Fast and Sleek Glyph Rendering for Interactive HARDI Data Exploration

T.H.J.M. Peeters, V. Prčkovska, M. van Almsick, A. Vilanova, B.M. ter Haar Romeny

ABSTRACT

High angular resolution diffusion imaging (HARDI) is an emerging magnetic resonance imaging (MRI) technique that overcomes some decisive limitations of its predecessor diffusion tensor imaging (DTI). HARDI can resolve locally more than one direction in the diffusion pattern of water molecules and thereby opens up the opportunity to display and track crossing fibers. Showing the local structure of the reconstructed, angular probability profiles in a fast, detailed, and interactive way can improve the quality of the research in this area and help to move it into clinical application. In this paper we present a novel approach for HARDI glyph visualization or, more generally, for the visualization of any function that resides on a sphere and that can be expressed by a Laplace series. Our GPU-accelerated glyph rendering improves the performance of the traditional way of HARDI glyph visualization as well as the visual quality of the reconstructed data, thus offering interactive HARDI data exploration of the local structure of the white brain matter in-vivo. In this paper we exploit the capabilities of modern GPUs to overcome the large, processor-intensive and memory-consuming data visualization.

Index Terms: I.3.3 [Computer Graphics]: Viewing Algorithms—; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—; J.3 [Computer Applications]: Life and Medical Sciences—

1 INTRODUCTION

Diffusion tensor imaging (DTI) is a popular magnetic resonance imaging (MRI) technique that provides a unique view of the fiber structure of the white brain matter in-vivo. Introduced more than a decade ago [2], this technique is based on calculating symmetric, positive-definite diffusion tensors (DT) from a minimum of seven diffusion weighted images to obtain a second order approximation of the probability density functions (PDF) that describes the local diffusivity of the water molecules. One of the most common applications of DTI is the prediction of fiber orientation specified by the principal eigenvector of the DT, thus determining the trajectories of fiber bundles or neuronal connections between regions in the brain.

DTI has been thoroughly explored and its potentials and limitations are well known. Due to the crude assumption that a simple 3D gaussian probability density function can model the diffusion process, DTI is limited to recover structures with at most one direction. In areas with more complex intravoxel heterogeneity (i.e. fiber crossing, kissing, divergence etc.), the second order DT approximation fails.

To overcome this limitation of DTI, more sophisticated models were introduced, known as high angular resolution diffusion imaging (HARDI) [27]. In HARDI about sixty to a few hundred diffusion gradients are scanned that together sample a sphere of given radius [6] in order to reconstruct the true PDF. This PDF-measurement is model-free and can recover the diffusion of water molecules of any underlying fiber population. Popular analysis techniques that transform the diffusion weighted data to certain probability function are Q-ball imaging [26], diffusion orientation transform (DOT) [20] and high order tensors (HOT) [19]. The result produced by the above mentioned techniques is always given in the form of a spherical function $\psi(\theta,\phi)$ that characterizes the local intra-voxel fiber structure. This function on a sphere can be represented with a Laplace series.

$$\psi(\theta,\phi) = \sum_{l=0}^{l_{max}} \sum_{m=-l}^{l} a_l^m Y_l^m(\theta,\phi) \,, \tag{1}$$

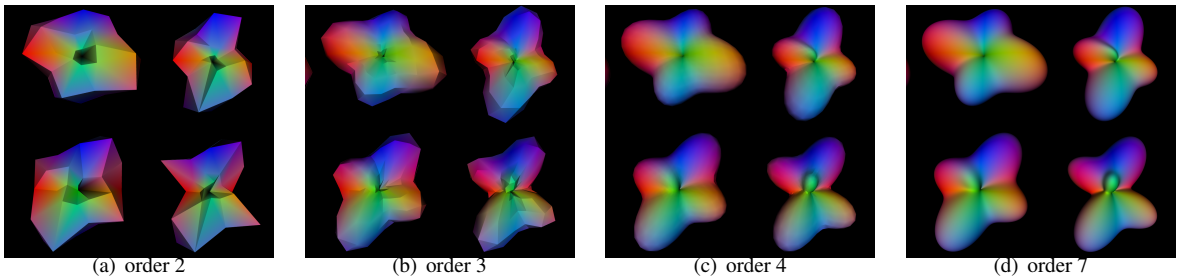
where a_l^m are spherical harmonic coefficients determined via a spherical harmonic transformation, Y_l^m represent the spherical harmonics, and l_{max} is the order of truncation of the Laplace series.

The most common way of representing HARDI data is the iconical representation, better known as glyphs. Each glyph corresponds to the local PDF on a sphere [20] or orientation distribution function (ODF) [27] calculated by the above mentioned techniques. The glyphs are given by the surface of the function graph $\{(r = \psi(\theta,\phi),\theta,\phi) \mid \theta \in [0,\pi], \phi \in [0,2\pi]\}$ defined on the spherical domain and plotted in radial direction. Recently, one can observe an increase in research activity regarding deterministic and probabilistic HARDI fiber tracking algorithms [4, 21, 13]. 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 disadvantages, such as choice of initialization, sensitivity with respect to the estimated principal direction, lack of connectivity information between regions of the brain (deterministic), high computationally expense (probabilistic), and most importantly the lack of thorough validation which is needed before fiber tracking can be used in clinical applications. Knowledge about the local 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 according to Eq.1. This visualization is slow and memory consuming, especially if one wants to achieve smooth surfaces. For very smooth surfaces an icosahedral tessellation of 16^{th} order is used, which generates 2256 points on a sphere. Thus, the visualization of a full brain slice and the interactive inspection of the data is practically impossible with this approach. 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 displaced, as is shown in Fig. 1.

There are many HARDI related research questions where fast and reliable glyph visualization is useful. As mentioned, in the development of fiber tracking algorithms, it is useful to visualize the local information to validate whether the fiber tracks follow the underlying information. Because the signal-to-noise ratio can be low smoothing algorithms on the spherical function are of high importance, but these algorithms often have many parameters to tune. Being able to see the output in a fast manner can benefit the research in this area.

In this paper we present a novel approach for fast detailed and accurate rendering of of high-dimensional diffusion-weighted (DW) data. After giving an overview of related work on HARDI visualization and GPU-based glyph rendering in section 2, we present the mathematical tools for visualizing spherical functions in section 3.



(a) order 2 (b) order 3 (c) order 4 (d) order 7

Figure 1: Traditional way of HARDI glyph visualization. The glyphs' surface is getting 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 glyphs' surface gets smoother.

In section 4 we show how to apply these tools to implement a GPU-based ray casting method for rendering HARDI glyphs. Analysis of the visual aspects and performance measurements follow in section 5. Finally, in section 6 we conclude this manuscript and address challenges in future work.

2 Related Work

In DTI, glyphs are the basic tool to visualize the local structure. As mentioned in the previous section, the diffusion PDF is approximated by a second-order, symmetric, positive-definite tensor and therefore tensor glyphs are the most useful tool for representing the full tensor information. Commonly used shapes for visualizing the DTs are cuboids, ellipsoids and superquadrics [15]. There has been recent work in glyph packing algorithms by Kindlmann [16] and Hlawitschka [11], but these algorithms change the original grid positions of the local information 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 [1] with the Qball plug-in [28] and Camino [3]. The latter one has very limited visualization capabilities since its core purpose is to process HARDI data. Recently, Shattuck et al. [23] 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 these 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. The interaction is reported as 10 frames/sec on a single brain data slice with 225 samples on a sphere that approximates 4^{th} order icosahedral tessellation. In their work, this limitation in rendering performance is being mentioned and GPU programming is suggested as a solution. A new and interesting work by Hlawitschka et al. [10] address 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 high order tensors (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 the rendering of glyphs where ray casting on the GPU has been applied. Application areas include mathematics [7] and molecule rendering [24]. 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, Hlawitschka et al. [9] used GPU ray casting to render superquadric tensor glyphs [14]. The dimensionality of HARDI data, and thus the number of parameters that define the shape of our glyph is much higher than in the above mentioned methods. This leads to more elaborate computations for the intersection of the view ray with the glyph and for the normal vector in this intersection. Using the properties of the spherical harmonics, we show how to solve these problems on the GPU in order to obtain a fast and accurate visualization of HARDI data that can be used in an interactive manner.

3 Spherical Harmonics for HARDI

We being this section with a very brief introduction to spherical harmonics (SH) and describe how to compute properties that are needed for the ray-casting algorithm. In section 3.1 we explain why SHs are used and how to obtain real-valued SHs. Then, in section 3.2 we show how to compute an upper radial bound for the linear combination of SHs, which we will use later to generate bounding spheres for the ray-casting algorithm. Finally, in section 3.3 we show how to compute the normals of a spherical function needed in the surface shading of the glyph.

3.1 Spherical Harmonics

Spherical harmonics $Y_l^m(\theta,\phi)$ (SH) [17] form a complete orthogonal system for arbitrary spherical functions $\psi(\theta,\phi)$, which is expressed by the Laplace series

$$\psi(\theta,\phi) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} a_l^m Y_l^m(\theta,\phi) \; . \tag{2}$$

The expansion of spherical functions $\psi(\theta,\phi)$ via the Laplace series is practical, since spherical harmonics Y_l^m possess several beneficial properties. The most important property with respect to HARDI is the fact that SHs are part of the solution of the homogeneous diffusion equation $\partial_t p(r,\theta,\phi,t) = \kappa \Delta p(r,\theta,\phi,t)$ that governs the underlying physical process. Solving the diffusion equation via the separation of variables [22] with $p(r,\theta,\phi,t) = T(t)R(r)\psi(\theta,\phi)$, one obtains for the angular part of the diffusion equation $\Delta_{S_2}\psi(\theta,\phi) = -l(l+1)\;\psi(\theta,\phi)$, with Δ_{S_2} being the Laplace operator on a sphere S_2 as defined in Eq. 3. Thus, the SHs are the eigenfunctions of this angular diffusion equation.

$$\begin{aligned} \Delta_{S_2} Y_l^m(\theta,\phi) &= \left(\frac{1}{\sin^2\theta}\partial_\phi^2 + \frac{1}{\sin\theta}\partial_\theta \sin\theta \partial_\theta \right) Y_l^m(\theta,\phi) \\ &= -l(l+1)\, Y_l^m(\theta,\phi) \; . \end{aligned} \tag{3}$$

Furthermore, one can view the Laplace series Eq. 2 as an inverse spherical Fourier transformation. As show in Eq. 3, the SHs are the Laplacian eigenfunction in spherical coordinates. On the other hand, $e^{i\vec{\omega}\cdot\vec{x}}$, $\cos(\vec{\omega}\cdot\vec{x})$, 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 SHs

form the basis functions of a spherical Fourier transformation, the spherical Fourier coefficients being a_l^m.

$$a_l^m \;=\; \int_0^{2\pi}\int_0^{\pi} \psi(\theta,\phi)\,\overline{Y_l^m}(\theta,\phi)\,\sin\theta\,d\theta\,d\phi\;. \tag{4}$$

The inverse spherical Fourier transformation is given by the Lapace series (see Eq. 2).

The SHs are explicitly defined via associated Legendre polynomials P_l^m.

$$Y_l^m(\theta,\phi) \;=\; e^{im\phi}\sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}}\,P_l^m(\cos\theta)\;, \tag{5}$$

with $\theta\in[0,\pi], \phi\in[0,2\pi]$ and integer indices $l\geq 0$ and $|m|\leq l$.

The SHs as defined in Eq. 5 are complex-valued due to the factor $e^{im\phi}$. Complex arithmetic is not established on GPUs, so that we have to rely on real-valued SHs. This is accomplished by the linear combinations of SHs Y_l^m and Y_l^{-m} to obtain $2\cos(m\phi) = (e^{-im\phi}+e^{im\phi})$ and $2\sin(m\phi) = (e^{-im\phi}-e^{im\phi})$. We adhere to the convention as in Descoteaux et al. [5] and define the real-valued SHs $\tilde{Y}_l^m$ as

$$\tilde{Y}_l^m(\theta,\phi) = \begin{cases} \sqrt{2}\,\mathrm{Re}\big(Y_l^m(\theta,\phi)\big) & \text{if} \quad m<0 \\ Y_l^0(\theta,\phi) & \text{if} \quad m=0. \\ \sqrt{2}\,\mathrm{Im}\big(Y_l^m(\theta,\phi)\big) & \text{if} \quad m>0 \end{cases} \tag{6}$$

$\mathrm{Re}\big(Y_l^m(\theta,\phi)\big)$ and $\mathrm{Im}\big(Y_l^m(\theta,\phi)\big)$ represent the real and imaginary part of the $Y_l^m(\theta,\phi)$. These real-valued SHs $\tilde{Y}_l^m(\theta,\phi)$ are again orthonormal due to the normalization factor $\sqrt{2}$ and also eigenfunctions of the spherical Laplacian Δ_{S_2}. Hence the previously stated properties of SHs remain valid. Only the tilde indicates the use of real-valued SHs.

Employing basis functions of Eq. 6, the computation for preprocessing and rendering of the HARDI data is significantly simplified. Furthermore, note, that HARDI signals are real-valued and symmetric under inversion. It is therefore sufficient to expand a HARDI signal $\psi(\theta,\phi)$ by real-valued coefficients $\tilde{a}_l^m$ and SHs $\tilde{Y}_l^m(\theta,\phi)$ with even $l = 0,2,4,\ldots l_{\max}$.

$$\psi(\theta,\phi) \approx \sum_{l=0,2,\ldots}^{l_{\max}}\sum_{m=-l}^{l} \tilde{a}_l^m\,\tilde{Y}_l^m(\theta,\phi)\;. \tag{7}$$

We truncate this Laplace series at a given $l_{\max}$, since the angular sampling of a HARDI signal is finite as well, and due to the fact that noise becomes more dominant in higher frequencies. On the other hand, the higher the order $l_{\max}$, the higher the angular frequencies that are captured, thus allowing more complicated and better resolved shapes for the reconstructed function $\psi(\theta,\phi)$. Hess et al. [8] proved, that order $l_{\max}=4$ results in good SNR performance and least angular error for the reconstruction of the correct orientation distribution function for HARDI data acquired in a clinical setup with b-values up to $4000 s/mm^2$.

One can determine the spherical harmonic coefficients $\tilde{a}_l^m$ via Eq. 4 or by any of the existing state-of-the-art HARDI techniques [5, 20, 19]. With small modifications the same approach can be applied to techniques like PASMRI [12], since the *persistent angular structure* exhibits functions similar to the orientation distribution function (ODF).

Once we have pre-processed data i.e. volumes of spherical functions calculated by any of the above mentioned HARDI modeling techniques, we can reconstruct the angular PDFs or ODFs using the Eq. 7.

3.2 Bounding Sphere

For scaling and ray-casting purposes, it is necessary to have an upper bound for the maximum radius of the spherical function $\psi(\theta,\phi)$ represented by Laplace series in Eq. 7.

From the Cauchy-Schwarz inequality, we know that

$$\left(\sum_n a_n b_n\right)^2 \leq \left(\sum_n a_n^2\right)\,\left(\sum_n b_n^2\right), \text{ for } a_n, b_n \in \mathbb{R}\;. \tag{8}$$

For $b_n = 1$ this amounts to

$$\left(\sum_{n=0}^{N} a_n\right)^2 \leq (N+1)\,\sum_{n=0}^{N} a_n^2\;. \tag{9}$$

If we apply these inequalities to Eq. 7, we obtain for even l and $l_{\max}$

$$\begin{aligned} \big(\psi(\theta,\phi)\big)^2 \;&=\; \left(\sum_{l=0,2,\ldots}^{l_{\max}}\sum_{m=-l}^{l} \tilde{a}_l^m\,\tilde{Y}_l^m\right)^2 \\ &\leq\; \left(\frac{l_{\max}}{2}+1\right)\sum_{l=0,2,\ldots}^{l_{\max}}\left(\sum_{m=-l}^{l}\tilde{a}_l^m\,\tilde{Y}_l^m\right)^2 \\ &\leq\; \left(\frac{l_{\max}}{2}+1\right)\sum_{l=0,2,\ldots}^{l_{\max}}\left(\sum_{m=-l}^{l}(\tilde{a}_l^m)^2\right)\left(\sum_{m=-l}^{l}\big(\tilde{Y}_l^m\big)^2\right)\;. \end{aligned} \tag{10}$$

With the SH identity [18]

$$\sum_{m=-l}^{l}\big(\tilde{Y}_l^m\big)^2 \;=\; \frac{2l+1}{4\pi} \tag{11}$$

we finally derive

$$|\psi(\theta,\phi)| \;\leq\; \sqrt{\frac{1}{2}l_{\max}+1}\,\sqrt{\sum_{l=0}^{l_{\max}}\left(\frac{2l+1}{4\pi}\sum_{m=-l}^{l}\big(\tilde{a}_l^m\big)^2\right)} \;=\; r_{\max}\;. \tag{12}$$

$r_{\max}$ can be used as the radius of an encompassing sphere that engulfs the glyph given by $\psi(\theta,\phi)$. This way we have a fairly good upper bound for the glyph size in the 3D volume of data voxels at hand.

3.3 Generating Normals

For proper shading it is essential to obtain the normals on the surface of the spherical functions $\psi(\theta,\phi)$. This is not a trivial task as these spherical functions consist of arbitrary, linear combinations of SHs. To resolve the problem we take a small detour and express the spherical function $\psi(\theta,\phi)$ by a volume density $\Phi(r,\theta,\phi)$ in spherical coordinates.

$$\Phi(r,\theta,\phi) \;:=\; r-\psi(\theta,\phi)\;. \tag{13}$$

Thus, the spherical function $\psi(\theta,\phi)$ constitutes the 0-contour of the volume density $\Phi(r,\theta,\phi)$.

$$\Phi(r,\theta,\phi)\Big|_{r=\psi(\theta,\phi)} \;=\; 0\;. \tag{14}$$

This detour is feasible, because the gradient vectors of a density field are normal to its contours. Hence,

$$\nabla\Phi(r,\theta,\phi)\Big|_{r=\psi(\theta,\phi)} \;\perp\; \psi(\theta,\phi)\;. \tag{15}$$

Furthermore, a gradient is a linear differential operator. So it is possible to pass it through the linear combination of SHs.

$$\nabla\Phi(r,\theta,\phi) \;=\; \nabla r \;-\; \sum_{l=0}^{l_{\max}} \sum_{m=-l}^{l} \tilde{a}_l^m \nabla\tilde{Y}_l^m(\theta,\phi)\;. \qquad (16)$$

Obviously, we do not want to calculate the gradient field for all r, but only at the 0-contour with $r = \psi(\theta,\phi)$. We therefore take a look at the explicit expressions for the gradients ∇r and $\nabla\tilde{Y}_l^m(\theta,\phi)$ to examine their r-dependence. Note, that the volume domain of $\Phi(r,\theta,\phi)$ is parameterized by spherical coordinates. The gradient vectors below, however, are given in a Cartesian tangent space with x, y, and z-components.

$$\nabla r \;=\; \begin{pmatrix} \cos(\phi)\sin(\theta) \\ \sin(\theta)\sin(\phi) \\ \cos(\theta) \end{pmatrix}\;. \qquad (17)$$

Apparently, ∇r is r-independent so that a substitution $r = \psi(\theta,\phi)$ is not an issue. The gradient of a SH is given by

$$\nabla\tilde{Y}_l^m(\theta,\phi) \;=\; \frac{1}{r}\begin{pmatrix} \cos(\theta)\cos(\phi)\,\partial_\theta - \csc(\theta)\sin(\phi)\,\partial_\phi \\ \cos(\theta)\csc(\theta)\,\partial_\phi + \cos(\theta)\sin(\phi)\,\partial_\theta \\ -\sin(\theta)\,\partial_\theta \end{pmatrix}\tilde{Y}_l^m(\theta,\phi)\;. \qquad (18)$$

Again, the SHs $\tilde{Y}_l^m(\theta,\phi)$ and also the vectorial differential operator in Eq. 18 are r-independent. Only a factor $1/r$ remains, which we can move outside the linear combination in Eq. 16. Hence, we can write

$$\nabla\Phi(r,\theta,\phi) \;=\; \nabla r \;-\; \frac{1}{r}\,s\sum_{l=0}^{l_{\max}} \sum_{m=-l}^{l} \tilde{a}_l^m\, r\nabla\tilde{Y}_l^m(\theta,\phi) \qquad (19)$$

and obtain

$$\nabla\Phi(r,\theta,\phi)\Big|_{r=\psi(\theta,\phi)} \;=\; \begin{pmatrix} \cos(\phi)\sin(\theta) \\ \sin(\theta)\sin(\phi) \\ \cos(\theta) \end{pmatrix} \;- \qquad (20)$$

$$\frac{1}{\psi(\theta,\phi)}\sum_{l=0}^{l_{\max}} \sum_{m=-l}^{l} \tilde{a}_l^m \begin{pmatrix} \cos(\theta)\cos(\phi)\,\partial_\theta - \csc(\theta)\sin(\phi)\,\partial_\phi \\ \cos(\phi)\csc(\theta)\,\partial_\phi + \cos(\theta)\sin(\phi)\,\partial_\theta \\ -\sin(\theta)\,\partial_\theta \end{pmatrix}\tilde{Y}_l^m(\theta,\phi)\;.$$

After normalization, the gradients above are unit normals of the glyphs defined by the spherical function $\psi(\theta,\phi)$.

4 GPU-Based Ray Casting of Spherical Harmonics

Our goal is to show a set of glyphs to represent the spherical functions $\psi(\theta,\phi)$ in a cross-section of the data. This cross-section is defined by a plane-of-interest (POI) which the user can translate, rotate and resize using the mouse. The seed points that define the locations of the glyphs are implicitly given by the POI and a user-specified seed distance. When we render the scene, we step through all the seed points in order to deal with each glyph separately. Per glyph, the calculations are divided between CPU and GPU. On the CPU, we do the computations 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

On the CPU, we load the pre-computed spherical harmonics coefficients. Before rendering, we manipulate the coefficients 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, determines a value by which $\tilde{a}_0^0$ is divided to reduce the influence of isotropic component $\tilde{Y}_0^0$ and, thus, to enhance the anisotropic maxima of the glyphs. Finally, the user has the option to scale all glyphs independently to have the same $\tilde{a}_0^0$ for each glyph, or to scale all glyphs by the same global maximum of $\tilde{a}_0^0$ so that the differences in diffusion magnitude between different voxels becomes apparent.

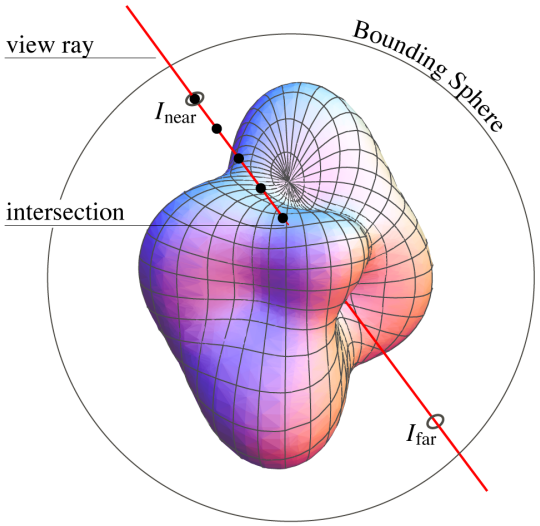


Figure 2: Illustration of our ray-casting algorithm.

After manipulating the coefficients we need to render a geometry that covers the pixels in the viewport potentially occupied by the projection of the glyph. For this, we compute the maximum radius $r_{\max}$ of the spherical function as given in Eq. 12 and render an bounding cube that is aligned along the world axis and positioned with the glyph's seed point as its center. The edges of the bounding cube have length 2 $r_{\max}$. On the GPU we need $r_{\max}$ to compute bounds for the ray casting, so we pass its value as a vertex attribute. We also pass the 15 coefficients $\tilde{a}_l^m$ as vertex attributes, since we truncate the Laplace series at order 4 for the reasons given in section 3.1.

4.2 GPU Algorithm Overview

In the vertex shader we only pass the coefficients $\tilde{a}_l^m$ to the fragment shader and compute the view direction $\vec{v}$ by substracting the eye position V from the current vertex positions of the bounding box. Vector $\vec{v}$ is automatically interpolated for use in the fragment shader, where we normalize it.

Figure 2 illustrates the ray-casting process that is executed in the fragment shader. The three basic steps of our method are:

1. Compute the intersections of view-ray $\mathscr{V}$ with bounding sphere $\mathscr{S}$ which has radius $r_{\max}$. If there are no intersections, discard the fragment. If there are intersections, we use them as bounds in the next step.

2. Approximate the intersection point I of $\mathscr{V}$ and the surface $\mathscr{H}$ defined by spherical function $\psi(\theta,\phi)$.

3. Compute the normal of $\mathscr{H}$ in I and do 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.

4.3 Intersection of View-Ray and Sphere

Before computing the intersection of the view ray $\mathscr{V}$ with surface $\mathscr{H}$ defined by $\psi(\theta,\phi)$, we compute the intersections of $\mathscr{V}$ with bounding sphere $\mathscr{S}$. There are two reasons for this:

1. If there is no intersection of $\mathscr{V}$ with $\mathscr{S}$, which can be determined very fast, the fragment can be discarded and we save the computations needed to determine intersection of $\mathscr{V}$ with $\mathscr{H}$.

2. We use the intersections of $\mathscr{V}$ with $\mathscr{S}$ as bounds in the algorithm that determines the intersection of $\mathscr{V}$ with $\mathscr{H}$.

The intersections I_{near}, I_{far} of $\mathscr{V}$ with $\mathscr{S}$ can be determined analytically by substituting the equation of the ray

$$\mathscr{V}(t) = V + t\vec{v} \tag{21}$$

into the equation of a sphere

$$(x-P_x)^2 + (y-P_y)^2 + (z-P_z)^2 = r_{\max}^2 \tag{22}$$

where $P = (P_x, P_y, P_z)$ is the center of the sphere and $r_{\max}$ the radius. The resulting equation can be solved using the quadratic formula.

$$\mu = \frac{-b \pm \sqrt{D}}{2a} \text{ , where } D = b^2 - 4ac \text{ ,} \tag{23}$$

with

$$a = \sum_{i \in \{x,y,z\}} \vec{v}_i^2 \text{ ,} \tag{24}$$

$$b = \sum_{i \in \{x,y,z\}} 2(V_i - P_i)\vec{v}_i \text{ ,} \tag{25}$$

$$c = \sum_{i \in \{x,y,z\}} (V_i - P_i)^2 - r^2 \text{ .} \tag{26}$$

The number of solutions depends on the value of D. For $D < 0$ there is no intersection and we discard the fragment. For $D > 0$, we have $I_{near} = \mathscr{V}((b^2 - \sqrt{D})/2a)$ and $I_{far} = \mathscr{V}((b^2 + \sqrt{D})/2a)$. For $D = 0$, we obtain $I_{near} = I_{far}$.

4.4 Intersection of view-ray and spherical function

Figure 2 shows a view ray $\mathscr{V}$ and how it can intersect a spherical surface $\mathscr{S}$. In order to find the intersection of $\mathscr{V}$ and $\mathscr{H}$, we start with a basic ray casting approach that does a linear search with discrete steps along $\mathscr{V}$ to find the first point on $\mathscr{V}$ that is inside $\mathscr{H}$. For this, we initialize the point p on $\mathscr{V}$ with I_{near} and vector *step* to have the same orientation as $\vec{v}$ and user-definable length h.

```
n = 0;
while ((!IsInsideSH(p)) && (n < MaxN))
  {
  p = p + step;
  n++;
  }
```

MaxN is computed from the distance between I_{near} and I_{far} and step-length h. If we reach $n = MaxN$ before a point inside $\mathscr{H}$ is found, then there is no intersection and we discard the fragment. The boolean function $IsInsideSH(p)$ is described below. The user can interactively change the value of h to ensure fast rendering and to make sure that h is not too large for important details to be missed. After the linear search, we initialize points *upper*, *lower* with p and p - *step*, which are on opposite sides of $\mathscr{H}$, and use a binary search algorithm to refine the position of the intersection:

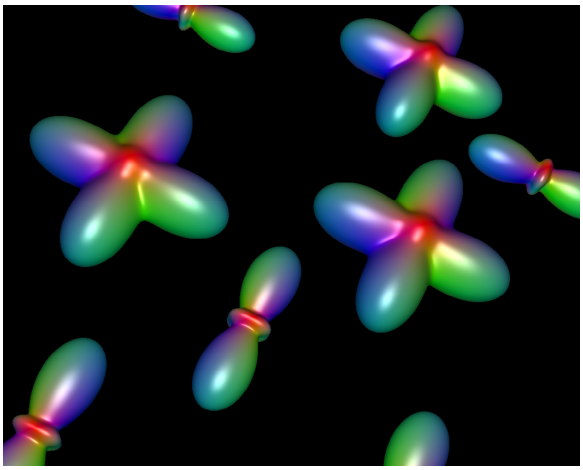


Figure 3: Close-up of our GPU glyphs showing a detail of a synthetic phantom dataset of two crossing fiber bundles under an angle of 90°

```
for (int i=0; i < NumRefineSteps; i++)
  {
  p = (lower + upper) / vec3(2.0);
  if (IsInsideSH(p))
    upper = p;
  else
    lower = p;
  }
p = (lower + upper) / vec3(2.0);
```

In order to determine whether a point p is inside or outside $\mathscr{H}$ in $IsInsideSH(p)$, we first compute p in local spherical coordinates for the current spherical function:

$$\begin{aligned} \vec{d} &= \text{normalize}(p - SHcenter) \\ \theta &= \text{atan}(\sqrt{d_x^2 + d_y^2}/d_z) \\ \phi &= \text{atan}(d_y/d_x) \\ r &= \text{length}(p - SHcenter) \end{aligned}$$

Now, if we compute $\psi(\theta,\phi)$ cf. Eq. 7 we get the radius of $\mathscr{H}$ in that direction. If $r > \psi(\theta,\phi)$ then p is outside $\mathscr{H}$ and if $r < \psi(\theta,\phi)$ then it is inside. For computing $\psi(\theta,\phi)$ we computed the analytical expressions for Y_l^m cf. Eq. 5 for l up to 4.

4.5 Normal Computation and Lighting

We compute the normals in the intersection points on the ray-casted surface as described in section 3.3. In order to do this, we use Eq. 20 and the analytical expressions of the spherical harmonics. Using the normal, we use standard lighting according to the Phong lighting model.

Furthermore, we use the commonly-used color coding in HARDI and DTI that maps the orientation vector that points from the center of the glyph to the surface, to a color, where red corresponds to mediolateral, green to anteroposterior, and blue to superoinferior orientation. The results are illustrated in Fig. 3.

5 Results

In Fig. 4 we show a region-of-interest on a coronal slice of centrum semiovale since fibers of the corpus callosum (CC), corona radiata (CR), and superior longitudinal fasciculus (SLF) form a three-fold crossing there. This is a very interesting region, with known crossings and therefore it is very often a target for qualitative analysis of DW- MRI techniques. We generate artificial noiseless synthetic data of a 90° crossing according to Soderman et al. [25] (see Fig.

#glyphs	step size h	#refine steps	rendering performance (FPS)
70	0.01	4	>100
732	0.01	4	70–90
4125	0.01	0	10–23
4125	0.01	4	8–23
4125	0.02	0	∼20
4125	0.02	4	∼20

Table 2: Rendering performance for our proposed GPU-based glyph-rendering method with varying parameters.

3). We use this data to validate our implementation. 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 ODF. The areas of crossings are correctly represented as well. In Fig. 3 we immediately observe the advantage of our GPU based rendering: the quality of the rendered glyphs is independent of the zooming factor.

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 1000, 1500, 2000, 3000 s/mm^2.

We visualize the data with our proposed technique and compare it with the previous methods that generate geometry via polygons. First, we analyze the visual results in 5.1 and then we give performance measurements in 5.2.

5.1 Visual Aspects

In Fig. 4(a) we show ODF glyphs rendered with the geometry-based method with 3^{rd} order of tessellation of an icosahedron. A similar ROI is shown in Fig. 4(b). The increase of visual quality can be immediately observed, crossings become visually more clear, and there is an obvious 3D effect that perceptually helps in the distinction of the CC and CR structures

5.2 Performance

We compared the performance of the proposed method to the performance of the traditional method where geometry is generated. The results of the traditional and the proposed method are shown in table 1 and table 2 respectively. Rendering was done on a PC with an Intel Pentium 4 3.20 GHz CPU, 3 GB RAM and a GeForce 8800 GTX graphics card with 768 MB of memory.

For the generation of geometry we used a custom VTK filter of which the output was rendered with a standard VTK polydata mapper. Although we did not optimize the methods that were used for generating and rendering geometry, the results clearly indicate that our proposed method outperforms the geometry-based method. Of course, in both 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 the orders of tessellation that give good enough visual results cannot be rendered in an interactive way. Furthermore, with the geometry-based method it is impossible to interactively define the region-of-interest (ROI), because it needs to generate geometry for the glyphs in the new ROI. Also, when changing parameters such as scaling and sharpening, the geometry needs to be changed and thus this also cannot be done interactively.

Our proposed method gives satisfactory visual results even with the lowest settings listed in table 2 (step size 0.02 and no refine steps). With these settings, the glyphs look smoother than with the traditional method using 4^{th} order of tessellation. Furthermore, we can define the current ROI interactively, since no geometry needs to be generated, and we can also change all rendering parameters interactively. For the figures 3 and 4b in this paper, we used the settings from the tables that give the best visual results, i.e. a step size $h = 0.01$ and 4 refinement steps.

6 Conclusions and Future Work

We presented a fast and flexible method for GPU rendering of HARDI glyphs. It is important to state that HARDI is an emerging area, exploding with new proposed techniques for data modeling, regularization, fiber tracking and more. Validation is still an open issue in HARDI. Showing the local information of the reconstructed PDF, ODF or any other function that resides on a sphere, in a fast way is essential for the development of the 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 methods from pure research into clinically applicable new and better DW-MRI techniques for depicting the white matter structure of the brain.

We stress that this visualization is aimed at researchers in the area of HARDI and not physicians, due to the complexity and cluttering of the data. This GPU-accelerated visualization will improve the working conditions of the researchers, allowing them smooth and interactive data exploration. The rendering technique is based on a spherical harmonic approximation of any function on a sphere. Therefore it is not method-specific. Any of the existing HARDI techniques that can be expressed by a SH basis or a high order tensor basis (there is a direct relationship between these two representations), can be used as a pre-processing step to the work with our algorithm.

It will be useful to extend the method with specialized coloring and sharpening methods that will highlight maxima of a spherical function and possibly highlight inter-voxel coherence. We will also integrate the new method into our visualization tools for DTI so that we can visualize different imaging modalities together. In order to further increase the performance of the ray-casting method, we are investigating numerical methods such as Newton-Raphson for the computation of the intersection of the view ray with the surface represented by a spherical function. Finally, we will investigate computer graphics techniques such as deferred shading and make use of geometry shaders and vertex buffers to further improve the performance.

Acknowledgements

References

[1] 3D slicer. http://www.slicer.org.

[2] P. J. Basser, J. Mattiello, and D. Lebihan. MR diffusion tensor spectroscopy and imaging. *Biophys. J.*, 66(1):259–267, January 1994.

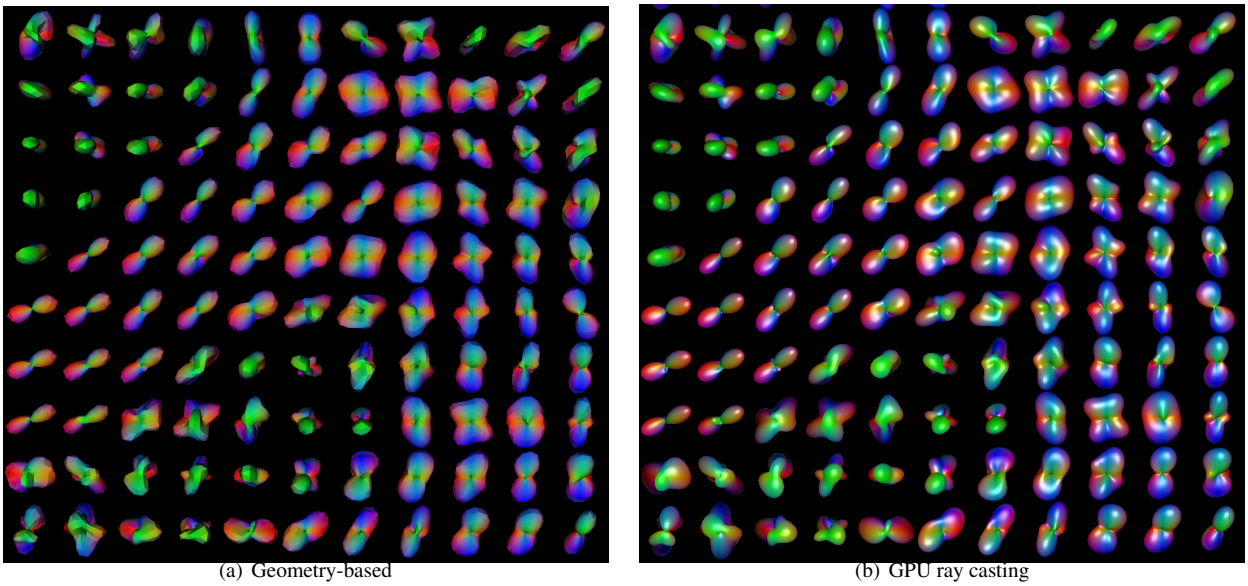
[3] P. Cook, Y. Bai, S. Nedjati-Gilani, K. Seunarine, M. Hall, G. Parker, and D. Alexander. Camino: Open-source diffusion-MRI reconstruction and processing. In *14th Scientific Meeting of the International Society for Magnetic Resonance in Medicine*, 2006.

[4] R. Deriche and M. Descoteaux. Splitting tracking through crossing fibers: Multidirectional q-ball tracking. In *ISBI*, pages 756–759. IEEE, 2007.

[5] M. Descoteaux, E. Angelino, S. Fitzgibbons, and R. Deriche. Regularized, fast and robust analytical q-ball imaging. *Magn. Reson. Med.*, 58:497–510, 2007.

[6] L. R. Frank. Anisotropy in high angular resolution diffusion-weighted MRI. *Magn Reson Med*, 45(6):935–9, 2001.

[7] S. Gumhold. Splatting illuminated ellipsoids with depth correction. In *VMV*, pages 245–252. Aka GmbH, 2003.



(a) Geometry-based (b) GPU ray casting

Figure 4: ODF representations of the DW-MRI data with 106 gradient directions and b=2000 s/mm^2 of a human subject in a ROI defined on a coronal slice in the centrum semiovale. The glyphs are shown for 4^{th} order spherical harmonics, which are min-max normalized (a): Traditional way of HARDI glyphs rendering with 3^{rd} order icosahedral tessellation so interaction is still possible. (b): Ray casting of the HARDI glyphs on GPU.

#glyphs	tessellation order	#points per glyph	generation time (s)	rendering performance (FPS)
70	3	168	<1	110
70	4	252	1.1	37
70	7	642	61	<1
700	3	168	3	20
700	4	252	22	5
700	5	362	40	<1
4125	4	252	55	<1

Table 1: Rendering performance of the classical geometry-based glyph-rendering method with varying order of tessellation.

[8] C. Hess, P. Mukherjee, E. Han, D. Xu, and D. Vigneron. Q-ball reconstruction of multimodal fiber orientations using the spherical harmonic basis. *Magn Reson Med*, 2006.

[9] M. Hlawitschka, S. Eichelbaum, and G. Scheuermann. Fast and memory efficient gpu-based rendering of tensor data. In *IADIS Computer Graphics and Visualization*, 2008.

[10] M. Hlawitschka, G. Scheuermann, A. Anwander, T. Knösche, M. Tittgemeyer, and B. Hamann. Tensor lines in tensor fields of arbitrary order. In *ISVC (1)*, volume 4841 of *Lecture Notes in Computer Science*, pages 341–350. Springer, 2007.

[11] M. Hlawitschka, G. Scheuermann, and B. Hamann. Interactive glyph placement for tensor fields. In *3rd International Symposium on Visual Computing*, 2007.

[12] K. Jansons and D. Alexander. Persistent angular structure: new insights from diffusion MRI data. dummy version. *Inf. Process. Med. Imaging*, 18, 2003.

[13] S. Jbabdi, M. W. Woolrich, J. L. R. Andersson, and T. E. J. Behrens. A bayesian framework for global tractography. *NeuroImage*, 37(1):116–129, August 2007.

[14] G. Kindlmann. Superquadric tensor glyphs. In *Joint Eurographics - IEEE TCVG Symposium on Visualization*, pages 147–154, 2004.

[15] G. Kindlmann. *Visualization and Analysis of Diffusion Tensor Fields*. PhD thesis, School of Computing, University of Utah, September 2004.

[16] G. Kindlmann and C.-F. Westin. Diffusion tensor visualization with glyph packing. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization / Information Visualization 2006)*, 12(5):1329–1335, September-October 2006.

[17] O. Marichev and M. Trott. http://functions.wolfram.com/05.10.02.0001.01.

[18] O. Marichev and M. Trott. http://functions.wolfram.com/05.10.23.0002.01.

[19] E. Özarslan and T. H. Mareci. Generalized diffusion tensor imaging and analytical relationships between diffusion tensor imaging and high angular resolution diffusion imaging. *Magn Reson Med*, 50(5):955–65, 2003.

[20] 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*, 36(3):1086–1103, July 2006.

[21] M. Perrin, Y. Cointepas, C. Poupon, 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. In *IPMI, Glenwood Springs, Colorado*, pages 52–63, 2005.

[22] A. D. Polyanin. *Handbook of Linear Partial Differential Equations for Engineers and Scientists*. Chapman and Hall/CRC Press, 2002.

[23] 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. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2008*, volume 5242 of *Lecture Notes in Computer Science*, pages 298–305. Springer, 2008.

[24] C. Sigg, T. Weyrich, M. Botsch, and M. Gross. GPU-based ray-casting of quadratic surfaces. In *Eurographics Symposium on Point-Based Graphics*, 2006.

[25] O. Söderman and B. Jönsson. Restricted diffusion in cylindirical geometry. *J. Magn. Reson. B*, 117(1):94–97, 1995.

[26] D. Tuch. Q-ball imaging. *Magn Reson Med*, 52:1358–1372, 2004.

[27] D. S. Tuch. *Diffusion MRI of complex tissue structure*. PhD thesis, Harvard, 2002.

[28] D. Wassermann, M. Descoteaux, R. Deriche, and C. Westin. Qball plug-in for slicer3d. http://www-sop.inria.fr/odyssee/en/software/qballslicer.