated efficiently using the recurrence relations

$$P_{0,0}(x) = 1, \tag{20}$$

$$P_{m,m}(x) = (1-2m)\sqrt{1-x^2}P_{m-1,m-1}(x), \tag{21}$$

$$P_{m+1,m}(x) = x(2m+1)P_{m,m}(x), \tag{22}$$

$$P_{l,m}(x) = x\frac{2l-1}{l-m}P_{l-1,m}(x) - \frac{l+m-1}{l-m}P_{l-2,m}(x). \tag{23}$$

The Laplace series has some useful properties that stem from spherical harmonics. A function on a sphere is uniformly represented in the azimuthal and polar directions when all m coefficients are obtained for a given l. Also, if the intensity and opacity functions have bilateral and radial symmetry (which they do for Phong style shading illuminated by a headlight), all coefficients with odd l are zero and need not be computed, regardless of how these functions are oriented with respect to the sphere. The *Laplace series coefficients*, $C_{l,m}$, can be computed via the orthogonal projection of $f(\theta, \phi)$ onto the spherical harmonics basis functions. The coefficients can be approximated for discrete samples by approximating the projection surface integral via a summation over the surface weighted by the area corresponding to each sample. For n samples numbered $i \in \{0, 1, \ldots, n-1\}$ with corresponding coordinates (θ_i, ϕ_i) on the unit sphere, and with corresponding area A_i, the Laplace series coefficients are

$$C_{l,m} \quad = \quad \sum_{i=0}^{n-1} f(\theta_i, \phi_i)\, Y_{l,m}(\theta_i, \phi_i)\, A_i. \tag{24}$$

We distribute the samples over the sphere in a fairly uniform manner. An octahedron is placed with its center at the origin in spherical coordinates, and the radial component of its vertices are scaled to unity. Each triangular face is then subdivided into four sub-triangles by introducing vertices at the midpoint of each edge, and scaling the radial component of each new vertex back up to unity. For k subdivision iterations on a polyhedron with p initial triangular faces, n triangles are produced according to $n = 4^k p$. In selecting an appropriate n, we considered the trade-off between quality and sampling speed. Each triangle center provides a sample location (θ_i, ϕ_i) and associated spherical triangle area A_i. The images in this paper used an octahedron, subdivided two or three times for a total of 128 or 512 sample points on the sphere. We discard any coefficient which does not significantly contribute to the accuracy of the approximated $f(\theta, \phi)$. The RMS error contribution of a coefficient $C_{l,m}$, which is $|C_{l,m}|^2$.

6 Rendering

In traditional 3D textured volume rendering, there is one rgbα value per voxel. A voxel is isotropic; it has the same rgbα value regardless of viewing direction. An AVR voxel, however, is anisotropic. Rendering consists of two stages. First, AVR voxels are evaluated according to the current view, results in traditional voxels. The second stage is to render the traditional voxels using any preferred renderer.

For the most accurate image, the spherical harmonics basis functions should be evaluated per voxel, according to the viewing direction for that voxel. However, to reduce rendering time, the basis functions are evaluated only once per block of voxels, where the blocks are small enough that all view vectors from within the block are, to a good approximation, equal. Once the basis functions have been evaluated for the current view, the coefficients from each AVR voxel are used to scale them to produce a corresponding single rgbα value.

When voxel boundaries are filtered for rendering, opacity weighted interpolation [Wittenbrink et al. 1998] is more important for preventing visual artifacts than when rendering solids, because the artifact happens on both sides of the thin projected line. This is particularly significant for volume line data because the effect happens at the border of all empty voxels surrounding a line.

7 Results

This section begins with a few images to demonstrate correctness and quality relative to an unscalable (but correct) approach. This is followed by images demonstrating the capabilities of AVR on several large datasets from computational fluid dynamics and electromagnetic and particle tracking simulations.

In verifying correctness of AVR's operation, we compare identical views of a small (100k segments), artificially constructed dataset consisting of red vortex spirals. The benchmark image that we compare against is these spirals rendered as polygonal tubes at 10 times the screen resolution (accomplished by repeatedly rendering the polygonal scene one tile at a time to produce a 10×10 mosaic of tiles, and then averaging the results back down to the normal image size). This supersampling of the polygonal model was necessary because the small triangles composing it aliased badly at a comparable resolution. Both AVR and the supersampled polygonal approach used identical material properties, light properties, and light orientation. The left column of Figure 4 compares the AVR image (at 256^3 voxel resolution) with the benchmark image.

When the number of segments is increased tenfold, even the supersampling approach breaks down, and artifacts appear. The middle and right columns of Figure 4 compares AVR with the supersampled approach for the larger (1M segments) spiral dataset, and shows that AVR does not suffer from this sort of aliasing problem. The reason that supersampling does not scale is that when the polygonal tube radius is reduced by a factor of n, the needed number of tiles increases by a factor of n^2.

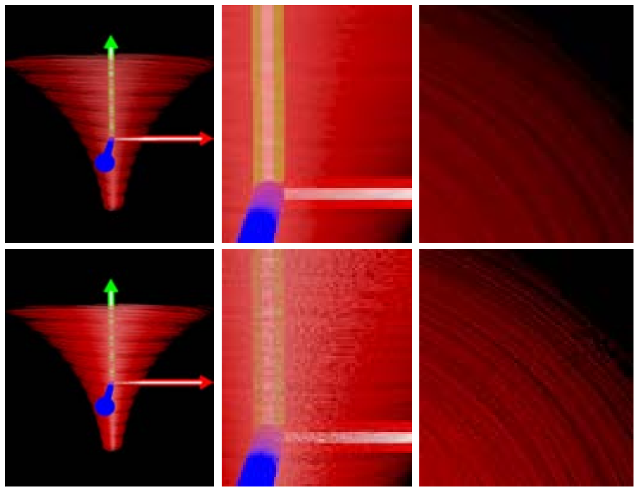

Figure 4: AVR images (top row) compare favorably to 10×10 supersampled geometry (bottom row), and avoid aliasing artifacts. less time to render.

The first simulation dataset, obtained from a dense streamline seeding of CFD simulation of air flow behind the tail of an aircraft, has over 1.3 million segments. The blue and orange lines represent velocity and vorticity, respectively. Figure 5 compares headlight illumination with a light held fixed in world space for different views. Each of these images consists of 256^3 AVR voxels and took 3 seconds to render.

The second simulation dataset consists of 10 million segments representing dark current paths of two categories of particles within a 5-cell linear accelerator structure. Red primary particles are emitted from the interior surface of the structure in response to strong local electric fields. Green secondary particles are emitted according to a stochastic distribution as the result of a particle collision with the surface. Secondaries greatly outnumber primaries. Figure 6 shows how the primaries can be seen alone or in the context of the secondaries by reducing the radius of the secondary paths and increasing the radius of the primary paths. Adjusting the radius can occur during the visualization run and does not require re-sampling the line data, although it does in-

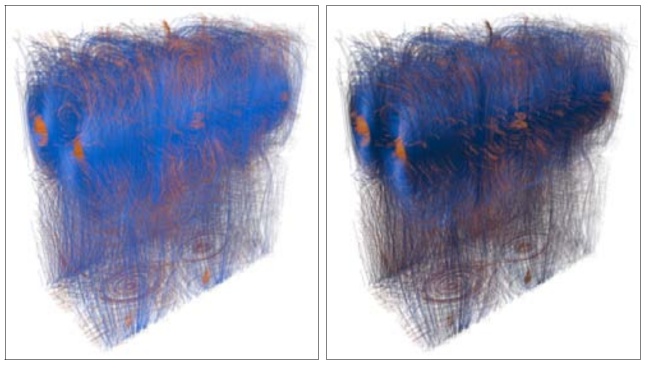

Figure 5: These images compare illumination with a white light for a headlight (left) and a fixed light shining straight down (right).

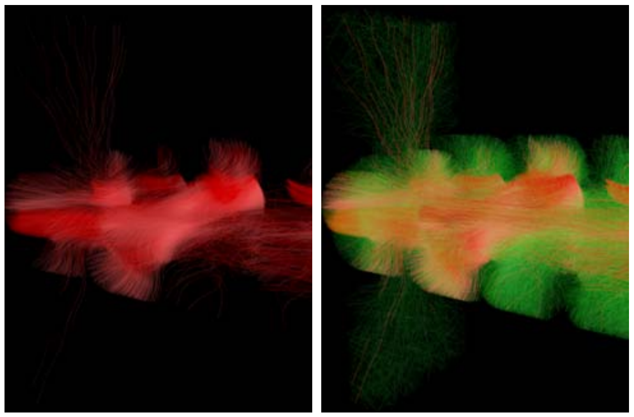

Figure 6: The left image shows primary particle paths only. The right image shows them in the context of secondaries whose paths have been deemphasized by reducing their radius. These were produced from 1024^3 voxel samples.

cur a performance penalty because the two sets of spherical harmonics coefficients would have to be merged at run time. The primaries image took 51 seconds to render, and the precombined image took 3 minutes. Preprocessing times are difficult to report meaningfully because preprocessing was done on a heterogeneous compute farm with CPUs having different speeds. The overall preprocessing CPU time was roughly 10% of overall simulation CPU time.

Another possibility with AVR is to sample at different resolutions so that once a user has seen an overview of the whole dataset and has identified an area of interest, a higher resolution sub-region can be swapped in and examined for fine detail. Interactively adjusting opacity can help discern local vs. global structure. This process is illustrated in Figure 7.

For a fixed AVR sampling resolution, there is a fixed upper bound on the storage requirements for the AVR data. This can convert $O(n)$ line segments to $O(1)$ storage. Although the constant is large, Figure 8 shows that the crossover can occur before some of the more dense datasets used in this paper. In other words, for suitably dense line data, AVR can provide useful data compression. As datasets grow in size and detail, this compression will become even more significant.

Similar to the case for storage, AVR also converts rendering time from $O(n)$ line segments to $O(1)$, again with a large

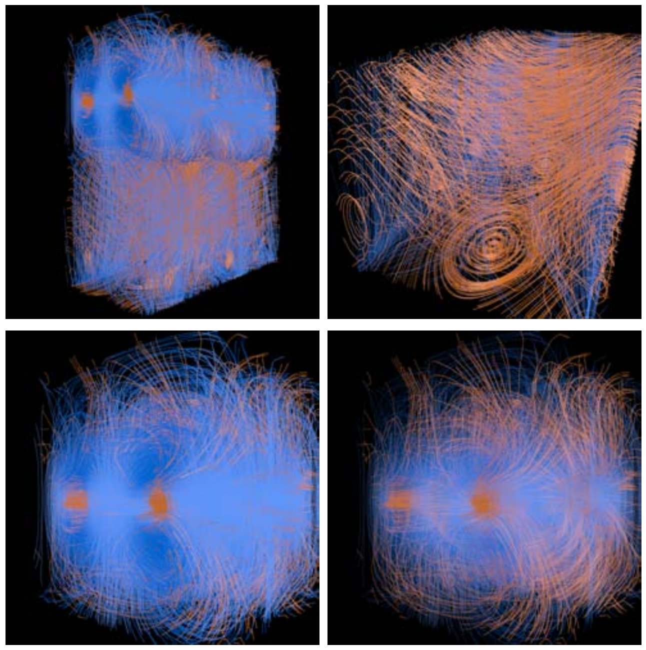

Figure 7: Once a global overview is presented (upper left), higher resolution blocks of AVR voxels can be loaded to show finer structure (upper right). Interactively adjusting opacity (bottom row) helps emphasize global or local detail.

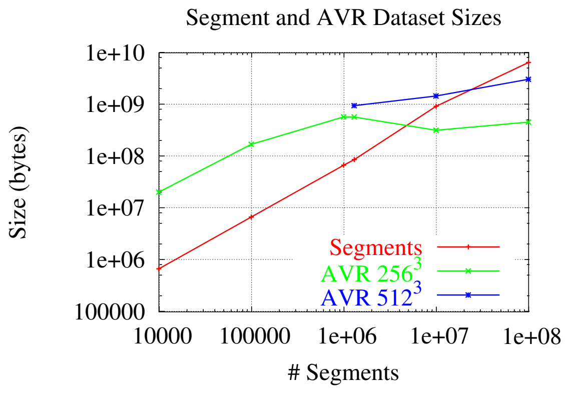

Figure 8: Disk storage requirements for segment data grow more quickly than for AVR as the number of segments increases.

constant. Figure 9 shows the crossover where AVR outperforms geometry for tubes in two ways. Tubes in a display list are clearly fast, but quickly run out of memory. Even ignoring memory and extrapolating the results for the display list, AVR still wins for the 100 million segment dataset. Tubes rendered in immediate mode avoid the memory problem, but are significantly slower than the display list. Although plain lines would be faster, doing so is still $O(n)$.

8 Conclusion and Future Work

This paper presents Anisotropic Volume Rendering as an efficient and effective way to show global and local detail in extremely dense line-based datasets. By using anisotropic vox-

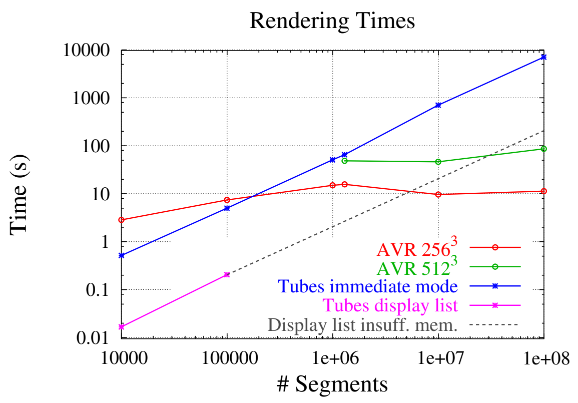

Figure 9: The large constant on the $O(1)$ rendering performance of AVR is still small enough to outperform the $O(n)$ geometry based methods for datasets we tried.

els, lighting details such as specular highlight help provide visual cues to convey the structure of the data. A specific solution for Phong style lighting as used by OpenGL is presented as an example, but the framework presented applies to more general lighting models. The AVR representation, based on a Laplace Series, is compact and efficient enough to compress very large and dense line datasets sufficiently to fit in memory and reduce rendering time over other line drawing methods. Because preprocessing is done entirely in software no quantization error occurs prior to rendering.

Future work consists of several things. Further optimization will be pursued. Because AVR has a large amount of available parallelism, it is a good candidate for running on a cluster for producing interactive visualization at very high resolution. Other lighting models will be investigated, including those which are anisotropic and/or non-photorealistic. Further perceptual enhancements should be possible. Additionally, the AVR voxel is a very general representation which could convey geometric primitives other than lines. The coming generation of graphics hardware may allow enough flexibility to evaluate the AVR voxels directly in hardware in spite of our nonlinear approach of discarding small Laplace Series coefficients to reduce storage.

9 Acknowledgments

This work has been sponsored in part by the U.S. National Science Foundation under contracts ACI 9983641 (PECASE award), ACI 0325934 (ITR), and ACI 00222991; and the U.S. Department of Energy under Memorandum Agreements No. DE-FC02-01ER41202 (SciDAC) and No. B523578 (ASCI VIEWS).

References

Arvo, J., and Kirk, D. 1989. *A survey of ray tracing acceleration techniques*. Academic Press, San Diego, CA, USA.

Banks, D. C. 1994. Illumination in diverse codimensions. *Proceedings of SIGGRAPH 94*, 327–334. ISBN 0-89791-667-0. Held in Orlando, Florida.

Cabral, B., Max, N., and Springmeyer, R. 1987. Bidirectional reflection functions from surface bump maps. In *Computer Graphics (Proceedings of SIGGRAPH 87)*, vol. 21, 273–281.

Daubert, K., Kautz, J., Seidel, H.-P., Heidrich, W., and Dischler, J.-M. 2003. Efficient light transport using precomputed visibility. *IEEE Computer Graphics and Applications 23*, 3 (May), 28–37.

Glassner, A. S. 1995. *Principles of Digital Image Synthesis*. Morgan Kaufmann, San Francisco, CA.

Interrante, V., and Grosch, C. 1998. Visualizing 3D flow. *IEEE Computer Graphics & Applications 18*, 4 (July – Aug.), 49–53.

Kajiya, J. T., and Kay, T. L. July 1989. Rendering fur with three dimensional textures. *Computer Graphics (Proceedings of SIGGRAPH 89) 23*, 3, 271–280. Held in Boston, Massachusetts.

Marschner, S., Jensen, H. W., Cammarano, M., Worley, S., and Hanrahan, P. 2003. Light scattering from human hair fibers. In *Proceedings of ACM SIGGRAPH 2003*, Computer Graphics Proceedings, Annual Conference Series. (to appear).

Ng, R., Ramamoorthi, R., and Hanrahan, P. 2003. All-frequency shadows using non-linear wavelet lighting approximation. *ACM Transactions on Graphics (TOG) 22*, 3, 376–381.

Porter, T., and Duff, T. 1984. Compositing digital images. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, ACM Press, 253–259.

Ramamoorthi, R., and Hanrahan, P. 2001. An efficient representation for irradiance environment maps. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, 497–500.

Schussman, G. L. 2003. *Interactive and Perceptually Enhanced Visualization of Large, Complex Line-Based Datasets*. PhD thesis, University of California Davis.

Sillion, F. X., Arvo, J. R., Westin, S. H., and Greenberg, D. P. 1991. A global illumination solution for general reflectance distributions. In *Computer Graphics (Proceedings of SIGGRAPH 91)*, vol. 25, 187–196.

Sloan, P.-P., Kautz, J., and Snyder, J. 2002. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM Press, 527–536.

Stalling, D., Zöckler, M., and Hege, H.-C. 1997. Fast Display of Illuminated Field Lines. *IEEE Transactions on Visualization and Computer Graphics 3*, 2 (Apr.), 118–128.

Wittenbrink, C. M., Malzbender, T., and Goss, M. E. 1998. Opacity-weighted color interpolation, for volume sampling. In *Proceedings of the 1998 IEEE symposium on Volume visualization*, ACM Press, 135–142.

Woo, M., et al. 1999. *OpenGL programming guide: the official guide to learning OpenGL, version 1.2*, third ed. Addison-Wesley, Reading, MA, USA.