Vision, Modeling, and Visualization (2023)
T. Grosch and M. Guthe (Eds.)

Ray Tracing Spherical Harmonics Glyphs

Christoph Peters[1], Tark Patel[1,2], Will Usher[1] and Chris R. Johnson[2]

[1] Intel Corporation
[2] University of Utah, Scientific Computing & Imaging (SCI) Institute, USA

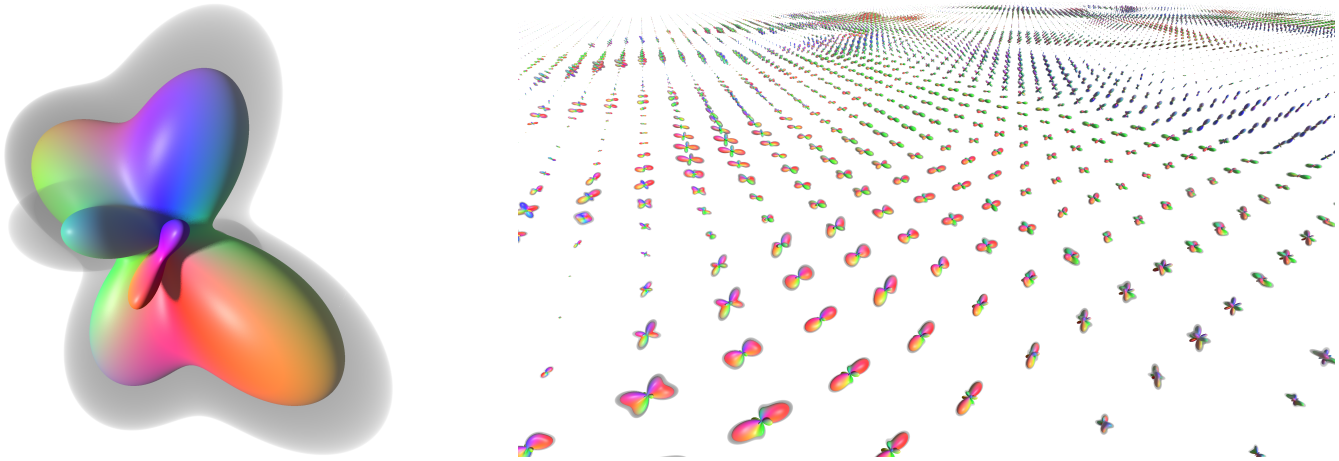


Figure 1: *Left: We render spherical harmonics glyphs using an efficient and accurate method based on polynomial root finding. A second glyph visualizes uncertainty and we trace secondary rays to render soft shadows. Right: Our method renders large HARDI datasets in real time on a single GPU.*

Abstract

Spherical harmonics glyphs are an established way to visualize high angular resolution diffusion imaging data. Starting from a unit sphere, each point on the surface is scaled according to the value of a linear combination of spherical harmonics basis functions. The resulting glyph visualizes an orientation distribution function. We present an efficient method to render these glyphs using ray tracing. Our method constructs a polynomial whose roots correspond to ray-glyph intersections. This polynomial has degree $2k+2$ for spherical harmonics bands $0, 2, \ldots, k$. We then find all intersections in an efficient and numerically stable fashion through polynomial root finding. Our formulation also gives rise to a simple formula for normal vectors of the glyph. Additionally, we compute a nearly exact axis-aligned bounding box to make ray tracing of these glyphs even more efficient. Since our method finds all intersections for arbitrary rays, it lets us perform sophisticated shading and uncertainty visualization. Compared to prior work, it is faster, more flexible and more accurate.

CCS Concepts

• ***Computing methodologies*** → ***Ray tracing;*** • ***Human-centered computing*** → ***Scientific visualization;*** • ***Mathematics of computing*** → *Solvers;*

1. Introduction

Diffusion weighted imaging (DWI) is a magnetic resonance imaging technique for non-invasively measuring the orientation and magnitude of fiber populations that correspond to local diffusion. DWI data are inherently multivariate, because an ensemble of orientation vectors and magnitudes are measured for each voxel. This high dimensionality makes their visualization challenging and many approaches have been put forward. Diffusion tensor imaging (DTI) works with a symmetric 3×3 matrix per voxel, which can be visualized as ellipsoid [WMK*99, BML94]. However, this rep-

resentation is misleading when the distribution of fiber orientations within a voxel is heterogeneous.

High angular resolution diffusion imaging (HARDI) [TRW*02] acquires more data and represents them more faithfully. Many of its variants store a fiber orientation distribution function (ODF) per voxel, encoded using a truncated spherical harmonics (SH) basis [TCGC04, TCC07, ÖSV*06]. These data may be visualized using tractography [BPP*00] but for validation a more direct visualization of the fiber ODFs using one SH glyph per voxel is useful [vPP*11]. An SH glyph is obtained by taking a sphere and scaling each point in proportion to the value of the ODF. Thus, angular maxima of the fiber ODF, which correspond to major fiber directions, are visualized as lobes of the glyph.

One option to visualize SH glyphs is to deform a densely tesselated sphere based on the ODF, e.g. through a compute shader [PDG21]. However, smooth glyphs call for more than thousand triangles [PPv*09, PDG21] and rendering that much geometry for many glyphs takes a lot of computation and bandwidth. Ray casting is compelling because its cost scales with the number of pixels, not with the number of glyphs. Prior work accomplishes that using different forms of ray marching [vPP*11, PPv*09]. The accuracy of these methods depends on the number of ray marching steps such that computing accurate results is quite costly. Additionally, the faster one of these methods [vPP*11] assumes that all rays originate at the camera and does not support secondary rays natively, e.g. for shadows.

We present an efficient method to compute all intersections of an arbitrary ray and an arbitrary SH glyph. To this end, we interpret the SH basis functions as homogeneous polynomials (Sec. 3.1). Then it turns out that all sought-after intersections correspond to roots of a polynomial of degree $2k+2$ for SH bands $0, 2, \ldots, k$ (Sec. 3.2). We compute coefficients of this polynomial in a numerically stable way (Secs. 3.3 and 3.4) and apply an off-the-shelf root-finding method [Yuk22] to solve it (Sec. 3.5).

Our homogeneous formulation of the SH basis also gives rise to a simpler formula for the surface normals of the glyph (Sec. 4.1). That in turn enables us to compute nearly exact axis-aligned bounding boxes (AABBs) for SH glyphs, which we use for hardware-accelerated ray tracing (Sec. 4.2). Unlike prior work [vPP*11], our method computes all intersections for arbitrary rays without precomputation. That enables us to perform efficient volume rendering for uncertainty visualization and to render our glyphs with soft shadows (Sec. 4.3). Our evaluation demonstrates that this method is more accurate, more flexible and faster than prior work [vPP*11] (Sec. 5). At the same time, the intersection test is considerably simpler (Alg. 1). We provide a GLSL implementation of the intersection test and C code for our AABB computation as supplemental.

2. Related Work

The high dimensionality of DWI data makes their visualization challenging and many approaches have been proposed to address the problem. A classic method for representing intravoxel fiber populations is DTI [BML94], which computes a 3×3 symmetric positive-definite tensor for each voxel. The eigenvectors of the tensor encode the main diffusion directions, while the eigenvalues encode the magnitude of diffusion [LBMP*01]. They can be visualized using ellipsoids [WMK*99] or superquadric tensor glyphs [Kin04]. The latter improve the perception of orientation and fractional anisotropy. A major limitation of DTI is that it cannot resolve multiple fiber directions within a voxel, although that is a common case, e.g. in the gray matter of the brain.

HARDI techniques overcome this disadvantage of DTI. They measure far more directional diffusion gradients, sixty to a few hundred, and apply higher-order methods to measure the local water diffusivity more accurately [GJZJ12, JGJJ11]. These data are noisy and have an indirect relationship to fiber orientations. They require further processing before they can be interpreted or visualized. Tuch et al. [TRW*02] assume that voxels contain two fiber directions and fit a mixture of two Gaussians. Q-ball imaging [Tuc04] reconstructs the diffusion ODF using the Funk-Radon transform. The fiber ODF is related to the diffusion ODF through a spherical convolution with the white-matter response function. Once this response has been calibrated based on a region with a single fiber direction, spherical deconvolution computes the fiber ODF [TCGC04]. The diffusion orientation transform [ÖSV*06] offers another way to estimate the fiber ODF.

A truncated SH basis serves as compact representation of the ODFs. Due to symmetry in the data, odd bands of the basis can be omitted. Even bands of the SH basis span the same space as homogeneous polynomials, which are commonly represented by $3 \times \ldots \times 3$ symmetric tensors [ÖM03, DAFD06]. Tikhonov regularization with the Laplace-Beltrami operator makes the coefficients more robust to noise [DAFD06]. Applying a different Tikhonov regularization iteratively to force negative distribution values towards zero allows for robust reconstructions of fiber ODFs up to band $k = 12$ [TCC07]. That corresponds to 91 coefficients per voxel, which need to be visualized in a meaningful way.

One option is to use tractography for fiber tracking [BPP*00]. The resulting visualizations show connections in an intuitive fashion. However, these techniques are prone to ambiguities in the data, susceptible to initialization issues, sensitive to the principal flow direction and computationally expensive. Thus, it is useful to visualize the fiber ODF for a subset of all voxels more directly, at least for validation of tractography results [vPP*11]. The most established approach is to start with a sphere and to scale each point on its surface in proportion to the value of the ODF for this direction vector [HMH*06]. The result is an SH glyph. Schultz and Kindlmann [SK10] explored the application of the higher-order maximum enhancing (HOME) glyphs [SWS09] as alternative to SH glyphs. HOME glyphs convey the ODF's shape and crossings well but their visualization relies on triangulation, which can lead to difficult tradeoffs between accuracy and efficiency of the rendering (see below).

Rendering of SH glyphs has received some attention since they are such an important tool for visualizing HARDI data. The most obvious approach to render them is to deform a densely tesselated sphere, e.g. by using a compute shader to generate vertex buffers [PDG21]. However, insufficient tesselation leads to an incorrect rendition of angular maxima. Smooth and reliable results call for more than a thousand triangles [PPv*09, PDG21], so this

approach does not scale well when rendering many glyphs (Fig. 1). Peeters et al. [PPv*09] instead embrace a ray casting approach. They construct a conservative bounding sphere around the SH glyph and determine the segment of the ray within this sphere. Then they perform ray marching along this segment. Once a point inside the glyph has been found, the intersection position is refined using bisection. They implement this method on the GPU using rasterization to trigger the necessary fragment shader invocations. Their method has also been combined with DTI visualization based on a semi-automatic classification of glyphs [PPv*11].

The inner loop for ray marching in the method of Peeters et al. [PPv*09] is costly, because it has to evaluate the ODF from scratch for every single sample. van Almsick et al. [vPP*11] make this approach more efficient by performing some precomputation outside of this loop. Per camera position, they use Wigner matrices to transform the SH coefficients to a coordinate frame where the camera is on the z-axis. In this coordinate frame, they are able to construct a polynomial of degree k representing the ODF along the ray. Then ray marching samples only evaluate this polynomial and relate its value to the current distance to the glyph center. Instead of bisection, van Almsick et al. rely on regula falsi and they use a bounding cylinder instead of a bounding sphere. This approach yields a considerable speedup but introduces a preprocessing step that depends on the camera position and has only been described in detail for truncation at band $k = 4$. The quality of both of these methods depends chiefly on the number of ray marching samples. Low values lead to low accuracy of intersection points and missed intersections but high values increase the rendering time. Our method is far more accurate, can process any ray, including secondary rays for shadows, without additional preprocessing, supports truncation up to band $k = 10$ and uses a tight AABB.

Jiao et al. [JPGJ12] propose a volume-based technique to visualize uncertainty in HARDI data. They create a small volume for each glyph that encodes the shape inclusion probability (SIP) function for the possible glyph surface locations. Although this approach conveys the uncertainty in fiber ODFs well, computing and storing the SIP volume for each glyph is a compute and memory intensive process. Our proposed ray-glyph intersection method robustly finds all intersections with the glyph along the ray, which enables us to use a more efficient form of volume rendering to convey uncertainty.

3. Ray-Glyph Intersection Test

The core of our method is our novel intersection test, which finds all intersections between an arbitrary ray and an SH glyph. To derive it, we first have to interpret the SH basis as homogeneous polynomials (Sec. 3.1). With this interpretation, we derive polynomials whose roots correspond to ray-glyph intersections (Sec. 3.2). To ensure good numerical stability, we do not solve for the ray parameter of intersections directly but apply a coordinate transform first (Sec. 3.3). Then we figure out how to compute coefficients of the resulting polynomial (Sec. 3.4) and compute roots using an iterative solver (Sec. 3.5). In Sec. 4, we use this intersection test for ray tracing.

3.1. Homogenizing Spherical Harmonics

The SH basis functions are commonly written in terms of spherical coordinates θ, φ. This formulation relies heavily on trigonometric functions, especially sine and cosine. We take a different approach: A point on the unit sphere $\mathbb{S}^2$ is given by its Cartesian coordinates $(x,y,z)^\mathsf{T} \in \mathbb{S}^2$. Then the SH basis functions turn out to be trivariate polynomials in the variables x, y, z [Slo13].

As an example, we consider SH bands 0, 2 and 4. Looking at polynomial expressions for these basis functions [Slo13], we find that they are linear combinations of the following monomials:

Degree 0: 1.

Degree 2: $x^2, xy, xz, y^2, yz, z^2$.

Degree 4: $x^4, x^3y, x^3z, x^2y^2, x^2yz, x^2z^2, xy^3, xy^2z, xyz^2, xz^3, y^4, y^3z, y^2z^2, yz^3, z^4$.

We have grouped these monomials according to their degree, i.e. the sum of all exponents. A polynomial is called homogeneous of degree k if all of its monomials are of degree k. For our next steps, we would like to have a homogeneous basis. Prior work accomplishes that by transitioning to a basis of monomials [ÖM03, DAFD06]. We prefer to use a modified SH basis since monomial basis functions such as x^4 are prone to numerical inaccuracies. As the name indicates, the SH basis functions are spherical. Their values away from the unit sphere are usually irrelevant. Thus, we homogenize the SH basis functions in a way that leaves their values on the unit sphere unchanged but changes the values elsewhere.

Consider the polynomial

$$x^2 + y^2 + z^2.$$

It is homogeneous of degree 2 and its value on the unit sphere is exactly one. If we multiply any monomial by this polynomial, its degree increases by 2 but its values on the unit sphere do not change. Thus, we can multiply each monomial of degree 2 within a polynomial by $x^2 + y^2 + z^2$ without changing its values on the unit sphere. Similarly, we can multiply each monomial of degree 0 by $(x^2 + y^2 + z^2)^2$ without changing its values on the unit sphere. That gives us polynomials, which are homogeneous of degree 4, such as

$$xy(x^2 + y^2 + z^2) = x^3y + xy^3 + xyz^2.$$

Through this procedure, we obtain a version of the SH basis functions in bands 0, 2 and 4, which is homogeneous of degree 4. Then we combine the $1 + 5 + 9 = 15$ basis functions into a single vector-valued function

$$\mathbf{Y}(x,y,z) \in \mathbb{R}^{15}.$$

Appendix A provides explicit expressions for this basis.

The example above uses SH bands 0, 2 and 4 but the underlying idea and our derivations are more general. If we consider all even SH bands up to band k, where k is even, these basis functions can be written as homogeneous polynomials of degree k. Similarly, odd bands up to band k can be written as homogeneous polynomials of degree k. Either way, we get $\binom{k+2}{2} = \frac{(k+1)(k+2)}{2}$ basis functions. However, we cannot mix odd and even bands because then

we would have to work with $\sqrt{x^2+y^2+z^2}$, which is no longer a polynomial. Our implementation supports $k \in \{2,4,\dots,12\}$ but has stability issues for $k \geq 10$ (see Sec. 5.2).

3.2. Intersections as Polynomial Roots

Now we are interested in an SH glyph. The glyph is characterized by SH coefficients $\mathbf{b} \in \mathbb{R}^{\binom{k+2}{2}}$ and its center position. For convenience, we work in a coordinate frame where the center is the origin. Let $\mathbf{p} := (x,y,z)^\mathsf{T} \in \mathbb{R}^3$ be a point on the glyph. With our vector-valued basis function, the corresponding linear combination of SH basis functions is simply

$$\mathbf{b}^\mathsf{T}\mathbf{Y}\left(\frac{\mathbf{p}}{\|\mathbf{p}\|}\right).$$

Then by definition of the glyph

$$\|\mathbf{p}\| = \left|\mathbf{b}^\mathsf{T}\mathbf{Y}\left(\frac{\mathbf{p}}{\|\mathbf{p}\|}\right)\right|. \tag{1}$$

Although $\mathbf{Y}$ consists of polynomials, this is not a polynomial equation. The norm $\|\mathbf{p}\|$ uses a square root and we also have a division and take an absolute value. This is where the homogenization comes in handy. Consider what happens, when we scale the inputs of a monomial $x^i y^j z^{k-i-j}$ of degree k by a factor $\lambda \in \mathbb{R}$:

$$(\lambda x)^i (\lambda y)^j (\lambda z)^{k-i-j} = \lambda^k x^i y^j z^{k-i-j}.$$

The factor λ can be pulled out with an exponent k. Since that is true for each monomial in a homogeneous polynomial, it is also true for the polynomial as a whole.

With this insight, we are ready to turn Eq. 1 into a polynomial equation:

$$\begin{aligned}
&\|\mathbf{p}\| = \left|\mathbf{b}^\mathsf{T}\mathbf{Y}\left(\frac{\mathbf{p}}{\|\mathbf{p}\|}\right)\right| \\
\Leftrightarrow\ &\|\mathbf{p}\|^2 = \left(\mathbf{b}^\mathsf{T}\mathbf{Y}\left(\frac{\mathbf{p}}{\|\mathbf{p}\|}\right)\right)^2 && (2)\\
\Leftrightarrow\ &\|\mathbf{p}\|^2 = \left(\frac{1}{\|\mathbf{p}\|^k}\mathbf{b}^\mathsf{T}\mathbf{Y}(\mathbf{p})\right)^2 \\
\Leftrightarrow\ &\|\mathbf{p}\|^{2k+2} = (\mathbf{b}^\mathsf{T}\mathbf{Y}(\mathbf{p}))^2 \\
\Leftrightarrow\ &(\mathbf{b}^\mathsf{T}\mathbf{Y}(\mathbf{p}))^2 - \|\mathbf{p}\|^{2k+2} = 0 && (3)
\end{aligned}$$

Thanks to the homogeneity of $\mathbf{Y}$, we were able to pull the normalization out of the basis evaluation. In Eq. 3, $(\mathbf{b}^\mathsf{T}\mathbf{Y}(\mathbf{p}))^2$ and $\|\mathbf{p}\|^{2k+2}$ are homogeneous polynomials of degree $2k$ and $2k+2$, respectively (because $\|\mathbf{p}\|$ has an even exponent).[†]

A ray originating at $\mathbf{o} \in \mathbb{R}^3$ and with direction $\mathbf{d} \in \mathbb{S}^2$ passes through points $\mathbf{o}+t\mathbf{d}$ where $t > 0$ is the ray parameter. Thus, the sought-after intersections satisfy

$$(\mathbf{b}^\mathsf{T}\mathbf{Y}(\mathbf{o}+t\mathbf{d}))^2 - \|\mathbf{o}+t\mathbf{d}\|^{2k+2} = 0. \tag{4}$$

Since $\mathbf{o}+t\mathbf{d}$ depends linearly on t and all monomials have degree $2k+2$ or less, this is a polynomial equation in t of degree $2k+2$ or less. For $k=4$, Fig. 2 illustrates that it is actually necessary to work with a polynomial of degree $2k+2=10$ to solve this problem.

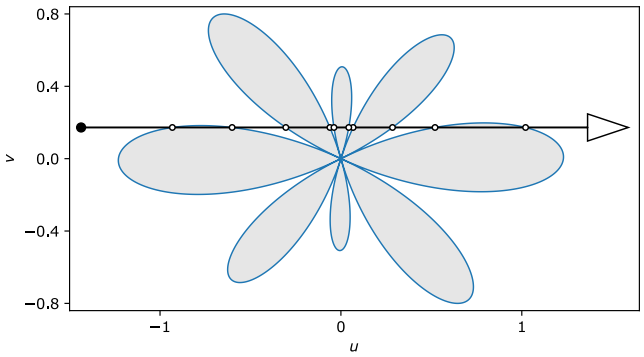


Figure 2: *A cross-section of an SH glyph with $k=4$ and a ray that has 10 distinct intersections with this glyph. Since a polynomial cannot have more real roots than its degree, a degree-10 polynomial is necessary for SH bands 0, 2 and 4. The coordinate frame is as described in Sec. 3.3.*

van Almsick et al. [vPP*11] construct a polynomial that is closely related to $\mathbf{b}^\mathsf{T}\mathbf{Y}(\mathbf{o}+t\mathbf{d})$. However, their construction requires a costly rotation of SH coefficients. Besides, their approach has no counterpart for Eq. 3. In the end, they solve a non-polynomial equation using ray marching.

3.3. Coordinate Transform

To make our approach useful, we have to compute polynomial coefficients of Eq. 4 in some form and then find its roots. It is ill-advised to consider these two problems separately. We could write the polynomial in the basis $1,t,t^2,\dots,t^{2k+2}$ and feed these coefficients to a root finding algorithm but that tends to be numerically unstable: When the ray origin is far from the glyph, all roots will be relatively close to $t = -\mathbf{d}^\mathsf{T}\mathbf{o}$ where the ray comes closest to the glyph center (i.e. $\|\mathbf{o}+t\mathbf{d}\|$ is minimal). Computing many closely spaced roots is difficult in presence of roundoff error (see Fig. 3a and Numerical Recipes chap. 9.5 [PTVF07]).

To overcome this problem, we use a different coordinate frame. It is spanned by the ray direction $\mathbf{d}$ and the orthonormal vector

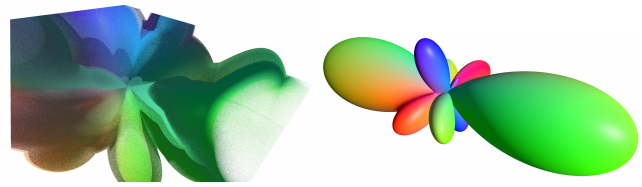
$$\mathbf{e} := \frac{\mathbf{o}-\mathbf{d}^\mathsf{T}\mathbf{o}\mathbf{d}}{\|\mathbf{o}-\mathbf{d}^\mathsf{T}\mathbf{o}\mathbf{d}\|}.$$

These vectors span the plane containing the entire ray and the glyph center at the origin (Fig. 2). The coordinates of a point on the ray in this coordinate frame are

$$\begin{pmatrix}u\\v\end{pmatrix} := \begin{pmatrix}\mathbf{d}^\mathsf{T}(\mathbf{o}+t\mathbf{d})\\ \mathbf{e}^\mathsf{T}(\mathbf{o}+t\mathbf{d})\end{pmatrix} = \begin{pmatrix}\mathbf{d}^\mathsf{T}\mathbf{o}+t\\ \mathbf{e}^\mathsf{T}\mathbf{o}\end{pmatrix}.$$

[†] For odd degree k, it is more efficient to work with two polynomials of half the degree:

$$\|\mathbf{p}\| = \pm\mathbf{b}^\mathsf{T}\mathbf{Y}\left(\frac{\mathbf{p}}{\|\mathbf{p}\|}\right) \quad\Leftrightarrow\quad \|\mathbf{p}\|^{k+1} = \pm\mathbf{b}^\mathsf{T}\mathbf{Y}(\mathbf{p})$$



(a) *Solving for t* **(b)** *Solving for u, v*

Figure 3: *Computing roots in terms of the ray parameter t directly, causes serious numerical instabilities. A different coordinate frame resolves these problems. Both images use single-precision arithmetic.*

Since we have added $\mathbf{d}^\mathsf{T}\mathbf{o}$ to the ray parameter t, the point of closest approach now corresponds to $u = 0$ and the relative differences of roots are larger.

In this coordinate frame, we want to solve the pair of equations

$$v = \mathbf{e}^\mathsf{T}\mathbf{o}, \qquad (\mathbf{b}^\mathsf{T}\mathbf{Y}(u\mathbf{d}+v\mathbf{e}))^2 - \|u\mathbf{d}+v\mathbf{e}\|^{2k+2} = 0.$$

To homogenize this equation, we multiply v^2 onto the left term and $(\mathbf{e}^\mathsf{T}\mathbf{o})^2$ onto the right term:

$$f(u,v) := v^2(\mathbf{b}^\mathsf{T}\mathbf{Y}(u\mathbf{d}+v\mathbf{e}))^2 - (\mathbf{e}^\mathsf{T}\mathbf{o})^2\|u\mathbf{d}+v\mathbf{e}\|^{2k+2} = 0. \quad (5)$$

In this manner, we have retained all original solutions where $v = \mathbf{e}^\mathsf{T}\mathbf{o}$ but $f(u,v)$ is now a homogeneous polynomial of degree $2k+2$. That means we can find coefficients $f_0, \dots, f_{2k+2}$ such that

$$f(u,v) = \sum_{i=0}^{2k+2} f_i u^i v^{2k+2-i}. \quad (6)$$

Constructing the polynomial in this changed coordinate frame gives a much more stable solution (Fig. 3b).

3.4. Computing Polynomial Coefficients

Now that we know what polynomial we want to solve, it is time to compute its coefficients. Among the factors in Eq. 5

$$g(u,v) := \mathbf{b}^\mathsf{T}\mathbf{Y}(u\mathbf{d}+v\mathbf{e})$$

is the most difficult to handle. It is homogeneous of degree k, i.e. given by coefficients $g_0, \dots, g_k \in \mathbb{R}$ via

$$g(u,v) = \sum_{i=0}^{k} g_i u^i v^{k-i}. \quad (7)$$

We can uniquely determine all $k+1$ coefficients if we know the value of this polynomial at $k+1$ different points. For each $l \in \{0, \dots, k\}$, we choose the point

$$u_l := \cos\left(l\frac{\pi}{k+1}\right), \qquad v_l := \sin\left(l\frac{\pi}{k+1}\right).$$

This choice is convenient, because it means that $\|u_l\mathbf{d} + v_l\mathbf{e}\| = 1$. Therefore, we do not actually need a homogenized SH basis $\mathbf{Y}$ for this approach to work. We just have to evaluate the common SH basis at $k+1$ points on the unit sphere to infer its values anywhere along the ray. This simplification is made possible by the homogenization of $f(u,v)$ in Eq. 5.

To obtain the polynomial coefficients, we solve the system of linear equations

$$g(u_l, v_l) = \sum_{i=0}^{k} g_i u_l^i v_l^{k-i}.$$

The solution can be written in terms of a Vandermonde matrix as follows:

$$V := (u_l^i v_l^{k-i})_{l,i=0}^{k} \in \mathbb{R}^{(k+1)\times(k+1)},$$
$$(g_0, \dots, g_k)^\mathsf{T} = V^{-1}(g(u_0,v_0), \dots, g(u_k,v_k))^\mathsf{T} \in \mathbb{R}^{k+1}.$$

The Vandermonde matrix V only depends on the degree k and thus its inverse can be precomputed.

Next, we are interested in coefficients $s_0, \dots, s_{2k} \in \mathbb{R}$ of the square $s(u,v) := g^2(u,v)$. They satisfy

$$s(u,v) = \sum_{i=0}^{2k} s_i u^i v^{2k-i} = \sum_{i=0}^{k}\sum_{j=0}^{k} g_i g_j u^{i+j} v^{2k-i-j}. \quad (8)$$

Thus, a double loop suffices to accumulate all these coefficients. For the polynomial $\|u\mathbf{d}+v\mathbf{e}\|^{2k+2}$, we exploit that $\mathbf{d}, \mathbf{e}$ are orthonormal and use the binomial theorem:

$$\|u\mathbf{d}+v\mathbf{e}\|^{2k+2} = (u^2+v^2)^{k+1} = \sum_{i=0}^{k+1}\binom{k+1}{i} u^{2i} v^{2(k+1-i)}.$$

Now we are ready to assemble the coefficients $f_0, \dots, f_{2k+2} \in \mathbb{R}$:

$$\begin{aligned} f(u,v) = \sum_{i=0}^{2k+2} f_i u^i v^{2k+2-i} &= v^2 s(u,v) - (\mathbf{e}^\mathsf{T}\mathbf{o})^2\|u\mathbf{d}+v\mathbf{e}\|^{2k+2} \\ &= \sum_{i=0}^{2k} s_i u^i v^{2k+2-i} - (\mathbf{e}^\mathsf{T}\mathbf{o})^2 \sum_{i=0}^{k+1}\binom{k+1}{i} u^{2i} v^{2(k+1-i)}. \end{aligned} \quad (9)$$

3.5. Polynomial Root Finding

Now we have reduced the intersection test to polynomial root finding. Most root finding algorithms expect dehomogenized polynomials, so we work with the dehomogenized $f(u) := f(u,1)$. Our goal is to find all real roots of this polynomial. Complex roots are irrelevant, we care about polynomials with degrees ranging from six to 26 and we only need moderate accuracy but want robust behavior in single-precision arithmetic and fast execution on GPUs. The bracketed Newton bisection of Yuksel [Yuk22] matches these requirements well. First it computes all real roots of the derivative of the polynomial. Between these roots, the polynomial is monotonic and a combination of Newton's method with bisection works well. The method operates recursively on derivatives until the quadratic formula is applicable.

This root finder works most efficiently when it is provided with a reasonably tight interval containing relevant roots. We could obtain those using our AABB computation (Sec. 4.2) but it is more convenient to keep these two problems separated. Instead, we recall that

roots (u,v) satisfy Eq. 5 and exploit that $g(u,v)$ is homogeneous:

$$\begin{aligned} & v^2 g^2(u,v) = (\mathbf{e}^\mathsf{T}\mathbf{o})^2 (u^2+v^2)^{k+1} \\ \Leftrightarrow\ & v^2 g^2\left(\frac{(u,v)}{\sqrt{u^2+v^2}}\right) = (\mathbf{e}^\mathsf{T}\mathbf{o})^2 (u^2+v^2) \\ \overset{v=1}{\Leftrightarrow}\ & u^2 = \frac{1}{(\mathbf{e}^\mathsf{T}\mathbf{o})^2} g^2\left(\frac{(u,1)}{\sqrt{u^2+1^2}}\right) - 1. \end{aligned} \tag{10}$$

In the last step, we assume $v=1$ since that matches how we dehomogenized $f(u)$. Thus, we can attain an upper bound for roots u from an upper bound for the polynomial g at points on the unit circle $\mathbb{S}^1$. Appendix B derives the following upper bound:

$$\max_{(u,v)\in\mathbb{S}^1} |g(u,v)| \le \sum_{i=0}^{k} |g_i| \max_{(u,v)\in\mathbb{S}^1} |u^i v^{k-i}| = \sum_{i=0}^{k} |g_i| \sqrt{\frac{i^i (k-i)^{k-i}}{k^k}}.$$

Prior work uses a similar approach for bounding sphere computation [PPv*09] but doing it per ray gives tighter bounds. If we only care about roots within a certain interval, we can tighten the bounds accordingly.

We set the error tolerance for Newton bisection to $\varepsilon := 2 \cdot 10^{-4} u_{\max}$, where $u_{\max} > 0$ is the bound on u derived in Eq. 10. Typically, errors will be much lower than ε [Yuk22]. We also limit the number of iterations of Newton bisection to 90 to avoid endless loops. With single-precision arithmetic and degree $k \le 8$, we sometimes observe numerical instabilities for rays pointing directly at the glyph center. When viewing a single glyph up close at 3840×2160, that may manifest in a few incorrect pixels near the glyph center. In this boundary case, the polynomial $f(u)$ has roots with high multiplicity, which is challenging for the solver [Yuk22]. Though, these artifacts are hard to spot and overall the method works reliably and efficiently. As an alternative for $k=4$, we tried computing all roots, including complex ones, using Laguerre's method (chap. 9.5.3 [PTVF07]). That also works well but is ca. 2.7 times slower since it computes more roots and relies on complex arithmetic heavily. For $k \ge 10$, numerical issues become more serious (see Sec. 5.2).

We have completed our intersection procedure now. Algorithm 1 summarizes all steps.

Algorithm 1 Computes all intersections of an SH glyph and a ray.

Input: SH coefficients $\mathbf{b} \in \mathbb{R}^{\binom{k+2}{2}}$ of degree $k \in \mathbb{N}$, ray origin $\mathbf{o} \in \mathbb{R}^3$ (relative to the glyph center) and normalized ray direction $\mathbf{d} \in \mathbb{S}^2$.

Output: Ray parameters t for all ray-glyph intersections.

$\mathbf{e} := \dfrac{\mathbf{o} - \mathbf{d}^\mathsf{T}\mathbf{o}\mathbf{d}}{\|\mathbf{o} - \mathbf{d}^\mathsf{T}\mathbf{o}\mathbf{d}\|}$.

$u_l := \cos\left(l\dfrac{\pi}{k+1}\right), \quad v_l := \sin\left(l\dfrac{\pi}{k+1}\right)$ for $l \in \{0,\dots,k\}$.

$(g_i)_{i=0}^k := V^{-1}(\mathbf{b}^\mathsf{T}\mathbf{Y}(u_l\mathbf{d} + v_l\mathbf{e}))_{l=0}^k$ where $V := (u_l^i v_l^{k-i})_{l,i=0}^k$.

$g_{\max} := \sum_{i=0}^{k} |g_i| \sqrt{\dfrac{i^i (k-i)^{k-i}}{k^k}}$.

If $|g_{\max}| < |\mathbf{e}^\mathsf{T}\mathbf{o}|$: Return no intersections.

$u_{\max} := \sqrt{\dfrac{g_{\max}^2}{(\mathbf{e}^\mathsf{T}\mathbf{o})^2} - 1}$

Compute $s_0,\dots,s_{2k} \in \mathbb{R}$ using Eq. 8.

Compute $f_0,\dots,f_{2k+2} \in \mathbb{R}$ using Eq. 9.

Compute all roots $u_0,\dots,u_{n-1}$ of $\sum_{i=0}^{2k+2} f_i u^i$ in $[-u_{\max}, u_{\max}]$.

For $l \in \{0,\dots,n-1\}$: Output $t_l := u_l \mathbf{e}^\mathsf{T}\mathbf{o} - \mathbf{d}^\mathsf{T}\mathbf{o}$.

4. Ray Tracing Spherical Harmonics Glyphs

We now have the means to figure out whether and where a ray intersects an SH glyph. To render glyphs, we need proper shading. Therefore, we derive a simple formula to compute surface normals (Sec. 4.1). This formula for the normals also enables us to compute nearly exact AABBs as bounding volumes for ray tracing (Sec. 4.2). Finally, we exploit that our method produces all intersections for arbitrary rays to implement sophisticated shading with shadows and uncertainty visualization (Sec. 4.3).

4.1. Computing Normals

Eq. 2 characterizes our glyph as implicit surface. It is well-known that the gradient of the implicit function at an intersection $\mathbf{p} \in \mathbb{R}^3$ provides a normal vector. We use the chain rule and the quotient rule to compute the non-normalized normal:

$$\begin{aligned} \mathbf{n}^\mathsf{T}(\mathbf{p}) :=& \frac{1}{2}\frac{\partial}{\partial \mathbf{p}} \left(\mathbf{p}^\mathsf{T}\mathbf{p} - \left(\mathbf{b}^\mathsf{T}\mathbf{Y}\left(\frac{\mathbf{p}}{\|\mathbf{p}\|}\right)\right)^2\right) \\ =& \mathbf{p}^\mathsf{T} - \mathbf{b}^\mathsf{T}\mathbf{Y}\left(\frac{\mathbf{p}}{\|\mathbf{p}\|}\right)\mathbf{b}^\mathsf{T}\mathbf{Y}'\left(\frac{\mathbf{p}}{\|\mathbf{p}\|}\right)\frac{\mathbf{p}^\mathsf{T}\mathbf{p}\mathbf{I} - \mathbf{p}\mathbf{p}^\mathsf{T}}{\sqrt{\mathbf{p}^\mathsf{T}\mathbf{p}}^3}, \end{aligned} \tag{11}$$

where $\mathbf{Y}'(\mathbf{p}) \in \mathbb{R}^{\binom{k+2}{2}\times 3}$ denotes the Jacobian matrix of the SH basis $\mathbf{Y}(\mathbf{p})$ and $\mathbf{I} \in \mathbb{R}^{3\times 3}$ is the identity matrix. Since we start from Eq. 2 instead of Eq. 3, the homogenization of the SH basis is irrelevant for this derivation. Therefore, the normal $\mathbf{n}(\mathbf{p})$ will also be correct when $\mathbf{Y}'(\mathbf{p})$ is the Jacobian of an inhomogeneous SH basis. Homogenization only changes components of the derivatives, which are normal to the unit sphere at $\frac{\mathbf{p}}{\|\mathbf{p}\|}$, but the matrix $\mathbf{p}^\mathsf{T}\mathbf{p}\mathbf{I} - \mathbf{p}\mathbf{p}^\mathsf{T} \in \mathbb{R}^{3\times 3}$ eliminates such components. Prior work has similar formulas [PPv*09, vPP*11] that give the same result, though ours is simpler.

4.2. Computing Bounding Boxes

There are two major ways to make rendering of implicit surfaces faster: One can make the ray intersection test itself faster, or one can ensure that it is executed less frequently. We want to render many SH glyphs. Therefore, we enclose every single one of them in an AABB. These AABBs in turn are stored in a bounding volume hierarchy. We only execute the intersection test when a ray traverses the AABB of an SH glyph. Smaller AABBs mean fewer tests and faster rendering. Prior work uses crude upper bounds to construct bounding spheres [PPv*09] or bounding cylinders [vPP*11]. We instead strive to compute nearly exact AABBs.

Without loss of generality, we seek the extent of the AABB along

the z-axis. Otherwise, we permute the axes. Our SH glyphs are point symmetric: If $\mathbf{p} \in \mathbb{R}^3$ is on the glyph, so is $-\mathbf{p}$. Thus, we only care about the absolute value of the extent. Since our glyph is smooth, this maximal extent must be realized by a point $\mathbf{p} \in \mathbb{R}^3$ on the glyph where the AABB is tangential to the glyph, i.e.

$$\|\mathbf{p}\| = \left|\mathbf{b}^\mathsf{T}\mathbf{Y}\left(\frac{\mathbf{p}}{\|\mathbf{p}\|}\right)\right| \quad \text{and} \quad \mathbf{n}_\mathrm{x}(\mathbf{p}) = \mathbf{n}_\mathrm{y}(\mathbf{p}) = 0.$$

Together, these three equations characterize the tangent point $\mathbf{p}$. To make them easier to solve, we eliminate the first equation by using it to homogenize the formula for computation of the normal vector:

$$\tilde{\mathbf{n}}(\mathbf{p}) := \mathbf{b}^\mathsf{T}\mathbf{Y}\left(\frac{\mathbf{p}}{\|\mathbf{p}\|}\right)\frac{\mathbf{p}}{\|\mathbf{p}\|} - \frac{\mathbf{p}^\mathsf{T}\mathbf{p}\mathbf{I} - \mathbf{p}\mathbf{p}^\mathsf{T}}{\|\mathbf{p}\|^2}\mathbf{Y}'^\mathsf{T}\left(\frac{\mathbf{p}}{\|\mathbf{p}\|}\right)\mathbf{b}.$$

For points $\mathbf{p}$ on the glyph, $\mathbf{n}(\mathbf{p}) = \tilde{\mathbf{n}}(\mathbf{p})$, but $\tilde{\mathbf{n}}(\mathbf{p})$ is homogeneous of degree zero: For all non-zero $\lambda \in \mathbb{R}$, $\tilde{\mathbf{n}}(\mathbf{p}) = \tilde{\mathbf{n}}(\lambda\mathbf{p})$. Thus, we discard the first equation and instead keep $\mathbf{p}_\mathrm{z}$ fixed while solving $\tilde{\mathbf{n}}_\mathrm{x}(\mathbf{p}) = \tilde{\mathbf{n}}_\mathrm{y}(\mathbf{p}) = 0$ for $\mathbf{p}_\mathrm{x}, \mathbf{p}_\mathrm{y}$ only. We do so using Newton's method. If $\mathbf{p}_i \in \mathbb{R}^3$ is our current estimate for $\mathbf{p}$, our next estimate is

$$\begin{pmatrix}\mathbf{p}_{i+1,\mathrm{x}}\\ \mathbf{p}_{i+1,\mathrm{y}}\end{pmatrix} := \begin{pmatrix}\mathbf{p}_{i,\mathrm{x}}\\ \mathbf{p}_{i,\mathrm{y}}\end{pmatrix} - \begin{pmatrix}\frac{\partial\tilde{\mathbf{n}}_\mathrm{x}}{\partial\mathbf{p}_\mathrm{x}}(\mathbf{p}_i) & \frac{\partial\tilde{\mathbf{n}}_\mathrm{x}}{\partial\mathbf{p}_\mathrm{y}}(\mathbf{p}_i)\\ \frac{\partial\tilde{\mathbf{n}}_\mathrm{y}}{\partial\mathbf{p}_\mathrm{x}}(\mathbf{p}_i) & \frac{\partial\tilde{\mathbf{n}}_\mathrm{y}}{\partial\mathbf{p}_\mathrm{y}}(\mathbf{p}_i)\end{pmatrix}^{-1}\begin{pmatrix}\tilde{\mathbf{n}}_\mathrm{x}(\mathbf{p}_i)\\ \tilde{\mathbf{n}}_\mathrm{y}(\mathbf{p}_i)\end{pmatrix}.$$

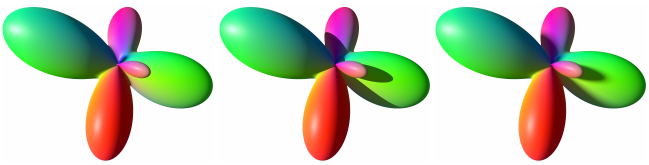
The complete formulas for these partial derivatives of $\tilde{\mathbf{n}}(\mathbf{p})$ can be found in our supplemental code. They use the Hessian matrix of $\mathbf{b}^\mathsf{T}\mathbf{Y}(\mathbf{p})$, which is about as easy to compute as the Jacobian matrix. Upon convergence, the z-coordinate of the point on the glyph $\mathbf{b}^\mathsf{T}\mathbf{Y}\left(\frac{\mathbf{p}_i}{\|\mathbf{p}_i\|}\right)\frac{\mathbf{p}_i}{\|\mathbf{p}_i\|}$ is a candidate for the sought-after extent.

If the initialization to Newton's method is sufficiently close to a solution, the method converges quadratically (chap. 9.4 [PTVF07]). However, it will not always converge and not every solution is the global maximum that we seek. We overcome this problem with an extensive search for good initializations. For degree k, we take $N := k \cdot 256$ spherical Fibonacci points in the upper hemisphere [KISS15], i.e. point $l \in \{0, \dots, N-1\}$ is given by

$$z_l := 1 - \frac{2l+1}{2N}, \quad \varphi_l := \frac{4\pi l}{\sqrt{5}+1}, \quad \mathbf{p}_l := \begin{pmatrix}\sqrt{1-z_l^2}\cos\varphi_l\\ \sqrt{1-z_l^2}\sin\varphi_l\\ z_l\end{pmatrix}.$$

For each of these direction vectors, we compute the corresponding point on the glyph $\mathbf{b}^\mathsf{T}\mathbf{Y}(\mathbf{p}_l)\mathbf{p}_l$. If it has the largest absolute z-coordinate encountered thus far, we remember the point. In the end, the best candidate becomes the initialization to Newton's method, which we run for exactly five iterations. If it leads to a point on the glyph with an even larger absolute z-coordinate, we use that. Otherwise, we stick to the one found previously.

We run this method for all three axes in a single pass. Since it is not guaranteed to find a large enough AABB, we subdued it to extensive testing for degrees $k \in \{2, 4, \dots, 12\}$. Per degree, we generated 100,000 vectors $\mathbf{b} \in [-1, 1]^{15}$ uniformly at random and computed an AABB. Then we compared this AABB to one found by sampling at a million points. Very seldomly, our AABB was smaller than the ground truth but never by more than 1.9% for $k \leq 8$ or 3.07% for $k \geq 10$. Consequently, we use our method for computation of a tight AABB and then scale it up by 2.5% for $k \leq 8$ or 3.5% for $k \geq 10$ to be reasonably sure that we never miss an intersection.



(a) *No shadows* **(b)** *Hard shadows* **(c)** *Soft shadows*

Figure 4: *Shadows make it easier to understand spatial relations between different parts of the glyph. Soft shadows provide additional cues about distances and are less distracting than perfectly hard shadows.*

4.3. Shading and Uncertainty Visualization

To complete our method, we need to implement shading for the glyphs. We compute a base color for each point on the glyph by interpreting the absolute entries of $\frac{\mathbf{p}}{\|\mathbf{p}\|}$ as sRGB triple [vPP*11]. Then we utilize the dielectric glTF BRDF with roughness 0.45. Additionally, we use constant ambient lighting to avoid pitch black surfaces.

Unlike prior work [vPP*11], our intersection test naturally supports arbitrary ray origins. Thus, we can trace secondary rays to render self-shadowing of a glyph. Fig. 4 shows how shadows aid the understanding of spatial relationships. In Fig. 4b, we trace a single deterministic shadow ray towards a directional light. The resulting hard shadows are somewhat distracting and fail to convey the distance of shadow casters. Fig. 4c resolves this problem using soft shadows. The size of their smooth penumbra regions provides visual cues about the magnitude of angles between fiber populations. We render them using Monte Carlo integration with 16 random shadow rays towards a spherical light. To avoid noise in places where the light is partially below the horizon, we sample the spherical light proportional to the cosine term from the rendering equation [PD19].

In presence of uncertainty in the data, each SH coefficient is associated with a standard deviation. Jiao et al. [JPGJ12] convey this uncertainty using volume rendering. This approach is powerful but also has considerable overhead. We propose a more efficient alternative. We assume that the region of uncertainty can be approximated by a second SH glyph. Then we visualize this region using an efficient form of volume rendering (Fig. 5). We compute all intersections up to the first opaque surface for the second glyph. From the intersections we can compute the distance $d \geq 0$ that the ray travels through the glyph. Then we treat it as homogeneous volume with constant emission. The transmittance is $T := \exp(-\sigma d)$ where $\sigma > 0$ is a user-defined extinction parameter. Finally, we use alpha blending with opacity $\alpha := 1 - T$ to blend the current color with a constant emission color (we use the sRGB triple $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$).

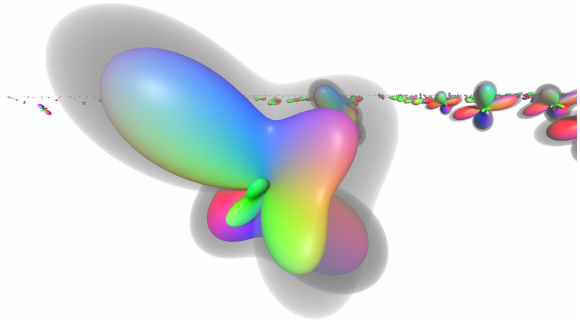
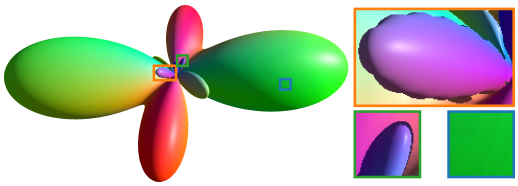
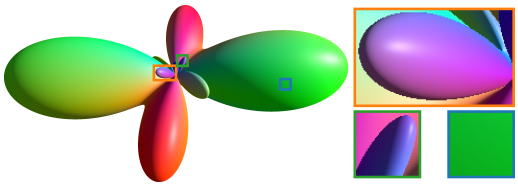


Figure 5: *We visualize uncertainty by rendering two SH glyphs. An opaque glyph visualizes the mean. The other glyph is rendered as homogeneous volume with constant absorption and gray emission to visualize uncertainty.*



(a) *Ray marching with 100 steps [vPP*11]*



(b) *Our method*

Figure 6: *Ray marching with 100 steps [vPP*11] resolves small features poorly, leads to incorrect normals at silhouettes and causes banding. Our method is far more accurate.*

5. Results

We now present additional results of our method, compare it to prior work in terms of quality (Sec. 5.1), validate it for truncation up to band $k = 12$ (Sec. 5.2) and discuss its run time (Sec. 5.3). Our evaluation uses a diffusion dataset [SCAD22] with 60 diffusion gradient directions, which is an undersampling of the humane connectome project of aging dataset performed on 3T MR scanners [HSA*18]. For postprocessing, we use MRtrix3[‡] with spherical deconvolution [TCGC04] to compute SH coefficients. All figures except Fig. 2 and Fig. 7 show glyphs from this dataset using SH bands 0, 2 and 4.

We were unable to obtain data with known uncertainty. To evaluate this part of our rendering method, we generate a second set of SH coefficients from the first set: The SH coefficient for the constant component Y_0^0 is multiplied by 1.7 such that positive parts of the glyph grow. Additionally, each SH coefficient is contaminated with noise by multiplying it by a Gaussian random variable with mean 1 and standard deviation 0.05.

We have implemented our method on GPUs using Vulkan and the VK_KHR_ray_query extension in a compute shader. Of course, rasterizing the AABBs is a viable alternative. Additionally, we implemented the ray marching method of van Almsick et al. [vPP*11] for $k = 4$ in the same framework. The implementation follows the description in the paper closely. However, we use ray tracing instead of rasterization and need an AABB. To this end, we use either our tight AABBs or construct an AABB enclosing a bounding cylinder around the z-axis with the method of van Almsick et al. [vPP*11]. Either way, the shader performs an early out when there is no intersection with the camera-aligned bounding cylinder.

5.1. Quality Comparisons

van Almsick et al. rely on ray marching and propose to partition the ray segment in the cylinder into ca. 100 segments [vPP*11]. Fig. 6 demonstrates that this parameter choice leads to inaccurate results. Most notably, smaller lobes of the glyph suffer from undersampling. They appear smaller than they should be and have jagged silhouettes (orange inset). Additionally, incorrect hit points lead to incorrect normals. That results in black outlines at silhouettes (green inset). Also note that smooth surfaces suffer from slight banding artifacts (blue inset). Our method is more accurate and has none of these artifacts. Using 200 ray marching steps, the artifacts become negligible.

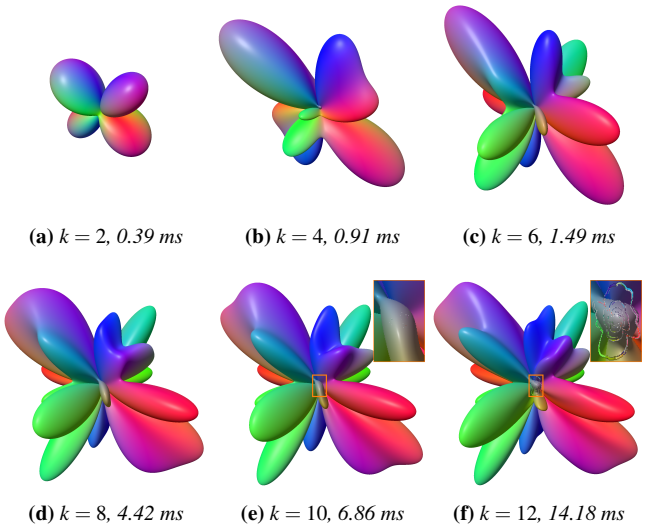
It is also noteworthy that van Almsick et al. rely on a preprocessing step that depends on the camera position. They assume that all rays originate at the camera, which makes sense with rasterization. In our ray tracing framework, it is compelling to use secondary rays, which clashes with this preprocessing approach. Our method naturally supports such secondary rays and we use them for soft shadows (Fig. 4).

Fig. 1 shows an overview of one slice of our test dataset with 19,600 glyphs. Every single glyph is rendered with full accuracy. The cost of such a rendering scales primarily with the number of pixels that are covered by a glyph. The only computation that is done per glyph is our AABB computation (Sec. 4.2) and construction of a bounding volume hierarchy. An approach based on triangulated glyphs would have to process upwards of thousand triangles per glyph in each frame to get smooth results [PDG21].

5.2. Higher Bands

We derived our method to support glyphs of arbitrary degree $k \in 2\mathbb{N}$. Fig. 7 shows results with truncation at bands up to and including band 12. For $k \leq 8$ the renderings are accurate. At $k = 10$, there are minor numerical issues resulting in a few incorrect pixels (magnified inset). At $k = 12$, these numerical issues become more severe. Most of the glyph is still rendered accurately but the artifacts are quite distracting. We expect that they could be alleviated using double-precision arithmetic. Note that we could not generate meaningful data for $k \geq 10$ from our dataset with 60 diffusion gradient

[‡] https://github.com/jdtournier/mrtrix3



(a) $k = 2$, *0.39 ms* **(b)** $k = 4$, *0.91 ms* **(c)** $k = 6$, *1.49 ms*

(d) $k = 8$, *4.42 ms* **(e)** $k = 10$, *6.86 ms* **(f)** $k = 12$, *14.18 ms*

Figure 7: *Renderings of a single glyph at resolution* 1440^2 *using truncation at different bands k. Using more bands results in more detailed shapes while increasing the frame time.*

directions [TCGC04] as we do not use super resolution [TCC07]. The coefficients for band 10 and 12 are not measured but synthesized using zero-mean Gaussian noise so that we can still demonstrate our rendering algorithm.

5.3. Run Time

We measure timings of our GPU implementation on a system with an Intel Arc A770 GPU with 16 GB VRAM running Windows. Compared to the method of van Almsick et al. for $k = 4$, our method has three key differences that are relevant for timings: We use tighter AABBs, we do not have a preprocessing step to rotate SH coefficients and we do not rely on ray marching.

To analyze the impact of the AABBs, we try all techniques using either our tight AABB or an AABB enclosing a bounding cylinder around the z-axis [vPP*11]. Our implementation of the method of van Almsick et al. does not actually use a preprocess. Instead, we rotate the SH coefficients as part of the intersection test. That makes it as flexible as our method but more costly. To produce data that are more representative of the original approach, we also measure timings where we simply skip this step of the algorithm. Results are incorrect but the timings are representative of the original method with camera-specific preprocessing. Finally, we vary the maximal number of ray marching steps. At 100 steps there are clear artifacts (Fig. 6), at 200 steps the quality is closer to our method.

The timings in Tab. 1 show that our method outperforms ray marching. Ray marching with 200 steps and per-ray rotation with the bounding cylinder has previously been the method of choice for something that is as flexible and nearly as accurate as our approach. Our method with our AABB is 3.8 times faster. The cost of rotating SH coefficients relative to the cost of ray marching is not as big as one might expect. With identical bounding volumes, only

Table 1: *Frame times for ray tracing the scene in Fig. 1 at a resolution of* 1920×1080 *using different methods. We either use a bounding cylinder around the z-axis as described by van Almsick et al. [vPP*11] or our tight AABBs.*

	Method	Time
Cylinder	Ray marching (100 steps) [vPP*11]	1.55 ms
	Ray marching (no rotation, 100 steps) [vPP*11]	1.13 ms
	Ray marching (200 steps) [vPP*11]	2.23 ms
	Ray marching (no rotation, 200 steps) [vPP*11]	1.80 ms
	Ours	0.95 ms
Our AABB	Ray marching (100 steps) [vPP*11]	0.73 ms
	Ray marching (no rotation, 100 steps) [vPP*11]	0.59 ms
	Ray marching (200 steps) [vPP*11]	1.07 ms
	Ray marching (no rotation, 200 steps) [vPP*11]	0.89 ms
	Ours	0.59 ms
	Ours with uncertainty	1.93 ms
	Ours with uncertainty and hard shadows	2.00 ms
	Ours with uncertainty and soft shadows	4.02 ms

100 steps and without rotation, ray marching performs similar to our method but is less accurate, less flexible and more complicated to implement.

Adding our uncertainty visualization more than triples the cost, presumably because the uncertainty glyphs cover more pixels than the opaque glyphs. The overhead of a single shadow ray for hard shadows is low but soft shadows with 16 rays double the cost again. Nonetheless, a frame time of 4 ms is still clearly fast enough for real-time rendering.

Fig. 7 reports total frame times for rendering a single glyph using our method with truncation at different bands k. For these experiments, we simply execute the intersection test for each pixel. The timings scale roughly in proportion to k^3, which is to be expected based on the nesting of loops in the method for polynomial root finding. The choice $k = 8$ is common in practical applications. With a frame time of 4.42 ms, it is clearly fast enough for real-time rendering and there is still headroom to add soft shadows or uncertainty visualization. Even at $k = 12$ our method gives real-time frame rates (albeit with some artifacts).

Our method for AABB computation is GPU friendly but for simplicity we wrote a single-threaded implementation in C. We measure timings using an Intel Core i7-11700KF and 64 GB RAM. The AABB computation takes 29 microseconds per glyph for $k = 4$. Thus, the data set in Fig. 1 with 19,600 glyphs including uncertainty is processed in a bit more than a second. The bulk of the work is the evaluation of the SH basis in the extensive search for an initialization. Therefore, we expect this cost to scale linearly with the number of basis function evaluations, which is $\binom{k+2}{2}N$.

6. Conclusions

Our intersection test is a simple self-contained algorithm without preprocessing. It reduces the problem to its core, namely polynomial root finding, where excellent solution strategies are available. The technique is accurate, numerically stable and is guaranteed to

find all intersections. We hope our work will help to make high-quality SH glyphs more widely available as a visualization tool for HARDI or other data. Furthermore, we provide a foundation for developing interactive uncertainty visualization techniques through a combination of surface and volume rendering. All of these contributions are enabled by homogenization of the SH basis and we hope that others may find this method useful for future work.

Appendix A: The Homogeneous Spherical Harmonics Basis

For reference, we provide the version of the SH basis, which has been used for $k = 4$ in this paper. It follows the conventions of Descouteaux et al. [DAFD07], which differ from the ones of Sloan [Slo13]. The homogenized version, which is used for derivations but not needed in an implementation, uses $r := x^2 + y^2 + z^2$. Otherwise, it is valid to set $r := 1$. The basis functions are:

$$Y_0^0 := \frac{r^2}{2\sqrt{\pi}} \qquad Y_2^{-2} := \sqrt{\frac{15}{16\pi}}(x^2 - y^2)r$$

$$Y_2^{-1} := \sqrt{\frac{15}{4\pi}}xzr \qquad Y_2^0 := \sqrt{\frac{5}{16\pi}}(3z^2 - r)r$$

$$Y_2^1 := -\sqrt{\frac{15}{4\pi}}yzr \qquad Y_2^2 := \sqrt{\frac{15}{4\pi}}xyr$$

$$Y_4^{-4} := \frac{3}{16}\sqrt{\frac{35}{\pi}}(x^4 - 6x^2y^2 + y^4) \qquad Y_4^{-3} := \frac{3}{8}\sqrt{\frac{70}{\pi}}x(x^2 - 3y^2)z$$

$$Y_4^{-2} := \frac{3}{8}\sqrt{\frac{5}{\pi}}(x^2 - y^2)(7z^2 - r) \qquad Y_4^{-1} := \frac{3}{8}\sqrt{\frac{10}{\pi}}xz(7z^2 - 3r)$$

$$Y_4^0 := \frac{3}{16\sqrt{\pi}}(35z^4 - 30z^2r + 3r^2)$$

$$Y_4^1 := -\frac{3}{8}\sqrt{\frac{10}{\pi}}yz(7z^2 - 3r) \qquad Y_4^2 := \frac{3}{4}\sqrt{\frac{5}{\pi}}xy(7z^2 - r)$$

$$Y_4^3 := \frac{3}{8}\sqrt{\frac{70}{\pi}}(y^2 - 3x^2)yz \qquad Y_4^4 := \frac{3}{4}\sqrt{\frac{35}{\pi}}xy(x^2 - y^2)$$

For our basis $\mathbf{Y}(x, y, z) \in \mathbb{R}^{15}$, we stack all bands into a single vector in the order in which they are listed above. To evaluate the SH basis and its first and second derivatives, we have used a code generator much like the one described by Sloan [Slo13]. Evaluation code up to band 12 can be found in our supplemental.

Appendix B: Computing Bounding Circles

For even $k \in \mathbb{N}$, we seek the maximum

$$\max_{(u,v)\in\mathbb{S}^1} |u^i v^{k-i}| = \max_{(u,v)\in\mathbb{R}^2\setminus\{0\}} \left|\frac{u^i v^{k-i}}{(u^2+v^2)^{\frac{k}{2}}}\right| = \sup_{u\in\mathbb{R}} \left|\frac{u^i}{(u^2+1)^{\frac{k}{2}}}\right|.$$

We use the quotient rule to identify real critical points:

$$\begin{aligned}
&\frac{\partial}{\partial u}\frac{u^i}{(u^2+1)^{\frac{k}{2}}} = 0 \\
\Leftrightarrow\ & iu^{i-1}(u^2+1)^{\frac{k}{2}} = ku^{i+1}(u^2+1)^{\frac{k}{2}-1} \\
\Leftrightarrow\ & u^{i-1} = 0 \quad \vee \quad i(u^2+1) = ku^2 \\
\Leftrightarrow\ & u^{i-1} = 0 \quad \vee \quad u = \pm\sqrt{\frac{i}{k-i}}
\end{aligned}$$

The value at the second candidate for a critical point is

$$\frac{\sqrt{\frac{i}{k-i}}^i}{\left(\frac{i}{k-i}+1\right)^{\frac{k}{2}}} = \sqrt{\frac{\left(\frac{i}{k-i}\right)^i}{\left(\frac{k}{k-i}\right)^k}} = \sqrt{\frac{i^i(k-i)^{k-i}}{k^k}}.$$

It is easy to verify that this value is always greater or equal to the values for $u = 0$, $u \to \infty$ and $u \to -\infty$, which are 0 or 1 depending on i. Thus, this critical point constitutes the global maximum.

References

[BML94] BASSER P., MATTIELLO J., LEBIHAN D.: MR diffusion tensor spectroscopy and imaging. *Biophysical Journal 66*, 1 (1994). doi:10.1016/S0006-3495(94)80775-1. 1, 2

[BPP*00] BASSER P. J., PAJEVIC S., PIERPAOLI C., DUDA J., ALDROUBI A.: In vivo fiber tractography using DT-MRI data. *Magnetic Resonance in Medicine 44*, 4 (2000). doi:10.1002/1522-2594(200010)44:4<625::AID-MRM17>3.0.CO;2-O. 2

[DAFD06] DESCOTEAUX M., ANGELINO E., FITZGIBBONS S., DERICHE R.: Apparent diffusion coefficients from high angular resolution diffusion imaging: Estimation and applications. *Magnetic Resonance in Medicine 56*, 2 (2006). doi:10.1002/mrm.20948. 2, 3

[DAFD07] DESCOTEAUX M., ANGELINO E., FITZGIBBONS S., DERICHE R.: Regularized, fast, and robust analytical q-ball imaging. *Magnetic Resonance in Medicine 58*, 3 (2007). doi:https://doi.org/10.1002/mrm.21277. 10

[GJZJ12] GUR Y., JIAO F., ZHU S., JOHNSON C.: White matter structure assessment from reduced HARDI data using low-rank polynomial approximations. In *Proceedings of MICCAI 2012 Workshop on Computational Diffusion MRI (CDMRI12)* (2012), Lecture Notes in Computer Science (LNCS). URL: https://pubmed.ncbi.nlm.nih.gov/24818174. 2

[HMH*06] HESS C. P., MUKHERJEE P., HAN E. T., XU D., VIGNERON D. B.: Q-ball reconstruction of multimodal fiber orientations using the spherical harmonic basis. *Magnetic Resonance in Medicine 56*, 1 (2006). doi:10.1002/mrm.20931. 2

[HSA*18] HARMS M. P., SOMERVILLE L. H., ANCES B. M., ANDERSSON J., BARCH D. M., BASTIANI M., BOOKHEIMER S. Y., BROWN T. B., BUCKNER R. L., BURGESS G. C., COALSON T. S., CHAPPELL M. A., DAPRETTO M., DOUAUD G., FISCHL B., GLASSER M. F., GREVE D. N., HODGE C., JAMISON K. W., JBABDI S., KANDALA S., LI X., MAIR R. W., MANGIA S., MARCUS D., MASCALI D., MOELLER S., NICHOLS T. E., ROBINSON E. C., SALAT D. H., SMITH S. M., SOTIROPOULOS S. N., TERPSTRA M., THOMAS K. M., TISDALL M. D., UGURBIL K., VAN DER KOUWE A., WOODS R. P., ZÖLLEI L., VAN ESSEN D. C., YACOUB E.: Extending the human connectome project across ages: Imaging protocols for the lifespan development and aging projects. *NeuroImage 183* (2018). doi:10.1016/j.neuroimage.2018.09.060. 8

[JGJJ11] JIAO F., GUR Y., JOHNSON C., JOSHI S.: Detection of crossing white matter fibers with high-order tensors and rank-k decompositions. In *Proceedings of the International Conference on Information Processing in Medical Imaging (IPMI 2011)* (2011), vol. 6801 of *Lecture Notes in Computer Science (LNCS)*. doi:10.1007/978-3-642-22092-0_44. 2

[JPGJ12] JIAO F., PHILLIPS J. M., GUR Y., JOHNSON C. R.: Uncertainty visualization in HARDI based on ensembles of ODFs. In *2012 IEEE Pacific Visualization Symposium* (2012), IEEE. doi:10.1109/PacificVis.2012.6183591. 3, 7

[Kin04] KINDLMANN G.: Superquadric tensor glyphs. *Eurographics / IEEE VGTC Symposium on Visualization* (2004). doi:10.2312/VISSYM/VISSYM04/147-154. 2

[KISS15] KEINERT B., INNMANN M., SÄNGER M., STAMMINGER M.: Spherical Fibonacci mapping. *ACM Trans. Graph. 34*, 6 (2015). doi:10.1145/2816795.2818131. 7

[LBMP*01] LE BIHAN D., MANGIN J.-F., POUPON C., CLARK C. A., PAPPATA S., MOLKO N., CHABRIAT H.: Diffusion tensor imaging: concepts and applications. *Journal of Magnetic Resonance Imaging: An Official Journal of the International Society for Magnetic Resonance in Medicine 13*, 4 (2001). 2

[ÖM03] ÖZARSLAN E., MARECI T. H.: Generalized diffusion tensor imaging and analytical relationships between diffusion tensor imaging and high angular resolution diffusion imaging. *Magnetic Resonance in Medicine 50*, 5 (2003). doi:10.1002/mrm.10596. 2, 3

[ÖSV*06] ÖZARSLAN E., SHEPHERD T. M., VEMURI B. C., BLACKBAND S. J., MARECI T. H.: Resolution of complex tissue microarchitecture using the diffusion orientation transform (DOT). *NeuroImage 31*, 3 (2006). doi:10.1016/j.neuroimage.2006.01.024. 2

[PD19] PETERS C., DACHSBACHER C.: Sampling projected spherical caps in real time. *Proc. ACM Comput. Graph. Interact. Tech. 2*, 1 (2019). doi:10.1145/3320282. 7

[PDG21] POIRIER C., DESCOTEAUX M., GILET G.: Accelerating geometry-based spherical harmonics glyphs rendering for dMRI using modern OpenGL. In *Computational Diffusion MRI* (2021), Cetin-Karayumak S., Christiaens D., Figini M., Guevara P., Gyori N., Nath V., Pieciak T., (Eds.), Springer International Publishing. doi:10.1007/978-3-030-87615-9_13. 2, 8

[PPv*09] PEETERS T., PRCKOVSKA V., VAN ALMSICK M., VILANOVA A., TER HAAR ROMENY B.: Fast and sleek glyph rendering for interactive HARDI data exploration. In *2009 IEEE Pacific Visualization Symposium* (2009), IEEE. doi:10.1109/PACIFICVIS.2009.4906851. 2, 3, 6

[PPv*11] PRCKOVSKA V., PEETERS T., VAN ALMSICK M., TER HAAR ROMENY B., VILANOVA I BARTROLI A.: Fused DTI/HARDI visualization. *IEEE TVCG 17*, 10 (2011). doi:10.1109/TVCG.2010.244. 3

[PTVF07] PRESS W. H., TEUKOLSKY S. A., VETTERLING W. T., FLANNERY B. P.: *Numerical recipes: The art of scientific computing (third edition)*. Cambridge University Press, 2007. 4, 6, 7

[SCAD22] SEYYEDKAZEM H., CHEN R.-R., ADLURU G., DIBELLA E. V. R.: Jointly estimating parametric maps of multiple diffusion models from undersampled q-space data: A comparison of three deep learning approaches. *Magnetic Resonance in Medicine 87*, 6 (2022). doi:10.1002/mrm.29162. 8

[SK10] SCHULTZ T., KINDLMANN G.: A maximum enhancing higher-order tensor glyph. *Computer Graphics Forum 29* (2010). doi:10.1111/j.1467-8659.2009.01675.x. 2

[Slo13] SLOAN P.-P.: Efficient spherical harmonic evaluation. *Journal of Computer Graphics Techniques (JCGT) 2*, 2 (2013). URL: http://jcgt.org/published/0002/02/06/. 3, 10

[SWS09] SCHULTZ T., WEICKERT J., SEIDEL H.-P.: A higher-order structure tensor. In *Visualization and Processing of Tensor Fields*. Springer, 2009. doi:10.1007/978-3-540-88378-4_13. 2

[TCC07] TOURNIER J.-D., CALAMANTE F., CONNELLY A.: Robust determination of the fibre orientation distribution in diffusion MRI: Non-negativity constrained super-resolved spherical deconvolution. *NeuroImage 35*, 4 (2007). doi:10.1016/j.neuroimage.2007.02.016. 2, 9

[TCGC04] TOURNIER J.-D., CALAMANTE F., GADIAN D. G., CONNELLY A.: Direct estimation of the fiber orientation density function from diffusion-weighted MRI data using spherical deconvolution. *NeuroImage 23*, 3 (2004). doi:10.1016/j.neuroimage.2004.07.037. 2, 8, 9

[TRW*02] TUCH D., REESE T., WIEGELL M., MAKRIS N., BELLIVEAU J., WEDEEN V.: High angular resolution diffusion imaging reveals intravoxel white matter fiber heterogeneity. *Magnetic resonance in medicine: official journal of the Society of Magnetic Resonance in Medicine / Society of Magnetic Resonance in Medicine 48*, 4 (2002). doi:10.1002/mrm.10268. 2

[Tuc04] TUCH D. S.: Q-ball imaging. *Magnetic Resonance in Medicine 52*, 6 (2004). doi:10.1002/mrm.20279. 2

[vPP*11] VAN ALMSICK M., PEETERS T., PRCKOVSKA V., VILANOVA A., TER HAAR ROMENY B.: GPU-based ray-casting of spherical functions applied to high angular resolution diffusion imaging. *IEEE TVCG 17*, 5 (2011). doi:10.1109/TVCG.2010.61. 2, 3, 4, 6, 7, 8, 9

[WMK*99] WESTIN C. F., MAIER S. E., KHIDHIR B., EVERETT P., JOLESZ F. A., KIKINIS R.: Image processing for diffusion tensor magnetic resonance imaging. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI'99* (1999), Taylor C., Colchester A., (Eds.), Springer. 1, 2

[Yuk22] YUKSEL C.: High-performance polynomial root finding for graphics. *Proc. ACM Comput. Graph. Interact. Tech. 5*, 3 (2022). doi:10.1145/3543865. 2, 5, 6