GPU-Based Ray-Casting of Spherical Functions Applied to High Angular Resolution Diffusion Imaging

Markus van Almsick, Tim H.J.M. Peeters, Vesna Prčkovska, Anna Vilanova, and Bart ter Haar Romeny

Abstract—Any sufficiently smooth, positive, real-valued function $\psi : S^2 \rightarrow \mathbb{R}^+$ on a sphere S^2 can be expanded by a Laplace expansion into a sum of spherical harmonics. Given the Laplace expansion coefficients, we provide a CPU and GPU-based algorithm that renders the radial graph of ψ in a fast and efficient way by ray-casting the glyph of ψ in the fragment shader of a GPU. The proposed rendering algorithm has proven highly useful in the visualization of high angular resolution diffusion imaging (HARDI) data. Our implementation of the rendering algorithm can display simultaneously thousands of glyphs depicting the local diffusivity of water. The rendering is fast enough to allow for interactive manipulation of large HARDI data sets.

Index Terms—Computer graphics, viewing algorithms, three-dimensional graphics and realism, computer applications, life and medical sciences.

1 Introduction

The rendering algorithm introduced in this paper implements a new primitive for geometric modeling and visualization. The new primitive can present any smooth, positive, real-valued function ψ that is defined on an ordinary sphere S^2.

$$\psi : S^2 \rightarrow \mathbb{R}^+. \tag{1}$$

We refer to these functions as *spherical functions*. One can visualize a spherical function by a deformed sphere, where the extent of the radial in- or extrusion is given by ψ. The parameterization of the spherical function domain S^2 is usually given by spherical angles $\vartheta \in [0, \pi]$ and $\varphi \in [0, 2\pi)$. Hence, the shape or so-called glyph of ψ is given by the radial graph defined by $r = \psi(\vartheta, \varphi)$.

Any sufficiently smooth spherical function ψ can be expanded in a set of orthonormal, spherical basis functions. The default choice is the orthonormal set of spherical harmonics Y_l^m. Such an expansion is called a Laplace expansion. We assume that the function ψ can be given or can be approximated by a finite Laplace expansion. Our goal is to render a spherical function in such an efficient way, that hundreds or even thousands of these glyphs can be displayed at once in real time.

There are basically two techniques to achieve this in a standard graphics pipeline. For each glyph of a spherical function, one can either generate a vertex geometry of a sphere via a nth-order tessellation and then in- or extrude the resulting vertices according to ψ. Or, alternatively, one can apply a ray-casting algorithm to the spherical function and determine the shading of each pixel in a simple polygon which covers the glyphs projection toward the viewport.

We opt for the second approach and outline an efficient ray-casting algorithm for spherical functions in the following sections.

The purpose of this endeavor has been a visualization task in high angular resolution diffusion imaging (HARDI), a magnetic resonance imaging (MRI) technique, that provides a unique view of the fiber structure of white brain matter in vivo. HARDI is an extension of the better known diffusion tensor imaging (DTI), a technique introduced in its current form in 1994 by Basser et al. [1]. In DTI, one determines symmetric, positive-definite diffusion tensors (DT) from a minimum of seven diffusion weighted images. This gives a second order approximation of the probability density functions (PDF) which describes the local diffusivity of the water molecules by a Gaussian distribution. One of the most important application of DTI is the prediction of fiber orientation, which is specified by the principal eigenvector of the DT. The local fiber orientations determine the trajectories of fiber bundles and consequently the neuronal connections between regions in the brain.

Due to the crude assumption that a simple 3D Gaussian probability density function can capture the Green's function of a diffusion process [2], DTI is limited. It can only recover structures with at most one direction per voxel. In areas with more complex intravoxel heterogeneity (i.e., fiber crossing, kissing, divergence, etc.), the second order DT approximation fails, which is a severe limitation particularly when tracking the path of a fiber bundle.

To overcome the limitation of DTI, HARDI [3] has been introduced. In HARDI, about sixty to a few hundred diffusion gradients are scanned that together sample a sphere of given radius [4] in order to better reconstruct a

• *The authors are with the Department of Biomedical Engineering, Biomedical Image Analysis, Eindhoven University of Technology, Den Dolech 2, PO Box 513, 5600 MB Eindhoven, The Netherlands. E-mail: {M.v.Almsick, T.Peeters, V.Prckovska, A.Vilanova, B.M.terhaarRomeny}@tue.nl.*

Manuscript received 1 July 2009; revised 8 Jan. 2010; accepted 16 Feb. 2010; published online 8 Apr. 2010.
Recommended for acceptance by H. Shen.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCGSI-2009-07-0133.
Digital Object Identifier no. 10.1109/TVCG.2010.61.

1077-2626/11/$26.00 © 2011 IEEE Published by the IEEE Computer Society

more realistic PDF. This PDF-measurement aims to recover the diffusion of water molecules in a multidirectional fiber population. Popular analysis techniques that transform the diffusion weighted data to certain probability density functions are Q-ball imaging [5], diffusion orientation transform (DOT) [6], and high order tensors (HOT) [7]. The results produced by these techniques can always be presented in the form of a spherical function $\psi(\theta, \phi)$ that characterizes the local intravoxel fiber structure. A spherical function $\psi(\theta, \phi)$ is expandable in a Laplace series and is as such subject to our rendering technique. Consequently, it is straightforward to represent HARDI data by a field of spherical function glyphs, each glyph corresponding to a local PDF or orientation distribution function (ODF).

Recently, one can observe an increase in research activity regarding deterministic and probabilistic HARDI fiber tracking algorithms [8], [9], [10]. The resulting fiber visualizations give a clear representation of the data while a glyph visualization can be confusing and cluttered for the end user. However, all fiber tracking techniques exhibit several ambiguities and disadvantages, such as choice of initialization, sensitivity with respect to the estimated principal direction, lack of connectivity information between regions of the brain in deterministic trackings, considerable computational effort in probabilistic trackings, and most importantly, the lack of thorough validation. Validation is needed before fiber tracking can be used in clinical applications. Knowledge about the local HARDI information is still quite important for testing and improving the fiber tracking techniques and, to the best of our knowledge, showing the local structure by glyph representation is the most common and reliable way of visualizing HARDI data.

The traditional way of visualizing HARDI glyphs starts by generating a mesh of points on a sphere, and then deforming it as mentioned above. This visualization is potentially slow and memory consuming, especially if one wants to achieve smooth and exact surfaces which requires an icosahedral tessellation of fourth order or higher. Using a coarser mesh is an option to gain speed, but then lots of subtle details on the surface are missed and quite often the angular maxima are slightly displaced, as shown in Fig. 1.

The glyph ray-casting approach in this paper achieves a fast, detailed, and accurate rendering of high-dimensional diffusion-weighted (DW) data. Furthermore, it is applicable to any form of data that can be expressed by spherical functions. We give an overview of related work on HARDI visualization and GPU-based glyph rendering in Section 2. The mathematical tools for ray-casting spherical functions is the subject of Section 3. In Section 4, we show how to incorporate these mathematical tools in a GPU-based ray-casting implementation for rendering HARDI glyphs. The analysis of visual aspects and performance measurements follows in Section 5. Finally, in Section 6, we conclude this manuscript by addressing future challenges.

2 Related Work

In DTI, glyphs are the basic tools to visualize the local diffusivity of water. The covariant matrix of the Gaussian PDF is a second order, symmetric, positive-definite tensor. Tensor glyphs are, therefore, the appropriate tool to depict

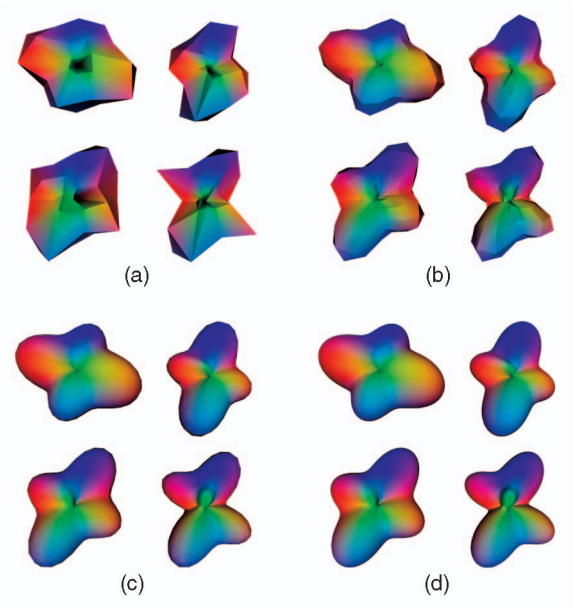

Fig. 1. Traditional way of HARDI glyph visualization. The glyphs' surface becomes smoother as the order of tessellation of the icosahedron is increased from order 2 (a) to order 7 (d). Changes of the maxima can be observed as the sampling of the glyphs' surface becomes denser. (a) Order 2. (b) Order 3. (c) Order 4. (d) Order 7.

the complete tensor information. Commonly used shapes for visualizing the DTs are cuboids, ellipsoids, and superquadrics [11]. There has been work in glyph packing algorithms by Kindlmann [12] and Hlawitschka et al. [13], but these algorithms change the glyph location for better visual perception.

HARDI techniques overcome the disadvantages of DTI by applying high-order methods in the description of water molecule diffusion. However, these methods significantly raise the complexity of data processing and visualization. Among the available software packages for visualizing and processing HARDI data are Slicer [14] with the Qball plug-in [15] and Camino [16]. The latter one has very limited visualization capabilities since its core purpose is the processing of HARDI data. Recently, Shattuck et al. [17] developed a set of tools for visualizing ODF models. All of the mentioned packages apply the same concept. To obtain an ODF shape via polygons, they generate a mesh of points on a sphere and in- or extrude the vertices. This approach renders an interactive scene only after the geometry is calculated. Furthermore, there is always a trade-off between the visual quality and rendering speed. In Shattuck et al. [17], the interaction is reported as 10 fps on a single brain data slice with 225 samples on a sphere that approximates fourth order icosahedral tessellation. In their work, this limitation in rendering performance is being mentioned and GPU programming is suggested as a solution. Higher frame rates were reported by Schultz [18] using the same technique. He obtained a frame rate of up to 77 fps for 196 glyphs with third order icosahedral tessellation on a 2 GHz processor and a NVIDIA Quadro NVS 110M.

Interesting work by Hlawitschka et al. [19] addresses the problem of improving the speed and robustness of the fiber tracking algorithms in HARDI. However, no performance information is given and the underlying field of HOT or

spherical harmonics (SH) representation still needs sampling on a predefined grid, which gives rise to the same disadvantages discussed above.

Recent advances in capability and performance of GPUs have triggered several publications about glyph rendering where ray-casting on GPUs has been applied. Application areas include mathematics [20] and molecule rendering [21]. However, those methods focus on simple glyph shapes with few parameters, such as ellipsoids, where the intersection with the view-ray can be computed analytically. Recently, Mario Hlawitschka and Scheuermann [22] used GPU ray-casting to render superquadric tensor glyphs [23]. The degrees of freedom of spherical functions and, thus, the number of parameters that define the shape of our glyph is much higher than in the above-mentioned cases. This leads to more elaborate computations for the intersection of the view-ray with the glyph and for the normal vector at the intersection.

In a preceding paper [24], we have published a basic ray-casting algorithm for spherical functions. In the current work, we now fully utilize all applicable properties of spherical harmonics and introduce a far more efficient and faster algorithm. Our more advanced algorithm solves the HARDI visualization problem on the GPU and one obtains a substantially faster, accurate visualization of HARDI data sets well suited for interactive usage.

3 Methods

First, we introduce all relevant mathematical concepts of our glyph rendering algorithm before discussing the implementation details in the next section. We start with the Laplace expansion of real-valued spherical functions ψ in real-valued spherical harmonics $\tilde{Y}_l^m$. We then cast view-rays from an observer onto the surface of the spherical function ψ in 3-dimensional space. For this, we need to determine the location of a ray-glyph intersection and we need to determine the normal vector with respect to the glyph surface at that intersection.

To facilitate and to speed up the indispensable numerical calculations needed for the ray-glyph intersection, we alter the representation of the spherical harmonics $\tilde{Y}_l^m$ in the Laplace expansion in two steps. First, we align the parameterization of the spherical harmonics $\tilde{Y}_l^m$ with the view axis via rotation with Wigner matrices. Then, we rewrite the Laplace expansion in cylindrical coordinates to obtain a fairly simple, polynomial expression Φ whose 0-contour marks the glyph surface. The polynomial expression Φ requires fewer trigonometric function calls, multiplications, and additions than the explicit formula with $\tilde{Y}_l^m$, thereby speeding up the GPU-based process, in which we numerically probe expression Φ along the view-ray utilizing the bisection method or *regula falsi* [25] to obtain the ray-glyph intersection. The glyph normal that determines the shading of the view-ray pixel is given by the gradient of expression Φ.

We also consider a bounding cylinder that encompasses the glyph and its spherical function. The tight cylindrical encapsulation improves the overall efficiency of the algorithm. Rays outside the cylinder do not have to be considered and the numerical sampling for the ray-glyph intersection can be confined to the inside of the cylinder.

3.1 Laplace Expansion

A real-valued function ψ on a sphere S^2, $\psi : S^2 \to \mathbb{R}^+$, can be expanded in an orthonormal basis of spherical functions, orthonormal with respect to the hermitian inner product

$$\langle \psi_2 | \psi_1 \rangle := \int_0^{2\pi} \int_0^{\pi} \psi_1(\vartheta, \varphi)\ \psi_2^*(\vartheta, \varphi) \sin\vartheta\ d\vartheta\ d\varphi. \tag{2}$$

The default choice is the orthonormal set of complex-valued spherical harmonics Y_l^m [26],

$$Y_l^m : S^2 \to \mathbb{C}, \text{ for all } l \in \mathbb{N}^0 \text{ and } m \in \mathbb{Z} \text{ with } |m| \le l, \tag{3}$$

which are explicitly defined by associated Legendre polynomials P_l^m

$$Y_l^m(\vartheta, \varphi)\ =\ e^{i\,m\,\varphi} \sqrt{\frac{2l+1}{4\pi} \frac{(l-m)!}{(l+m)!}}\, P_l^m(\cos\vartheta), \tag{4}$$

with spherical angles $\vartheta \in [0, \pi]$ and $\varphi \in [0, 2\pi)$.

The expansion of a spherical function ψ in Y_l^m is known as the Laplace expansion.

$$\psi(\vartheta, \varphi) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} a_l^m\ Y_l^m(\vartheta, \varphi). \tag{5}$$

The a_l^ms denote the expansion coefficients. Obviously, the above expansion is only feasible in an implementation if one can safely truncate the outer sum at a given $l_{\max}$.

The expansion of a spherical function ψ via the Laplace series (5) is quite practical, since spherical harmonics Y_l^m posses several convenient properties. The most important property with respect to HARDI is the fact that the spherical harmonics Y_l^m are eigenfunctions of the spherical component Δ_{S_2} of the Laplacian differential operator Δ.

$$\begin{aligned} \Delta_{S_2}\ Y_l^m(\vartheta, \varphi) &= \left(\frac{1}{\sin^2\vartheta} \partial_\varphi^2 + \frac{1}{\sin\vartheta} \partial_\vartheta\ \sin\vartheta \partial_\vartheta \right) Y_l^m(\vartheta, \varphi) \\ &= -l(l+1)\ Y_l^m(\vartheta, \varphi). \end{aligned} \tag{6}$$

Note that the Laplace operator Δ governs the homogeneous diffusion equation $\partial_t = D\,\Delta$ and that diffusion is the underlying physical process of diffusion weighted molecular resonance imaging.

The Y_l^ms are the Laplacian eigenfunction in spherical coordinates where as $e^{i\,\vec{\omega}\cdot\vec{x}}, \cos(\vec{\omega}\cdot\vec{x})$, and $\sin(\vec{\omega}\cdot\vec{x})$ are Laplacian eigenfunctions in Cartesian coordinates. An expansion in these latter functions constitutes a Fourier transformation. Correspondingly, the Y_l^ms form the basis functions of a spherical Fourier transformation, the spherical Fourier coefficients being the Laplacian expansion coefficients a_l^m.

$$a_l^m\ =\ \langle Y_l^m | \psi \rangle\ =\ \int_0^{2\pi} \int_0^{\pi} \psi(\vartheta, \varphi)\ \left(Y_l^m(\vartheta, \varphi) \right)^*\ \sin\vartheta\ d\vartheta\ d\varphi. \tag{7}$$

The inverse spherical Fourier transformation is given by the Laplace series (see (5)).

The Laplace series (5) deals with complex-valued functions. We, however, have to consider the expansion of

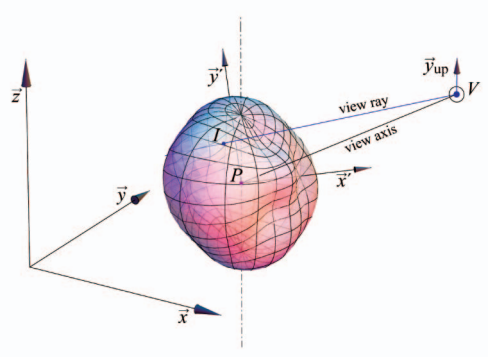

Fig. 2. The glyph of a spherical function ψ at position P is rendered via ray-casting. The view-ray starting at the observer position V intersects the surface of the glyph at position I.

a real-valued function ψ. Thus, the expansion in complex-valued spherical harmonics Y_l^m inflicts an unnecessary computational burden. To overcome this issue, we utilize the identity

$$\left(Y_l^m\right)^* \;=\; (-1)^m\, Y_l^{-m}, \tag{8}$$

which allows us to form the following real-valued linear combinations (9) and (10) of spherical harmonics that constitute a new set of basis functions.

$$\sqrt{2}\,\mathrm{Re}\left(Y_l^m\right) = \frac{1}{\sqrt{2}}\left(Y_l^m + (-1)^m\, Y_l^{-m}\right), \tag{9}$$

$$\sqrt{2}\,\mathrm{Im}\left(Y_l^m\right) = \frac{-i}{\sqrt{2}}\left(Y_l^m - (-1)^m\, Y_l^{-m}\right). \tag{10}$$

Be aware that $\mathrm{Re}(Y_l^m)$ and $\mathrm{Re}(Y_l^{-m})$ as well as $\mathrm{Im}(Y_l^m)$ and $\mathrm{Im}(Y_l^{-m})$ are linearly dependent, so that we consider the following set of basis functions $\tilde{Y}_l^m$ according to the convention used by Descoteaux et al. [27].

$$\tilde{Y}_l^m(\vartheta,\varphi) := \begin{cases} \sqrt{2}\,\mathrm{Re}\left(Y_l^m(\vartheta,\varphi)\right), & \text{if } m<0,\\ Y_l^0(\vartheta,\varphi), & \text{if } m=0,\\ \sqrt{2}\,\mathrm{Im}\left(Y_l^m(\vartheta,\varphi)\right), & \text{if } m>0. \end{cases} \tag{11}$$

These real-valued spherical harmonics $\tilde{Y}_l^m(\vartheta,\varphi)$ are again orthonormal due to the normalization factor $\sqrt{2}$. The tilde indicates the use of real-valued instead of complex-valued spherical harmonics and the corresponding Laplace expansion reads

$$\psi(\vartheta,\varphi) = \sum_{l=0}^{\infty}\sum_{m=-l}^{l} \tilde{a}_l^m\, \tilde{Y}_l^m(\vartheta,\varphi). \tag{12}$$

One obtains the expansion coefficients $\tilde{a}_l^m$ by substituting $\tilde{Y}_l^m(\vartheta,\varphi)$ into (7).

In the application of DTI and HARDI visualization, we encounter only spherical functions ψ_S that are symmetric under spatial inversion at their center P. Consequently, the in- or extrusion of ψ_S-glyphs in one direction are the same as in the opposite direction.

$$\psi_S(\vartheta,\varphi) \;=\; \psi_S(\pi-\vartheta,\varphi+\pi). \tag{13}$$

Spherical harmonics $\tilde{Y}_l^m$ with even l also exhibit this symmetric property, whereas $\tilde{Y}_l^m$ with odd l are antisymmetric under spatial inversion. Thus, the Laplace expansion

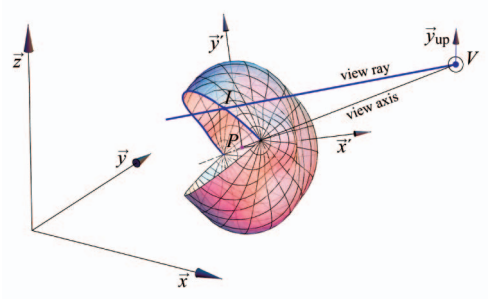

Fig. 3. A spherical parameterization of glyph ψ that is aligned with the PV-axis has the benefit, that the point of view-ray and glyph intersection I has to be on a given ϑ'-coordinate line depicted in red.

(12) of an inversion symmetric spherical function ψ_S is independent of spherical harmonics $\tilde{Y}_l^m$ with odd l and the expansion simplifies to

$$\psi_S(\vartheta,\varphi) = \sum_{l=0,2,4,\ldots}^{\infty}\sum_{m=-l,-l+1,\ldots}^{l} \tilde{a}_l^m\, \tilde{Y}_l^m(\vartheta,\varphi). \tag{14}$$

3.2 Ray-Casting

Our ray-casting geometry for a glyph is depicted in Fig. 2. The spherical functions ψ or ψ_S given by the Laplace expansions (12) or (14) with a truncation value $l_{\max}$ define a smooth surface around the glyph center P. We need to determine the intersection I where a view-ray-casted by the observer at position V intersects the glyph surface.

3.2.1 Realignment

To facilitate the calculation, we use a glyph-observer-based coordinate system with basis vectors $\vec{x}'$, $\vec{y}'$, and $\vec{z}'$ located at the glyph center P. Unit vector $\vec{z}'$ (not depicted in Fig. 2) is defined by the view axis $\overrightarrow{PV}$ that runs through the glyph center P and the observer position V. Basis vector $\vec{x}'$ is orthogonal to $\vec{z}'$ and the up-vector $\vec{y}'_{\text{up}}$ of the observer. The remaining $\vec{y}'$-vector is given by $\vec{z}' \times \vec{x}'$.

$$\vec{x}' := \frac{\vec{y}'_{\text{up}} \times \vec{z}'}{\left\|\vec{y}'_{\text{up}} \times \vec{z}'\right\|}, \tag{15}$$

$$\vec{y}' := \vec{z}' \times \vec{x}', \tag{16}$$

$$\vec{z}' := \frac{\overrightarrow{PV}}{\left\|\overrightarrow{PV}\right\|}. \tag{17}$$

It is a nontrivial challenge to determine the intersection point I, where the view-ray hits the surface of the glyph ψ. For Laplace expansions of arbitrary but finite $l_{\max}$, this can only be achieved via numerical methods. However, the parameterization of ψ by spherical (world) coordinates ϑ and φ is not optimal for this task. The default spherical coordinate system is usually aligned along the world z-axis and, thus, the coordinate poles and the ϑ-coordinate lines of ψ do not correspond with the glyph-ray geometry (see Fig. 2). To overcome this handicap, we rotate the spherical coordinates that parameterize ψ such that they align with $\vec{z}'$ on the PV-axis (see Fig. 3). To do so, we perform a rotation around $\vec{x}$, $\vec{y}$, and $\vec{z}$ with the Euler angles α, β, γ such that the world basis $\vec{x}, \vec{y}, \vec{z}$ would align with the glyph-observer basis $\vec{x}', \vec{y}', \vec{z}$. The respective Euler angles are

$$\alpha := \arctan \frac{z'_x}{z'_y}, \tag{18}$$

$$\beta := \arctan \frac{z'_z}{\sqrt{(x'_z)^2 + (y'_z)^2}}, \tag{19}$$

$$\gamma := \arctan \frac{x'_z}{y'_y}, \tag{20}$$

where z'_x denotes the x-component of vector $\vec{z}'$ with respect to world coordinates, and so on.

Keep in mind that only the spherical coordinate parameterization of ψ has to be rotated, not the glyph ψ itself. We achieve this by rotating the complete glyph with its spherical coordinate parameterization and then undoing the rotation by rotating the glyph back via its Laplace expansion coefficients a_l^m.

Recall that there are three ways to rotate a spherical function given by a Laplace expansion. One can rotate the spherical function ψ by simply substituting the spherical angles ϑ' and φ' of a rotated coordinate system (21), one can rotate the spherical function by rotating the function basis Y_l^m with the corresponding rotation matrices, the so-called Wigner matrices $\mathbf{D}_l^{m\,m'}$, as in (22), or one can apply the inverse Wigner matrices to the expansion coefficients a_l^m as in (23).

$$\mathcal{R}_{\alpha\beta\gamma}[\psi(\vartheta,\varphi)] = \psi(\vartheta',\varphi'), \tag{21}$$

$$\mathcal{R}_{\alpha\beta\gamma}[\psi(\vartheta,\varphi)] = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} a_l^m \left(\sum_{m'} \mathbf{D}_l^{m\,m'}(\alpha,\beta,\gamma)\, Y_l^{m'}(\vartheta,\varphi) \right), \tag{22}$$

$$\mathcal{R}_{\alpha\beta\gamma}[\psi(\vartheta,\varphi)] = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \left(\sum_{m'} \mathbf{D}_l^{m\,m'}(-\gamma,-\beta,-\alpha)\, a_l^{m'} \right) Y_l^m(\vartheta,\varphi). \tag{23}$$

We rotate the glyph with its spherical coordinates ϑ and φ along the z-axis by simply substituting the spherical coordinates ϑ' and φ' along the z'-axis as in (21). Then, we undo the rotation of just the glyph by performing the inverse rotation on the expansion coefficients $\tilde{a}_l^m$ as in (23) via the Wigner matrices $\tilde{\mathbf{D}}_l^{m\,m'}(\alpha,\beta,\gamma)$ as defined in Appendix A. Note, however, that the regular Wigner matrices are defined for the complex-valued spherical harmonics Y_l^m. With the tilde, we identify the Wigner matrices that are transformed according to (9) to the basis of real-valued spherical harmonics $\tilde{Y}_l^m$ given in (11).

The glyphs with the rotated parameterization are then given by the expansion

$$\psi(\vartheta',\varphi') = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \tilde{b}_l^m\, \tilde{Y}_l^m(\vartheta',\varphi'), \tag{24}$$

with the rotated, real-valued expansion coefficients $\tilde{b}_l^m$ given by

$$\tilde{b}_l^m = \sum_{m'} \tilde{\mathbf{D}}_l^{m\,m'}(\alpha,\beta,\gamma)\, \tilde{a}_l^{m'}, \quad \forall\, l \in \mathbb{N}^0. \tag{25}$$

The transformation of Laplace expansion coefficients needs to be done for every glyph once and needs to be redone whenever the viewpoint V changes. The refinements in the subsequent section need to be performed for every view-ray once, but not for every sampling point on the view-ray. The refinements lead to a fairly simple mathematical expression Φ, that provides an efficient way for sampling the view-ray, thus, facilitating the numerical determination of the ray-glyph intersection I.

3.2.2 Cylindrical Coordinates

The realignment of the glyph parameterization is just a first step to simplify the expressions in the Laplace expansion (12). In a second step, we replace the rotated spherical coordinates r', ϑ', φ' of $\tilde{Y}_l^m(\vartheta',\varphi')$ by cylindrical coordinates ρ', φ', z' that are even more suitable for the description of the view-ray $\overrightarrow{VI}$. The cylindrical coordinates are given by

$$\rho' = r' \sin\vartheta', \quad \varphi' = \varphi', \quad z' = r'\, \cos\vartheta'. \tag{26}$$

The inverse transformation from cylindrical back to spherical coordinates is

$$r' = \sqrt{\rho'^2 + z'^2}, \quad \varphi' = \varphi', \quad \vartheta' = \arctan\frac{\rho'}{z'}. \tag{27}$$

Up to now, the definition of our glyph has been in explicit form. $\psi(\vartheta',\varphi')$ gives the radial extent r' of the glyph. In cylindrical coordinates, however, r' is incorporated in the coordinates ρ' and z' and the explicit definition of $\psi(\vartheta',\varphi') = r'$ becomes an implicit equation for ρ' and z'.

$$\psi(\rho',\varphi',z') = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \tilde{b}_l^m\, \tilde{Y}_l^m(\rho',\varphi',z') = \sqrt{\rho'^2 + z'^2}, \tag{28}$$

where we have used a kind of operator overload-notation $\psi(\rho',\varphi',z')$ and $\tilde{Y}_l^m(\rho',\varphi',z')$ for ψ and $\tilde{Y}_l^m$ in cylindrical coordinates.

The implicit nature of (28) is not a disadvantage in the numerical calculations that will follow. Traversing along a view-ray from the observer V to the point of ray-glyph intersection I and beyond, one only has to verify, if one is still out- or already inside the glyph. This is still possible with the implicit (28). As long as the cylindrical coordinates are outside of glyph ψ, the glyph-radius, given by $\psi(\rho',\varphi',z')$, will be smaller than the distance $\sqrt{\rho'^2 + z'^2}$ of the sample point location to the glyph origin P. Vice versa, for a sample point inside the glyph, the radius $\psi(\rho',\varphi',z')$ will be larger than $\sqrt{\rho'^2 + z'^2}$. Hence, for

$$\Phi(\rho',\varphi',z') \;:=\; \psi(\rho',\varphi',z') - \sqrt{\rho'^2 + z'^2}, \tag{29}$$

we obtain the inequalities

$$\Phi(\rho',\varphi',z') \geq 0, \quad \text{for } \rho' \text{ and } z' \text{ inside } \psi, \tag{30}$$

$$\Phi(\rho',\varphi',z') < 0, \quad \text{for } \rho' \text{ and } z' \text{ outside } \psi. \tag{31}$$

A further simplification of expression $\Phi(\rho',\varphi',z')$ in the above inequalities depends on the explicit choice of the Laplace expansion truncation $l_{\max}$. If we multiply the above inequalities with $(\sqrt{\rho'^2 + z'^2})^{l_{\max}}$, we obtain for the Laplace expansion of $\psi(\rho',\varphi',z')$ a sum of two homogeneous polynomials in ρ' and z' of degree $l_{\max}$ and $l_{\max} - 1$.

$$\begin{aligned}\left(\sqrt{\rho'^2+z'^2}\right)^{l_{\max}}\psi(\rho',\varphi',z') &= \sum_{n=0}^{l_{\max}} c_n^{\text{even}}(\varphi')\ \rho'^n\ z'^{l_{\max}-n} \\ &+ \sqrt{\rho'^2+z'^2}\sum_{n=0}^{l_{\max}-1} c_n^{\text{odd}}(\varphi')\ \rho'^n\ z'^{l_{\max}-1-n}.\end{aligned} \tag{32}$$

It is straightforward to derive the φ'-dependent coefficients $c_n^{\text{even}}(\varphi')$ and $c_n^{\text{odd}}(\varphi')$ by expanding the spherical harmonics $\tilde{Y}_l^m$ of (28) in (32). The specific case of $l_{\max}=4$ is listed in Appendix B. Furthermore, all coefficients $c_n^{\text{odd}}(\varphi')$ are 0 for glyphs ψ_S that are symmetric under spatial inversion. Hence, for $l_{\max}=4$, we are left with five real-valued coefficients $c_n^{\text{even}}(\varphi')$ to specify a symmetric glyph ψ_S and another four real-valued coefficients $c_n^{\text{odd}}(\varphi')$ for general glyphs ψ.

To obtain a numerically more stable implementation, we use the following substitution:

$$p := \frac{\rho'}{\sqrt{\rho'^2+z'^2}} = \frac{\rho'}{r'} = \sin\vartheta', \tag{33}$$

$$q := \frac{z'}{\sqrt{\rho'^2+z'^2}} = \frac{z'}{r'} = \cos\vartheta'. \tag{34}$$

With p and q, we can write for $\psi(\rho',\varphi',z')$

$$\begin{aligned}\psi(\rho',\varphi',z') = &\sum_{n=0}^{l_{\max}} c_n^{\text{even}}(\varphi')\ p^n\ q^{l_{\max}-n} \\ &+ \sum_{n=0}^{l_{\max}-1} c_n^{\text{odd}}(\varphi')\ p^n\ q^{l_{\max}-1-n}.\end{aligned} \tag{35}$$

Before sampling a view-ray, we simply need to substitute the specific φ' of the view-ray into the equations for $c_n^{\text{even}}(\varphi')$ and $c_n^{\text{odd}}(\varphi')$. Then, we can readily sample the view-ray via (35) for position I in a rather efficient manner.

3.3 Glyph Normals

To determine the correct shading of a pixel whose view-ray intersects a glyph at position I, we still need to determine the glyph normal at I. Recall that the 0-contour of expression Φ in (29) constitutes the surface of glyph ψ and that gradients are always orthogonal to contours. Hence, the gradient of Φ at position I provides the normal vector that we seek.

The derivation of the gradient vector $\nabla\Phi$ is straightforward, though slight complications arise due to the cylindrical coordinates. The gradient vector $\nabla\Phi$ in Cartesian coordinates $\vec{x}'$, $\vec{y}'$, $\vec{z}'$ is given by

$$\nabla\Phi = \nabla\psi - \nabla r', \tag{36}$$

with

$$\nabla\, r' \;=\; \nabla\,\sqrt{\rho'^2+z'^2} = \begin{pmatrix} p\,\cos\varphi' \\ p\,\sin\varphi' \\ q \end{pmatrix}, \tag{37}$$

and

$$\nabla\,\psi(\rho',\varphi',z') \;=\; \begin{pmatrix} \cos\varphi'\,\partial_{\rho'} \;-\; \frac{\sin\varphi'}{\rho'}\,\partial_{\varphi'} \\ \sin\varphi'\,\partial_{\rho'} \;+\; \frac{\cos\varphi'}{\rho'}\,\partial_{\varphi'} \\ \partial_{z'} \end{pmatrix}\psi(\rho',\varphi',z'). \tag{38}$$

The partial differentials with respect to ρ', φ', and z' are

$$\begin{aligned}\partial_{\rho'}\,\psi(\rho',\varphi',z') &= \sum_{n=1}^{l_{\max}} c_n^{\text{even}}(\varphi')\frac{p^2(n-l_{\max})+n\,q^2}{r'}\ p^{n-1}q^{l_{\max}-n} \\ &+ \sum_{n=1}^{l_{\max}-1} c_n^{\text{odd}}(\varphi')\ \frac{p^2(n+1-l_{\max})+n\,q^2}{r'}\ p^{n-1}q^{l_{\max}-n-1}.\end{aligned} \tag{39}$$

$$\begin{aligned}\partial_{z'}\,\psi(\rho',\varphi',z') &= -\sum_{n=0}^{l_{\max}-1} c_n^{\text{even}}(\varphi')\ \frac{p^2(n-l_{\max})+n\,q^2}{r'}\ p^n q^{l_{\max}-n-1} \\ &- \sum_{n=0}^{l_{\max}-2} c_n^{\text{odd}}(\varphi')\ \frac{p^2(n+1-l_{\max})+n\,q^2}{r'}\ p^n q^{l_{\max}-n-2}.\end{aligned} \tag{40}$$

$$\begin{aligned}\partial_{\varphi'}\,\psi(\rho',\varphi',z') = &\sum_{n=0}^{l_{\max}}\left(\partial_{\varphi'}\ c_n^{\text{even}}(\varphi')\right)p^n q^{l_{\max}-n} \\ &+ \sum_{n=0}^{l_{\max}-1}\left(\partial_{\varphi'}\ c_n^{\text{odd}}(\varphi')\right)p^n q^{l_{\max}-n-1}.\end{aligned} \tag{41}$$

At the point of ray-glyph intersection I, the coordinate values r', p, q, and φ' are given and the gradient is readily computed via (36) through (41). The resulting normal vector determines the reflection of the view-ray toward the light source and the corresponding pixel shading.

3.4 Outer Bound

Traversing casted rays for glyph intersections is a computational expensive task. We, therefore, try to avoid it if possible. For this purpose, we determine an outer bound for each glyph. A ray outside the outer bound will not intersect the glyph surface and the above considerations are futile. However, if a ray passes through the outer bound, a ray-glyph intersection I is likely. In this case, the outer bound provides a convenient bracket for the view-ray sampling of Φ.

In accordance with the cylindrical coordinates, we consider a cylinder for an outer bound as depicted in Fig. 5. We estimate the cylinder size given by its radius $\rho_{\max}$ and depth $z'_{\max}$ by constructively adding the maximal radii ${\rho_{\max}}_l^m$ and the maximal cylinder depths ${z'_{\max}}_l^m$ of all real-valued spherical harmonics $\tilde{Y}_l^m$ according to the Laplace expansion (12). We thereby obtain an estimate for a cylindrical outer bound that will definitely encompass the glyph inside.

$$\rho'_{\max} = \sum_{l=0}^{l_{\max}}\sum_{m=-l}^{l}\left|\tilde{b}_l^m\right|\,{\rho_{\max}}_l^m. \tag{42}$$

$$z'_{\max} = \sum_{l=0}^{l_{\max}}\sum_{m=-l}^{l}\left|\tilde{b}_l^m\right|\,{z_{\max}}_l^m. \tag{43}$$

The maximal radii ${\rho_{\max}}_l^m$ and cylinder depths ${z_{\max}}_l^m$ for $l\le 4$ are given in Appendix C.

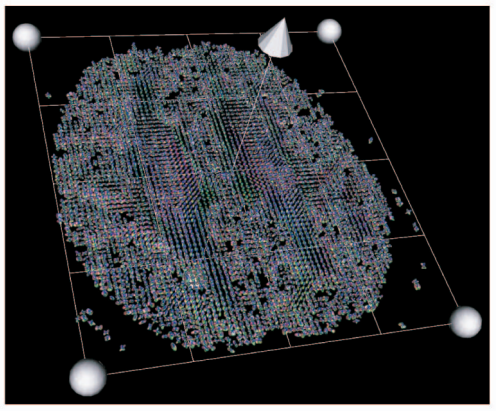

Fig. 4. The interactive rendering of glyphs in a POI of close to 100×100 seed points in a HARDI data set of a human brain.

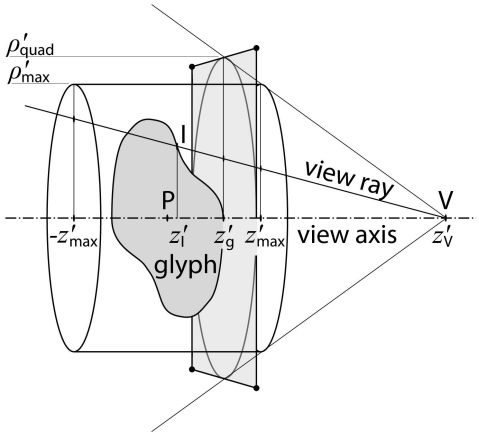

Fig. 5. The outer bound of a glyph ψ is given by a cylinder along the rotated z'-axis which coincides with the view axis $\overrightarrow{PV}$. The square polygon that acts as a screen displaying the glyph is attached at the glyph on the view axis at z'_{g}. Sampling the view-ray for intersection I is done within the cylinder bound.

In the preceding publication [24], we consider a sphere as an outer bound. The cylinder, however, tends to encompass the glyph in a more tight fashion resulting in a more efficient algorithm.

4 Implementation

The mathematics outlined above can be utilizes to visualize any kind of spherical function ψ. The subsequent algorithm also holds for any spherical function ψ. The only feature specific to our HARDI application is the omission of odd l-values in the Laplace expansion of ψ.

The ψ-glyphs, in our HARDI application, display the spherical characteristics of water diffusivity in a cross section of a HARDI volume. The cross section of a HARDI volume is defined by a plane-of-interest (POI), which the user can translate, rotate, and resize using the mouse. It is impractical to render a volume, a thick slice of several glyph-layers, or an arbitrarily dense array of glyphs, since it would be visually overwhelming for the user. Instead, we consider a 2-dimensional array of seed points that define the locations of the glyphs and which is implicitly given by the POI and a user-specified seed grid constant. An example of a POI of close to 100×100 seed points of a human brain is given in Fig. 4. It is also possible to consider other distributions of seed points, such as strings along fiber tracks.

Rendering a scene, we step through all the seed points sequentially and deal with each glyph separately. Per glyph, the calculations are divided between CPU and GPU. On the CPU, we perform all the calculations that can be done *per glyph* and on the GPU we run a ray-casting algorithm *per pixel*. In the following sections, we describe these calculations in more detail.

4.1 CPU

Prior to any rendering calculation, the HARDI data have to be expressed in a set of real-valued Laplace expansion coefficients $\tilde{a}_l^m$ ($l = 1, \ldots, 4$ and $|m| \leq l$) for each voxel. This is achieved via (7) and appropriate numerical methods as described in Numerical Recipes [25].

Specifically, for the HARDI application, the following calculations are then performed for all glyphs at each seed point. We manipulate the expansion coefficients $\tilde{a}_l^m$ according to parameters that the user can set interactively. The user can specify a scaling factor, with which all coefficients $\tilde{a}_l^m$ are multiplied in order to change the size of the glyphs. Another user-defined parameter, the *sharpness* variable, rescales the $\tilde{a}_0^0$ via scalar division to reduce the influence of the isotropic spherical harmonic component $\tilde{Y}_0^0$, thus, enhancing the anisotropic maxima of the glyphs. Finally, the user has the option to scale all glyphs independently. The user may want to adjust the $\tilde{a}_0^0$ for each glyph to the same over all value, thus, annihilating all differences in isotropic diffusion magnitude. Alternatively, the user may want to rescale all glyphs by the global $\tilde{a}_0^0$-maximum such that the differences in diffusion magnitude between different voxels become apparent.

The last interactive manipulation by the user is the choice of viewpoint V and the observers up-vector $\vec{y}_{\text{up}}$. For each new viewpoint and up-vector, we need to rotate the Laplace expansion coefficients $\tilde{a}_l^m$ to obtain via (15), (16), (17), (18), (19), (20), and (25) their rotated version $\tilde{b}_l^m$, the coefficients $\tilde{b}_l^m$ decompose ψ with respect to the real-valued spherical harmonics that are aligned with the view axis through viewpoint V.

We also construct for each glyph a bounding cylinder to facilitate the ray-casting calculations. For each glyph, the bounding cylinder is aligned with the view axis $\overrightarrow{PV}$ as depicted in Fig. 5. The center of the cylinder is the seed point P of the glyph. The extent of the bounding cylinder is given by its radius ρ'_{max} and its size along the view axis reaching from $-z'_{\text{max}}$ to z'_{max} as defined in (42) and (43).

Finally, we need to generate for each glyph a geometry that covers all the pixels of the glyph projection onto the viewport. For each glyph, the geometry will consist of a mere square polygon that we position right in front of the glyph as seen from the viewpoint and as depicted in Fig. 5. In the glyph coordinate system, the center position of the square polygon is given by z'_g in $\vec{z}'$-direction, which is the intersection point of the glyph with the view axis $\overrightarrow{PV}$. One obtains z'_g by filling $\vartheta' = \varphi' = 0$ into (24).

$$z'_{\text{g}} = \sum_{l=0}^{l_{\text{max}}} \sqrt{\frac{2\,l+1}{4\pi}}\,\tilde{b}_l^0. \tag{44}$$

The square polygon is spanned by the orthogonal vectors $\vec{x}'$ and $\vec{y}'$. To assure that the square polygon covers the projection of the glyph and its bounding cylinder on the viewport, the square has to encompass a circle of radius ρ'_{quad}, which we derive by the rule of three with the two ratios $\rho'_{\text{quad}}/(z'_{\text{V}} - z'_{\text{g}})$ and $\rho'_{\text{max}}/(z'_{\text{V}} - z'_{\text{max}})$ as displayed in Fig. 5. We obtain for $z'_{\text{V}} > z'_{\text{g}}$

$$\rho'_{\text{quad}} = \rho'_{\text{max}} \, \frac{z'_{\text{V}} - z'_{\text{g}}}{z'_{\text{V}} - z'_{\text{max}}}, \tag{45}$$

with $z'_{\text{V}} = \|\overrightarrow{PV}\|$. If the viewpoint V is too close to the glyph ($z'_{\text{V}} \sim z'_{\text{max}}$) or even inside the bounding cylinder ($z'_{\text{V}} < z'_{\text{max}}$), it is sufficient to set ρ'_{quad} equal to the viewport size. In this latter case, the glyph tends to fill the screen. For $z'_{\text{V}} \leq z'_{\text{g}}$, no glyph will be drawn. Thus, for $z'_{\text{g}} < z'_{\text{V}}$, the geometry of each glyph is given by a square polygon and its four corner vertices at $(\pm\rho'_{\text{quad}}, \pm\rho'_{\text{quad}}, z'_{\text{g}})$ in glyph coordinates $\vec{x}'$, $\vec{y}'$, and $\vec{z}'$.

This concludes the calculations performed on the CPU. For each glyph, we pass the Laplace expansion coefficients $\tilde{b}_l^m$ as vertex attributes to the GPU. In our case of HARDI visualization, we have to cope with 15 such coefficients, since we truncate the Laplace expansion at $l_{\text{max}} = 4$ and since we only have to handle even l coefficients for inversion symmetric glyphs. Furthermore, we need to pass on the two parameters of the bounding cylinder and the four vertices of the square polygon. The remaining computation is handled by the GPU.

4.2 GPU

In the vertex shader, we merely compute for each glyph the view directions of its four square polygon corner vertices by subtracting the position vector of the viewpoint V from the position vector of each vertex. The view direction $\vec{v}$ of an arbitrary view-ray is then automatically interpolated for use in the fragment shader. The 15 Laplace expansion coefficients $\tilde{b}_l^m$ are simply passed on to the fragment shader.

The major computational load on the GPU is handled by the fragment shader performing a ray-cast for each pixel of the square polygon of each glyph. This ray-cast is achieved in three steps.

1. Compute the intersections of the view-ray with bounding cylinder of the glyph. If there are no intersections, discard the fragment. If there are intersections, use them as brackets in the next step.
2. Determine the ray-glyph intersection I numerically by sampling expression Φ on the view-ray inside the bracket that was determined in the previous step. Proceed, if an intersection exists.
3. Compute the normal $\vec{n}$ of the glyph surface at intersection I and perform lighting calculations to compute the color of the current fragment. Update the Z-buffer using I.

These three steps are discussed in detail in the following sections.

4.2.1 Intersections of View-Ray and Bounding Cylinder

The view-ray can be conveniently parameterized by the cylindrical coordinates of a glyph. Note, that each ray exhibits a fixed φ'. Within the plane determined by φ' and the view axis, a view-ray is given by a linear ρ'-function of z'.

$$\rho'_{\text{ray}}(z') \;=\; \frac{\vec{v}_{\rho'}}{\vec{v}_{z'}} \, \big(z' - z'_{\text{V}}\big), \tag{46}$$

with $\vec{v}_{\rho'}$ being the projection of the normalized view-ray vector $\vec{v}$ on ρ' and $\vec{v}_{z'}$ being the z'-component of the normalized view-ray vector $\vec{v}$.

To eventually find the intersection I of a view-ray and a glyph, we define a search bracket for the z'-parameter in (46). The bracket will consist of the points z'_{near} and z'_{far}, where the view-ray enters and exits the bounding cylinder.

We have to distinguish three scenarios. If the viewpoint is outside the bounding cylinder, an intersecting view-ray has to pass the front face of the bounding cylinder to intersect the glyph. This first intersection point of view-ray and bounding cylinder is given by $(\rho'_{\text{ray}}(z'_{\text{max}}), \varphi', z'_{\text{max}})$ in cylindrical glyph coordinates and the first bracket is $z'_{\text{near}} = z'_{\text{max}}$. If the viewpoint is already inside the bounding cylinder but still in front of the glyph, no initial ray-cylinder intersection exits. In this case, we consider the viewpoint V as the first bracket and set $z'_{\text{near}} = z'_{\text{V}}$. If the viewpoint is inside or passed the glyph, we discard the fragment. Hence, we obtain for the first bracket

$$z'_{\text{near}} = \min\big(z'_{\text{max}},\; z'_{\text{V}}\big), \quad \text{if } z'_{\text{g}} < z'_{\text{V}}. \tag{47}$$

There are two scenarios how a view-ray can exit the bounding cylinder. In the first scenario, the ray exits the cylinder through the rear face at position $(\rho'_{\text{ray}}(-z'_{\text{max}}), \varphi', -z'_{\text{max}})$ in cylindrical glyph coordinates and the second bracket is $z'_{\text{far}} = -z'_{\text{max}}$. In the second scenario, the view-ray exits the cylinder through the side. The corresponding z'-value can be derived by solving $\rho'_{\text{ray}}(z'_{\text{far}}) = \rho'_{\text{max}}$. Hence, we obtain for the second bracket

$$z'_{\text{far}} = \max\left(-z'_{\text{max}},\; z'_{\text{V}} + \frac{\vec{v}_{z'}}{\vec{v}_{\rho'}} \, \rho'_{\text{max}}\right), \quad \text{if } z'_{\text{g}} < z'_{\text{V}}. \tag{48}$$

If the view-ray does not intersect the bounding cylinder or if the viewpoint is inside or past the glyph, the view-ray will not strike the glyph from the outside and we can safely omit the following steps by discarding the fragment. If, however, the view-ray intersects the bounding cylinder, we are provided with a search bracket $(z'_{\text{far}}, z'_{\text{near}})$ to numerically determine the ray-glyph intersection I, if it exits.

4.2.2 Intersection of View-Ray and Spherical Function

In (29) of Section 3.2.2, we have derived function Φ in cylindrical glyph coordinates, with which one can efficiently probe the spatial extent of glyph ψ. Φ renders positive values, if its cylindrical parameters are inside the spherical function ψ, and Φ renders negative values if outside. For each view-ray with given φ', we need to initialize function Φ by determining the polynomial coefficients $c_n^{\text{even}}(\varphi')$ and $c_n^{\text{odd}}(\varphi')$ in (32). For inversion symmetric HARDI glyphs and a truncation of the Laplace expansion at $l_{\text{max}} = 4$, only five coefficients $c_n^{\text{even}}(\varphi')$ need to be determined as specified in Appendix B.

The ray-glyph intersection I is given by the root z'_{I} of

$$\Phi\big(\rho'_{\text{ray}}(z'), \varphi', z'\big) = 0, \tag{49}$$

that is nearest to viewpoint V. We can easily determine the roots of (49) by classical numerical means, since φ' is fixed

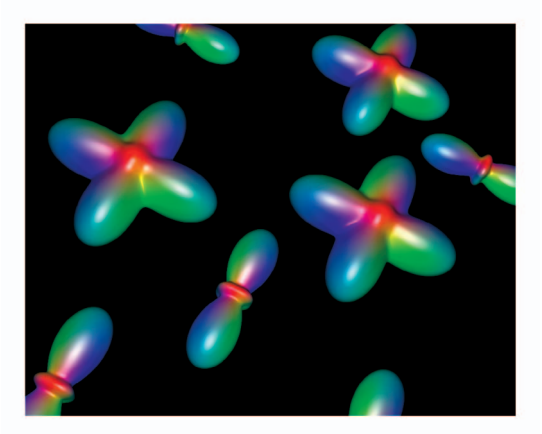

Fig. 6. Close-up of our GPU glyphs showing a detail of a synthetic phantom data set of two crossing fiber bundles under an angle of 90 degrees.

and since search brackets for z' are provided by (47) and (48). We have tested the bisection method and *regula falsi* (false position method) [25]. The latter has proven to be more efficient. The initial sampling of the view-ray is done with a step size $\Delta z'$ of typically 0.1 or about 1 percent of $|z'_{\text{far}}, z'_{\text{near}}|$ starting at z'_{near} and progresses toward z'_{far}. If a sign change in Φ is detected, two refinement steps of the *regula falsi* algorithm are performed. We thereby determine a good approximation of root z'_{I} nearest to viewpoint V. If one, however, reaches the end of the search bracket z'_{far} without detecting a sign change in Φ no intersection I exists and we can omit the subsequent step.

The initial step size $\Delta z'$ and the iteration depth of the *regula falsi* algorithm can be chosen by the user. The values quoted above are appropriate example values. If one choses a larger step size $\Delta z'$ one gains speed, but risks to miss the edges of the glyph. In this case, the appearance of the glyph seems to be intact, but the edges are missing and the glyph is smaller than it should be. A larger *regula falsi* iteration depth will render more accurate values for root z'_{I}, but the visual impact is hardly detectable.

If a casted ray exhibits an intersection with the spherical function ψ, one still needs to determine the normal surface vector at this intersection to provide an appropriate shading for the corresponding pixel.

4.2.3 Normal Vector and Lighting

We compute the normals at the intersection I on the ray-casted surface of the glyph as described in Section 3.3. The implementation is straightforward, since all the equations are given in an explicit form. With the normal vectors we can use standard lighting according to the Phong lighting model to determine the color of the view-ray pixel.

In our specific application example, we apply a color coding commonly used in HARDI and DTI. We map the spherical angles ϑ and φ of an intersection I with respect to the glyph center P to a color, so that red corresponds to mediolateral, green to anteroposterior, and blue to superoinferior orientation. The results are illustrated in Fig. 6.

TABLE 1
Rendering Performance of the Classical Geometry-Based Glyph-Rendering Method Compared with Our Two GPU-Based Ray-Casting Methods

number of glyphs	geometry generation [s]	geometry render [fps]	ray-casting (spherical) [24] [fps]	ray-casting (cylindrical) [fps]
100	2	15	70	260
800	9	4	22	150
9000	52	< 1	7	30
36000			1	8

For the geometry-based rendering, we used 252 vertices per glyph. The step size for the ray-casting measurements was 0.1 and we use 2 refinement steps.

5 Results

Glyphs of spherical functions depict directional distributions in 3-dimensional data sets. As with most 3-dimensional visualizations, interactivity via rotation or change of viewpoint provides the user with prominent clues about the spatial configuration of the data. Thus, the rendering speed that ensures a responsive interactivity is the first performance criteria that we discuss. We then describe the HARDI data used in our validation and address the visual aspects and qualities of the ray-casted glyphs.

5.1 Performance Measurements

The performance measurements of three glyph rendering methods are listed in Table 1. We compare the traditional, geometry-based method with our two ray-casting methods. For the traditional generation of geometry, we used a custom VTK filter whose output was rendered with a standard VTK polydata mapper. Our first ray-casting method [24] is a straightforward implementation that lacks all of the refinements of the second algorithm introduced above. The measurements for all three methods were performed on a PC with an Intel Pentium 4 3.20 GHz CPU, 2 GB RAM, and a GeForce 9,600 GT graphics card with 512 MB of memory and a display of 1262×880 pixels.

Although, we did not optimize the traditional method to efficiently generate and render geometry, the results clearly indicate that our proposed ray-casting methods outperform the geometry-based method. Of course, in all three methods the performance decreases when the number of glyphs increases. But the performance of the geometry-based method decreases drastically when the order of tessellation is increased. This poses a problem since tessellations of fifth order or higher, which are favorable for sufficiently good visual results, cannot be rendered in an interactive way. Furthermore, with the geometry-based method it is not possible to interactively redefine the region-of-interest (ROI). Doing so requires the time-consuming generation of geometry for the glyphs in the new ROI. The same applies when changing parameters, such as scaling and sharpening. The geometry needs to be changed and this cannot be done interactively. However, being able to interact and explore the data in real time is of utmost importance for the researchers in this field and it can make HARDI techniques more attractive for clinical applications.

The first, basic ray-casting method does not align the spherical function with the view axis and, thus, cannot

utilize the more efficient polynomial Φ in cylindrical coordinates. Instead, the spherical function is probed directly. As an outer bound, the first ray-casting method uses a sphere instead of a cylinder and it applies the bisection method instead of *regula falsi* to approximate the ray-glyph intersections.

Table 1 also shows that the refinements in the second ray-casting method lead to a performance boost that consistently triples the frame rate in comparison to the first, more basic ray-casting approach [24].

Three refinements in the second ray-casting algorithm contribute to the higher frame rates:

- the replacement of spherical coordinates by cylindrical coordinates and the alignment of the cylindrical coordinate system with the glyph-observer axis yielding polynomial Φ,
- the use of quads instead of boxes for pixel projection, and
- the use of tighter bounding cylinders instead of spheres.

All three modifications of the second algorithm are intimately connected. The use of quads is only possible in aligned coordinates, and bounding cylinders are only sensible if working in cylindrical coordinates. Nevertheless, we made performance measurements adding each of these three refinements at a time and we observed in the case of a 2,600 glyph data set (in a 757×928 viewport with an initial ray-casting step size of 0.05 and two subsequent step-size refinements) an overall 5-fold increase in speed. The speed increase consisted of a 250 percent fps increase due to the aligned cylindrical coordinates and polynomial ϕ, a 30 percent fps increase due to quads, and a 10 percent fps increase due to tighter cylindrical bounds.

5.2 Diffusion Data Acquisition

For the validation of the proposed HARDI visualization method, the following data sets were used. *Synthetic phantom data*: We generate artificial noiseless synthetic data of a 90 degree crossing according to Soderman and Jönsson [28]. *In vivo data*: DW-MRI acquisition was performed on a healthy volunteer (25-year old female) using a twice refocused spin-echo echo-planar imaging sequence on a Siemens Allegra 3T scanner (Siemens, Erlangen, Germany). The scanning was done using FOV 208×208 mm and voxel size $2.0 \times 2.0 \times 2.0$ mm. Ten horizontal slices were positioned through the body of the corpus callosum and centrum semiovale. Data sets were acquired with 106 directions, each at b-values of 1,000, 1,500, 2,000, and $3{,}000\ \mathrm{s/mm^2}$.

5.3 Visual Aspects

For validation of our proposed methods, we use a software phantom of 90 degree crossings (Fig. 6). The RGB color coding coincides with the orientation of the simulated volume. In the single fiber areas, we clearly observe the "peanut" shape representation of the ODFs. The areas of crossings are correctly represented as well. Here, we immediately observe the advantage of our GPU-based rendering. The quality of the rendered glyphs is independent of the zooming factor exhibiting every level-of-detail. In Fig. 7, we show a region-of-interest on a coronal slice of centrum semiovale. Fibers of the corpus callosum (CC), corona radiata (CR), and superior longitudinal fasciculus (SLF) form a three-fold crossing. This interesting region

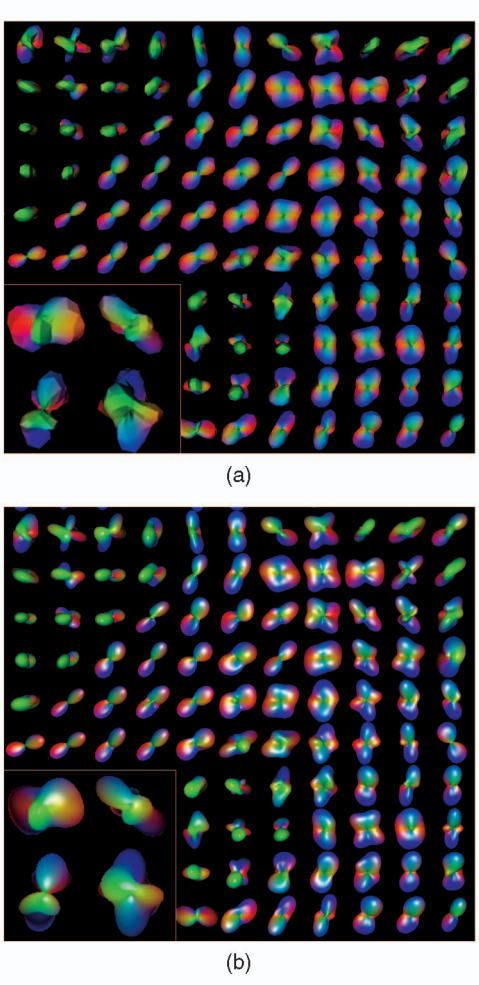

Fig. 7. ODF representations of the DW-MRI data with 106 gradient directions and $\mathrm{b} = 2{,}000\ \mathrm{s/mm^2}$ of a human subject in a ROI defined on a coronal slice in the centrum semiovale. The glyphs are shown for fourth order spherical harmonics, which are min-max normalized. The lower left corners exhibit inlets with a two-fold magnification of four glyphs each. (a) Traditional way of HARDI glyphs rendering with third order icosahedral tessellation so interaction is still possible. (b) Ray-casting of the HARDI glyphs on GPU. (a) Geometry-Based. (b) GPU Ray-Casting.

with known crossings is often targeted for the qualitative analysis of DW-MRI techniques. In Fig. 7a, we show ODF glyphs rendered by the geometry-based method with third order icosahedral tessellation. We have choosen this quality for the geometry-based method, since its frame rate is comparable to our proposed methods. However, although interaction is possible at this level of smoothness (except scaling and sharpening), we can observe a lower visual quality. The same ROI is shown in Fig. 7b represented by our ray-casting method. The increase of visual quality is immediately visible. Crossings become visually more apparent and there is an obvious 3D effect that perceptually helps in the distinction of CC and CR structures.

Both ray-casting methods give results that are substantially better than the geometry-based approach, visually and with respect to performance. With the settings used in the figures, the glyphs look much smoother than with the traditional method even if we use fourth order of tessellation (i.e., 252 vertices per glyph). Furthermore, we can define the current ROI interactively, since no geometry needs to be generated, and we can also change all rendering

parameters interactively. This enables HARDI researchers to easily inspect their PDFs or ODFs and it helps them to verify if their tissue models and fiber tracking algorithms correctly reflect the underlying HARDI data.

When we compare the two ray-casting approaches, we see that the new method, based on cylindrical coordinates, gives substantially higher frame rates than the previous method without compromising the visual quality of the rendered glyphs.

6 Conclusions and Future Work

We presented a fast and flexible CPU and GPU-based method for glyph rendering of spherical functions in general and HARDI data in particular. It is important to state that HARDI is an emerging area, rapidly growing, with new proposed techniques for data modeling, regularization, fiber tracking, and more. Validation is still an open issue. Showing the local information of the reconstructed PDFs, ODFs, or any other spherical function in a fast way is essential for the development of HARDI research. At the moment, to the best of our knowledge, there is no fast and interactive way for HARDI data exploration and visualization. Being able to interactively explore the quality of the obtained data, to detect the interesting areas for further inspection in a fast and reliable way, and to validate the new proposed fiber tracking techniques by supplying the local information will undoubtedly move the HARDI methodology from pure research toward a clinically applicable, new, and better DW-MRI technique for depicting the white matter structure of the brain.

We stress that this visualization is aimed at HARDI researchers and not physicians, due to the complexity and cluttering of the data. This GPU-accelerated visualization will improve the working condition of researchers, allowing them smooth and interactive data exploration. The rendering technique is based on the spherical harmonic approximation applicable to any function on a sphere. The proposed visualization technique is, therefore, not HARDI-specific. Any data that is given by spherical functions can be visualized this way.

It will be useful to extend the proposed visualization algorithm with specialized coloring and sharpening methods that will highlight maxima of a spherical function and possibly highlight intervoxel coherence. We will also integrate the new algorithm into our visualization tool for DTI so that we can combine different imaging modalities.

Appendix A

Wigner Matrices

According to the Euler rotations around the z, y, and z-axis, the Wigner rotation matrices are the concatenation of three rotation matrices.

$$\mathbf{D}_l^{m\,m'}(\alpha,\beta,\gamma) \;:=\; e^{+im\alpha}\, d_l^{m\,m'}(\beta)\, e^{i\gamma m'}. \tag{50}$$

The rotations around the z-axis are given by $c_l^{mm'}(\alpha) = e^{i\,m\,\alpha}\,\delta_{mm'}$ and by $c_l^{mm'}(\gamma) = e^{i\,m\,\gamma}\,\delta_{mm'}$. The rotation around the y-axis is given by the reduced Wigner function $d_l^{mm'}(\beta)$.

$$\begin{aligned} d_l^{mm'}(\beta) = &\sqrt{\frac{(l-m)!(l+m)!}{(l-m')!(l+m')!}}\,\sin^{m-m'}\left(\frac{\beta}{2}\right) \\ &\times \cos^{m'+m}\left(\frac{\beta}{2}\right) P_{l-m}^{(m-m',m'+m)}(\cos\beta), \end{aligned} \tag{51}$$

where $P_n^{(a,b)}(z)$ stands for the Jacobi polynomial.

Here, we provide matrix coefficients $\tilde{c}_l^{mm'}(\alpha)$ and the reduced Wigner function $\tilde{d}_l^{mm'}(\beta)$ for real-valued spherical harmonics for l up to 4.

$$\tilde{c}_l^{mm'}(\alpha) := \cos(m\,\alpha)\,\delta_{mm'} - \mathrm{sign}(m)(-1)^m \sin(m\,\alpha)\,\delta_{m\,-m'}. \tag{52}$$

All values of $\tilde{d}_l^{mm'}(\beta)$ for $l \le 4$ are 0 unless listed below:

$$\tilde{d}_0^{00}(\beta) = 1,$$

$$\begin{aligned} \tilde{d}_1^{1\text{-}1}(\beta) = \tilde{d}_1^{00}(\beta) &= \cos\beta, \\ \tilde{d}_1^{10}(\beta) = -\tilde{d}_1^{0\text{-}1}(\beta) &= -\sin\beta, \\ \tilde{d}_1^{11}(\beta) &= 1, \end{aligned}$$

$$\begin{aligned} \tilde{d}_2^{2\text{-}2}(\beta) &= \frac{1}{4}(\cos 2\beta + 3) \\ \tilde{d}_2^{2\text{-}1}(\beta) = -\tilde{d}_2^{1\text{-}2}(\beta) &= -\sin\beta\,\cos\beta \\ \tilde{d}_2^{20}(\beta) = \tilde{d}_2^{0\text{-}2}(\beta) &= \frac{\sqrt{3}}{2}\sin^2\beta \\ \tilde{d}_2^{1\text{-}1}(\beta) &= \cos 2\beta \\ \tilde{d}_2^{10}(\beta) = -\tilde{d}_2^{0\text{-}1}(\beta) &= -\sqrt{3}\cos\beta\sin\beta. \\ \tilde{d}_2^{00}(\beta) &= \frac{1}{4}(1 + 3\cos 2\beta) \\ \tilde{d}_2^{11}(\beta) = \tilde{d}_2^{22}(\beta) &= \cos\beta \\ \tilde{d}_2^{12}(\beta) = -\tilde{d}_2^{21}(\beta) &= -\sin\beta \end{aligned}$$

$$\begin{aligned} \tilde{d}_3^{3\text{-}3}(\beta) &= \frac{1}{16}(15\cos\beta + \cos 3\beta) \\ \tilde{d}_3^{3\text{-}2}(\beta) = -\tilde{d}_3^{2\text{-}3}(\beta) &= -\frac{1}{8}\sqrt{\frac{3}{2}}(5\sin\beta + \sin 3\beta) \\ \tilde{d}_3^{3\text{-}1}(\beta) = \tilde{d}_3^{1\text{-}3}(\beta) &= \frac{\sqrt{15}}{4}\cos\beta\sin^2\beta \\ \tilde{d}_3^{30}(\beta) = -\tilde{d}_3^{0\text{-}3}(\beta) &= -\frac{1}{2}\sqrt{\frac{5}{2}}\sin^3\beta \\ \tilde{d}_3^{2\text{-}2}(\beta) &= \frac{1}{8}(5\cos\beta + 3\cos 3\beta) \\ \tilde{d}_3^{2\text{-}1}(\beta) = -\tilde{d}_3^{1\text{-}2}(\beta) &= \frac{1}{8}\sqrt{\frac{5}{2}}(\sin\beta - 3\sin 3\beta) \end{aligned}$$

$$\begin{aligned} \tilde{d}_3^{20}(\beta) = \tilde{d}_3^{0\text{-}2}(\beta) &= \frac{\sqrt{15}}{2}\cos\beta\sin^2\beta \\ \tilde{d}_3^{1\text{-}1}(\beta) &= \frac{1}{16}(\cos\beta + 15\cos 3\beta) \\ \tilde{d}_3^{10}(\beta) = -\tilde{d}_3^{0\text{-}1}(\beta) &= -\frac{1}{8}\sqrt{\frac{3}{2}}(\sin\beta + 5\sin 3\beta) \\ \tilde{d}_3^{00}(\beta) &= \frac{1}{8}(3\cos\beta + 5\cos 3\beta) \end{aligned}$$

$$
\begin{aligned}
\tilde{d}_3^{11}(\beta) &= \frac{1}{8}(3+5\cos 2\beta)\\
\tilde{d}_3^{12}(\beta) &= -\tilde{d}_3^{21}(\beta) = -\sqrt{\frac{5}{2}}\cos\beta\sin\beta\\
\tilde{d}_3^{13}(\beta) &= \tilde{d}_3^{31}(\beta) = \frac{\sqrt{15}}{4}\sin^2\beta\\
\tilde{d}_3^{22}(\beta) &= \cos 2\beta\\
\tilde{d}_3^{23}(\beta) &= -\tilde{d}_3^{32}(\beta) = -\sqrt{\frac{3}{2}}\cos\beta\sin\beta\\
\tilde{d}_3^{33}(\beta) &= \frac{1}{8}(5+3\cos 2\beta)
\end{aligned}
$$

$$
\begin{aligned}
\tilde{d}_4^{4\text{-}4}(\beta) &= \frac{1}{64}(35+28\cos 2\beta+\cos 4\beta)\\
\tilde{d}_4^{4\text{-}3}(\beta) &= -\tilde{d}_4^{3\text{-}4}(\beta) = -\frac{1}{16\sqrt{2}}(14\sin 2\beta+\sin 4\beta)\\
\tilde{d}_4^{4\text{-}2}(\beta) &= \tilde{d}_4^{2\text{-}4}(\beta) = \frac{\sqrt{7}}{8}(3+\cos 2\beta)\sin^2\beta\\
\tilde{d}_4^{4\text{-}1}(\beta) &= -\tilde{d}_4^{1\text{-}4}(\beta) = -\frac{1}{2}\sqrt{\frac{7}{2}}\cos\beta\sin^3\beta\\
\tilde{d}_4^{40}(\beta) &= \tilde{d}_4^{0\text{-}4}(\beta) = \frac{\sqrt{35}}{8}\sin^4\beta\\
\tilde{d}_4^{3\text{-}3}(\beta) &= \frac{1}{8}(7\cos 2\beta+\cos 4\beta)\\
\tilde{d}_4^{3\text{-}2}(\beta) &= -\tilde{d}_4^{2\text{-}3}(\beta) = -\sqrt{\frac{7}{2}}\cos^3\beta\sin\beta\\
\tilde{d}_4^{3\text{-}1}(\beta) &= \tilde{d}_4^{1\text{-}3}(\beta) = \frac{\sqrt{7}}{4}(1+2\cos 2\beta)\sin^2\beta\\
\tilde{d}_4^{30}(\beta) &= -\tilde{d}_4^{0\text{-}3}(\beta) = -\frac{1}{2}\sqrt{\frac{35}{2}}\cos\beta\sin^3\beta\\
\tilde{d}_4^{2\text{-}2}(\beta) &= \frac{1}{16}(5+4\cos 2\beta+7\cos 4\beta)\\
\tilde{d}_4^{2\text{-}1}(\beta) &= -\tilde{d}_4^{1\text{-}2}(\beta) = \frac{1}{8\sqrt{2}}(2\sin 2\beta-7\sin 4\beta)\\
\tilde{d}_4^{20}(\beta) &= \tilde{d}_4^{0\text{-}2}(\beta) = \frac{\sqrt{5}}{8}(5+7\cos 2\beta)\sin^2\beta\\
\tilde{d}_4^{1\text{-}1}(\beta) &= \frac{1}{8}(\cos 2\beta+7\cos 4\beta)\\
\tilde{d}_4^{10}(\beta) &= -\tilde{d}_4^{0\text{-}1}(\beta) = -\frac{1}{16}\sqrt{\frac{5}{2}}(2\sin 2\beta+7\sin 4\beta)\\
\tilde{d}_4^{00}(\beta) &= \frac{1}{64}(9+20\cos 2\beta+35\cos 4\beta)\\
\tilde{d}_4^{11}(\beta) &= \frac{1}{16}(9\cos\beta+7\cos 3\beta)\\
\tilde{d}_4^{12}(\beta) &= -\tilde{d}_4^{21}(\beta) = -\frac{1}{8\sqrt{2}}(3\sin\beta+7\sin 3\beta)
\end{aligned}
\tag{53}
$$

$$
\begin{aligned}
\tilde{d}_4^{13}(\beta) &= \tilde{d}_4^{31}(\beta) = \frac{3\sqrt{7}}{4}\cos\beta\sin^2\beta\\
\tilde{d}_4^{14}(\beta) &= -\tilde{d}_4^{41}(\beta) = -\frac{1}{2}\sqrt{\frac{7}{2}}\sin^3\beta\\
\tilde{d}_4^{22}(\beta) &= \frac{1}{8}(\cos\beta+7\cos 3\beta)\\
\tilde{d}_4^{23}(\beta) &= -\tilde{d}_4^{32}(\beta) = \frac{1}{8}\sqrt{\frac{7}{2}}(\sin\beta-3\sin 3\beta)\\
\tilde{d}_4^{24}(\beta) &= \tilde{d}_4^{42}(\beta) = \frac{\sqrt{7}}{2}\cos\beta\sin^2\beta\\
\tilde{d}_4^{33}(\beta) &= \frac{1}{16}(7\cos\beta+9\cos 3\beta)\\
\tilde{d}_4^{34}(\beta) &= -\tilde{d}_4^{43}(\beta) = -\frac{1}{8\sqrt{2}}(7\sin\beta+3\sin 3\beta)\\
\tilde{d}_4^{44}(\beta) &= \frac{1}{8}(7\cos\beta+\cos 3\beta).
\end{aligned}
\tag{54}
$$

APPENDIX B
POLYNOMIAL COEFFICIENTS

For $l_{\max}=4$, the polynomial coefficients in (32) are determined by expanding polynomials (28). Note that the coefficients are only valid for a Laplace expansion with truncation at $l_{\max}=4$. Laplace expansion with different truncations will result in different coefficients.

$$
\begin{aligned}
c_0^{\text{even}}(\varphi') &= \frac{1}{2\sqrt{\pi}}\left(\tilde{b}_0^0+\sqrt{5}\,\tilde{b}_2^0+3\,\tilde{b}_4^0\right),\\
c_1^{\text{even}}(\varphi') &= \sqrt{\frac{5}{4\pi}}\Big(\left(\sqrt{3}\,\tilde{b}_2^{-1}+3\sqrt{2}\,\tilde{b}_4^{-1}\right)\cos(\varphi)\\
&\quad -\left(\sqrt{3}\,\tilde{b}_2^{1}+3\sqrt{2}\,\tilde{b}_4^{1}\right)\sin(\varphi)\Big)\\
c_2^{\text{even}}(\varphi') &= \frac{\sqrt{5}}{4\sqrt{\pi}}\Big(\left(\sqrt{3}\,\tilde{b}_2^{2}+9\,\tilde{b}_4^{2}\right)\sin(2\varphi)\\
&\quad +\sqrt{5}\left(\sqrt{3}\,\tilde{b}_2^{-2}+9\,\tilde{b}_4^{-2}\right)\cos(2\varphi)\\
&\quad +4\,\tilde{b}_0^0+\sqrt{5}\,\tilde{b}_2^0-18\,\tilde{b}_4^0\Big)\\
c_3^{\text{even}}(\varphi') &= \frac{1}{8}\sqrt{\frac{5}{\pi}}\Big(\left(9\sqrt{2}\,\tilde{b}_4^{1}-4\sqrt{3}\,\tilde{b}_2^{1}\right)\sin(\varphi)\\
&\quad +3\sqrt{14}\left(b_4^{-3}\cos(3\varphi)-\tilde{b}_4^{3}\sin(3\varphi)\right)\\
&\quad +\left(4\sqrt{3}\,\tilde{b}_2^{-1}-9\sqrt{2}\,\tilde{b}_4^{-1}\right)\cos(\varphi)\Big)\\
c_4^{\text{even}}(\varphi') &= \frac{1}{16\sqrt{\pi}}\left(3\sqrt{35}\left(b_4^4\sin(4\varphi)+\tilde{b}_4^{-4}\cos(4\varphi)\right)\right.\\
&\quad +2\sqrt{5}\left(2\sqrt{3}\,\tilde{b}_2^{2}-3\,\tilde{b}_4^{2}\right)\sin(2\varphi)\\
&\quad +2\sqrt{5}\left(2\sqrt{3}\,\tilde{b}_2^{-2}-3\,\tilde{b}_4^{-2}\right)\cos(2\varphi)\\
&\quad \left.+8\,\tilde{b}_0^0-4\sqrt{5}\,\tilde{b}_2^0+9\,\tilde{b}_4^0\right).
\end{aligned}
\tag{55}
$$

$$
\begin{aligned}
c_0^{\text{odd}}(\varphi') &= \frac{1}{\sqrt{4\pi}}\left(\sqrt{3}\,\tilde{b}_1^0+\sqrt{7}\,\tilde{b}_3^0\right)\\
c_1^{\text{odd}}(\varphi') &= \sqrt{\frac{3}{4\pi}}\Big(\left(b_1^{-1}+\sqrt{14}\,\tilde{b}_3^{-1}\right)\cos(\varphi)\\
&\quad -\left(b_1^{1}+\sqrt{14}\,\tilde{b}_3^{1}\right)\sin(\varphi)\Big)\\
c_2^{\text{odd}}(\varphi') &= \frac{1}{4\sqrt{\pi}}\left(\sqrt{105}\,\tilde{b}_3^{2}\sin(2\varphi)+\sqrt{105}\,\tilde{b}_3^{-2}\cos(2\varphi)\right.\\
&\quad \left.-3\sqrt{7}\,\tilde{b}_3^0+2\sqrt{3}\,\tilde{b}_1^0\right)\\
c_3^{\text{odd}}(\varphi') &= \frac{1}{8\sqrt{\pi}}\Big(\sqrt{3}\left(\sqrt{14}\,\tilde{b}_3^{1}-4\,\tilde{b}_1^{1}\right)\sin(\varphi)\\
&\quad +\sqrt{70}\left(\tilde{b}_3^{-3}\cos(3\varphi)-\tilde{b}_3^{3}\sin(3\varphi)\right)\\
&\quad +\sqrt{3}\left(4\,\tilde{b}_1^{-1}-\sqrt{14}\,\tilde{b}_3^{-1}\right)\cos(\varphi)\Big).
\end{aligned}
\tag{56}
$$

APPENDIX C
CYLINDRICAL BOUND

The following tables contain the maximal extents of the real-valued spherical harmonics $\tilde{Y}_l^m$ along the cylindrical radius ρ and depth z. The maximal radii ${\rho_{\max}}_l^m$ and maximal depths ${z_{\max}}_l^m$ have been determined numerically.

C.1 Maximal Cylindrical Radii ${\rho_{\max}}_l^m$

$$ {\rho_{\max}}_l^m = {\rho_{\max}}_l^{-m}. \tag{57} $$

${\rho_{\max}}_l^m$	$m=0$	1	2	3	4
$l=0$	0.282095				
1	0.244301	0.345494			
2	0.315392	0.297354	0.386274		
3	0.301382	0.323180	0.331900	0.417224	
4	0.317357	0.311653	0.334523	0.358249	0.442533

C.2 Maximal Cylindrical Heights ${z_{\max}}_l^m$

$$ {z_{\max}}_l^m = {z_{\max}}_l^{-m}. \tag{58} $$

${z_{\max}}_l^m$	$m=0$	1	2	3	4
$l=0$	0.282095				
1	0.488603	0.172747			
2	0.630783	0.297354	0.148677		
3	0.746353	0.388357	0.255496	0.135497	
4	0.846284	0.459798	0.335275	0.232690	0.126660

ACKNOWLEDGMENTS

This work has been supported in part by the Dutch Technology foundation (STW project 07487).

REFERENCES

[1] P.J. Basser, J. Mattiello, and D. Lebihan, "MR Diffusion Tensor Spectroscopy and Imaging," *Biophysical J.*, vol. 66, no. 1, pp. 259-267, Jan. 1994.
[2] H.C. Torrey, "Bloch Equations with Diffusion Terms," *Physical Rev.*, vol. 194, pp. 563-565, 1956.
[3] D.S. Tuch, "Diffusion MRI of Complex Tissue Structure," PhD dissertation, Harvard Univ., 2002.
[4] L.R. Frank, "Anisotropy in High Angular Resolution Diffusion-Weighted MRI," *Magnetic Resonance in Medicine*, vol. 45, no. 6, pp. 935-939, 2001.
[5] D. Tuch, "Q-Ball Imaging," *Magnetic Resonance in Medicine*, vol. 52, pp. 1358-1372, 2004.
[6] E. Özarslan, T.M. Shepherd, B.C. Vemuri, S.J. Blackband, and T.H. Mareci, "Resolution of Complex Tissue Microarchitecture Using the Diffusion Orientation Transform (DOT)," *NeuroImage*, vol. 36, no. 3, pp. 1086-1103, July 2006.
[7] E. Özarslan and T.H. Mareci, "Generalized Diffusion Tensor Imaging and Analytical Relationships between Diffusion Tensor Imaging and High Angular Resolution Diffusion Imaging," *Magnetic Resonance in Medicine*, vol. 50, no. 5, pp. 955-965, 2003.
[8] R. Deriche and M. Descoteaux, "Splitting Tracking through Crossing Fibers: Multidirectional Q-Ball Tracking," *Proc. Fourth IEEE Int'l Symp. Biomedical Imaging: From Nano to Macro (ISBI)*, pp. 756-759, 2007.
[9] M. Perrin, C. Poupon, Y. Cointepas, B. Rieul, N. Golestani, D. Rivière, A. Constantinesco, D.L. Bihan, and J.-F. Mangin, "Fiber Tracking in Q-Ball Fields Using Regularized Particle Trajectories," *Proc. 19th Int'l Conf. Information Processing in Medical Imaging*, vol. 19, pp. 52-63, 2005.
[10] S. Jbabdi, M.W. Woolrich, J.L.R. Andersson, and T.E.J. Behrens, "A Bayesian Framework for Global Tractography," *NeuroImage*, vol. 37, no. 1, pp. 116-129, Aug. 2007.
[11] G. Kindlmann, "Visualization and Analysis of Diffusion Tensor Fields," PhD dissertation, School of Computing, Univ. of Utah, Sept. 2004.
[12] G. Kindlmann and C.-F. Westin, "Diffusion Tensor Visualization with Glyph Packing," *IEEE Trans. Visualization and Computer Graphics*, vol. 12, no. 5, pp. 1329-1336, Sept.-Oct. 2006.
[13] M. Hlawitschka, G. Scheuermann, and B. Hamann, "Interactive Glyph Placement for Tensor Fields," *Advances in Visual Computing*, pp. 331-340, Springer, 2007.
[14] N.A. for Medical Image Computing. Slicer. http://www.na-mic.org/Wiki/index.php/Slicer, 2009.
[15] I.P.O. 2004, "QBall Plug-In for Slicer3D," http://www-sop.inria.fr/odyssee/en/software/QBallSlicer, 2009.
[16] U.C.L.C.S. Department, "Camino," http://www.cs.ucl.ac.uk/research/medic/camino/, 2009.
[17] D.W. Shattuck, M.-C. Chiang, M. Barysheva, K.L. McMahon, G.I. de Zubicaray, M. Meredith, M.J. Wright, A.W. Toga, and P.M. Thompson, "Visualization Tools for High Angular Resolution Diffusion Imaging," *Proc. Medical Image Computing and Computer-Assisted Intervention (MICCAI '08)*, pp. 298-305, 2008.
[18] T. Schultz, *HARDI Glyph Rendering*, Max-Planck-Institut für Informatik, Saarbrücken, personal comm., June 2009.
[19] M. Hlawitschka, G. Scheuermann, A. Anwander, T. Knösche, M. Tittgemeyer, and B. Hamann, "Tensor Lines in Tensor Fields of Arbitrary Order," *Proc. Third Int'l Symp. Visual Computing (ISVC)*, vol. 1, pp. 341-350, 2007.
[20] S. Gumhold, "Splatting Illuminated Ellipsoids with Depth Correction," *Proc. Vision, Modelling, and Visualization*, pp. 245-252, 2003.
[21] C. Sigg, T. Weyrich, M. Botsch, and M. Gross, "GPU-Based Ray-Casting of Quadratic Surfaces," *Proc. Eurographics Symp. Point-Based Graphics*, pp. 59-65, 2006.
[22] S.E. Mario Hlawitschka and G. Scheuermann, "Fast and Memory Efficient GPU-Based Rendering of Tensor Data," *Proc. IADIS Int'l Conf. Computer Graphics and Visualization '08*, pp. 28-30, 2008.
[23] G. Kindlmann, "Superquadric Tensor Glyphs," *Proc. Joint Eurographics/IEEE TCVG Symp. Visualization*, O. Deussen, C. Hansen, D. Keim, and D. Saupe, eds., pp. 147-154, 2004.
[24] T. Peeters, V. Prčkovska, M. van Almsick, A. Vilanova, and B. ter Haar Romeny, "Fast and Sleek Glyph Rendering for Interactive Hardi Data Exploration," *Proc. IEEE Pacific Visualization Symp.*, pp. 153-160, 2009.
[25] W. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical Recipes*, third ed., Cambridge Univ. Press, 2007.
[26] D.A. Varshalovich, A.N. Moskalev, and V.K. Khersonskii, *Quantum Theory of Angular Momentum*. World Scientific Publishing Co., 1988.
[27] M. Descoteaux, E. Angelino, S. Fitzgibbons, and R. Deriche, "Regularized, Fast and Robust Analytical Q-Ball Imaging," *Magnetic Resonance in Medicine*, vol. 58, pp. 497-510, 2007.
[28] O. Söderman and B. Jönsson, "Restricted Diffusion in Cylindrical Geometry," *J. Magnetic Resonance, Series A*, vol. 117, no. 1, pp. 94-97, 1995.

Markus van Almsick received the diplom (MSc degree) in physics from the Technische Universität München, Germany, in 1990. He worked for the University of Illinois, Wolfram Research Inc., and the Max-Planck Institute for Biophysics in Frankfurt before joining the Biomedical Image Analysis group in the Department of Biomedical Engineering at the Technische Universiteit Eindhoven. There, he received the PhD degree in 2007 developing "Context Models of Lines and Contours." His research interests are image analysis, computer algebra, and fundamental physics.

Tim H.J.M. Peeters received the MSc degree from Eindhoven University of Technology in 2004 and the PhD degree from the Department of Biomedical Engineering at Eindhoven University in 2009. His current research involves GPU-based visualization techniques for interactive exploration of diffusion-weighted MRI data.

Vesna Prčkovska received the diploma engineering degree (masters level equivalent) at the faculty of Electrical Engineering from the University Ss Cyril and Methodius, R. Macedonia, in 2006. She is currently a PhD student at Technical University of Eindhoven at the Department of Biomedical Engineering. Her research interests are medical visualization, multifield visualization, and modeling and processing of high angular resolution imaging (HARDI) data.

Anna Vilanova received the PhD degree from Vienna University of Technology in 2001. The title of her PhD thesis was visualization techniques for virtual endoscopy. She is assistant professor at the Biomedical Image Analysis group of the Biomedical Engineering Department at the Eindhoven University of Technology. Currently, she is leading a research group with several PhDs in the subject of multivalued image analysis and visualization. Her research interests include medical visualization, volume visualization, multivalued visualization, and medical image analysis.

Bart ter Haar Romeny received the MSc degree in applied physics from Delft University of Technology in 1978 and the PhD degree in biophysics from Utrecht University in 1983. Subsequently, he became the principal physicist of the Utrecht University Hospital Radiology Department and from 1986-1989, clinical project leader of the Dutch PACS project. He was cofounder and associate professor at the Image Sciences Institute (ISI) of Utrecht University (1989-2001). He is full professor at Eindhoven University of Technology, heading the Biomedical Image Analysis (BMIA) group in the Department of Biomedical Engineering. He has experience with biomedical image analysis and computer vision research for more than 20 years. His research interests include quantitative medical image analysis, its foundations, and clinical applications. His interests are, in particular, the mathematical modeling of visual perception and applying this knowledge in operational computer-aided diagnosis systems.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.