Instant visitation maps for interactive visualization of uncertain particle trajectories

Kai Bürger[a], Roland Fraedrich[a], Dorit Merhof[b] and Rüdiger Westermann[a]

[a]Computer Graphics & Visualization group, Technische Universität München, Garching, Germany;
[b]Visual Computing group, University of Konstanz, Konstanz, Germany

ABSTRACT

Visitation maps are an effective means to analyze the frequency of similar occurrences in large sets of uncertain particle trajectories. A visitation map counts for every cell the number of trajectories passing through this cell, and it can then be used to visualize pathways of a certain visitation percentage. In this paper, we introduce an interactive method for the construction and visualization of high-resolution 3D visitation maps for large numbers of trajectories. To achieve this we employ functionality on recent GPUs to efficiently voxelize particle trajectories into a 3D texture map. In this map we visualize envelopes enclosing particle pathways that are followed by a certain percentage of particles using direct volume rendering techniques. By combining visitation map construction with GPU-based Monte-Carlo particle tracing we can even demonstrate the instant construction of a visitation map from a given vector field. To facilitate the visualization of safety regions around possible trajectories, we further generate Euclidean distance transform volumes to these trajectories on the fly. We demonstrate the application of our approach for visualizing the variation of stream lines in 3D flows due to different numerical integration schemes or errors introduced through data transformation operations, as well as for visualizing envelopes of probabilistic fiber bundles in DTI tractography.

Keywords: flow envelopes, probabilistic trajectories, comparative visualization, GPUs and multi-core architectures

1. INTRODUCTION

Particle tracing in 3D flow fields is a well-established method for visualizing the complicated directional structures that can appear in such fields. However, even for very small perturbations of the local particle velocities the occurrence of particle trajectories in 3D flow fields can be significantly different. For instance, such perturbations can be caused by uncertainties in the vector field data itself, or they can result from different interpolation or integration schemes during particle traversal. For a thorough overview of the different sources of such perturbations let us refer to Ref. 1, Ref. 2, and Ref. 3. As a consequence, one has to be aware that in many cases a single trajectory only shows one possible realization of a particle path in the data, and that analyzing the data while ignoring the possible trajectory variations can severely mislead the viewer.

In this work we are addressing the problem to efficiently visualize variations of characteristic trajectories in 3D flow fields. To achieve this, we are going to present an interactive method for the construction and visualization of high-resolution 3D visitation maps for large numbers of trajectories. A visitation map counts for every cell the number of trajectories passing through this cell, and it can then be used to visualize pathways of a certain visitation percentage. Thus, visitation maps are an effective means to analyze the frequency of similar occurrences in large sets of uncertain particle trajectories.

Our method takes as input a set of trajectories or, as it will be assumed throughout this paper, a 3D flow field that can be further accompanied by any local or global perturbation model describing the possible directional variations in this field. By intertwining GPU-based Monte-Carlo particle tracing in this flow field with the visitation map construction we can even demonstrate the instant construction of such a map from a given flow field.

It is important to note that in this work we will not attempt to derive any perturbation model for describing the source of trajectory variations. Instead, we will assume at every point in the 3D domain a local uncorrelated probability distribution that gives a random variation of the flow vector at this point. When using our method for uncertainty visualization in flow

Further author information: (Send correspondence to K. Bürger)
E-mail: {buerger, fraedrich, westermann}@tum.de, Dorit.Merhof@uni-konstanz.de

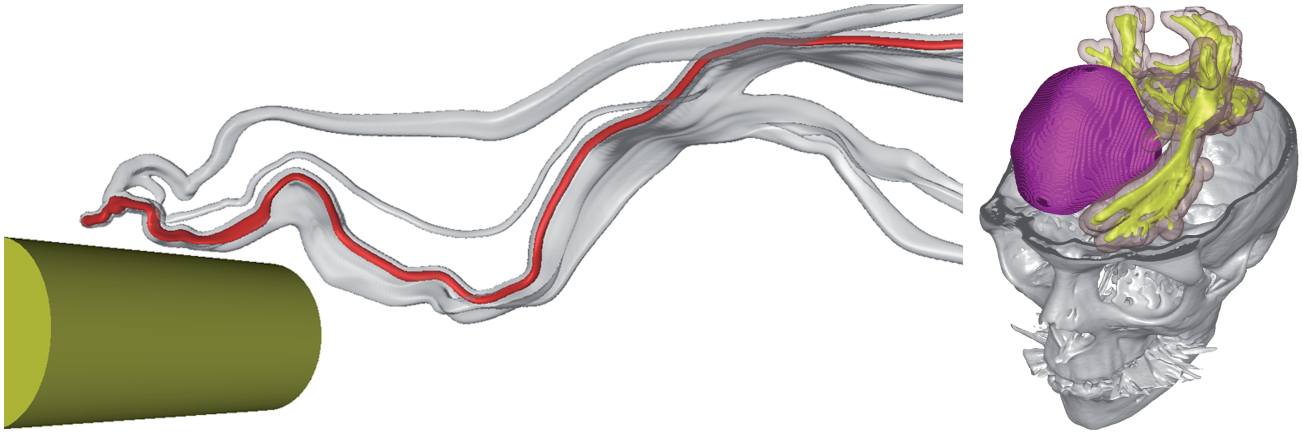

Figure 1. Left: The trajectories of two thousand particles starting at the same position in a 3D flow field are visualized. Along every trace the local particle velocities were slightly perturbed by a random deviation. Particle tracing, visitation map construction, and visualization takes 175 ms using our approach. Right: By computing distance volumes for a given visitation map on the fly, safety margins to possible pathways can be visualized. For a visitation map of size $256^2 \times 144$, the safety margin to a pathway of a certain visitation percentage can be computed in less than 130 milliseconds.

fields, this model has to be replaced by an application specific model describing the occurring directional variations. Given such a model, our approach can then be used to visualize the resulting trajectory variations at interactive rates.

Our method takes advantage of the ever increasing degree of parallelism and memory bandwidth on recent graphics programming units (GPUs), and it uses these capacities for tracing massive amounts of particles through a 3D flow field. In contrast to standard geometry-based flow visualization, however, characteristic trajectories are not rendered to the screen, but they are rasterized in turn into a 3D grid structure to construct a 3D visitation map. During trajectory extraction, this map continually collects the footprints of particles traversing along their paths, building a scalar field that indicates the frequency of particle occurrence at every map entry.

Building the visitation map is intertwined with a GPU rendering thread that shows isosurfaces in the map using volume ray-casting. We will subsequently call these surfaces “visitation envelopes”. As the map is constructed in such a way that isosurfaces in the accumulated scalar field enclose regions of a certain particle visitation percentage, i.e., pathways that have been followed by a certain number of particles, our method allows visualizing more or less likely pathways at very high speed. For instance, the visitation envelopes in Fig. 1 (left) were constructed and visualized in less than 180 ms using two thousand particles, each of which was followed for 1024 integration steps. To facilitate interactivity even for very large particle sets, we interleave the visitation map construction and the visualization of visitation envelopes. Instead of constructing the visitation map for all particles at once, the particle set is partitioned into sub-sets for which construction and visualization is performed successively. In this way, the map is build progressively at interactive rates, enabling the user to view the emerging map from any desired view point.

To further facilitate the visualization of safety regions around possible pathways, for instance to show which regions in the domain are very unlikely to be visited by any particle, we further employ the GPU for the efficient construction of 3D distance transforms to a given feature volume. Here the user can interactively select a specific isosurface in the visitation map to which a distance transform is computed. Hence, isosurfaces that are precisely a certain distance away from the possible pathways can be selected interactively and used to determine the minimal distance between two objects. Semi-transparent isosurfaces can then be rendered using scale-invariant volume ray-casting.[4] The right image in Fig. 1 shows such a safety margin that was visualized in a $256^2 \times 144$ visitation map in less than 190 ms on the GPU.

The specific contributions our paper makes are

- a novel interactive construction method for visitation volumes on the GPU,
- a new progressive approach for the interactive visualization of more or less likely pathways by interleaving visitation map construction and direct volume rendering,

- a new combination of GPU visitation maps and distance transforms to interactively visualize safety margins around more or less likely particle pathways.

The remainder of this paper is organized as follows: After reviewing previous work that is related to ours, Section 3 introduces our GPU-based technique to construct visitation maps in real time. Section 4 describes how safety margins around visitation envelopes within the visitation map can be computed interactively. In Section 5, we present a variety of visualization modalities and discuss exemplary experiments in which our framework was used to explore local uncertainty in 3D vector and tensor fields. In Section 6, we evaluate the performance of our technique, and we conclude the paper with an outline of future research in the field in Section 7.

2. RELATED WORK

Our approach is based on a number of established techniques in visualization, namely particle tracing, visitation map construction, and distance transform computation. A thorough overview of the respective research areas is beyond the scope of this paper; however, for some useful surveys on these areas let us refer to Ref. 5, Ref. 6, and Ref. 7, respectively.

Particle tracing: Interactive flow exploration heavily relies on interactive frame rates and is therefore often concerned with efficient solutions on graphics hardware. On recent GPUs it is now possible to interactively trace millions of particles in Cartesian grids using higher order integration schemes[8–11], and to instantly render these particles using a multitude of different options including oriented point sprites, lines, and more complex geometric representations like bands and tubes.[12–15] Techniques for visualizing uncertainties in vector fields as well as in the trajectories of particles in such fields have been proposed in.[1,16–19]

Visitation maps: In the field of Diffusion Tensor Magnetic Resonance Imaging (DT-MRI), visitation maps are often employed in combination with probabilistic tractography approaches to estimate the locations of fiber bundles and to analyze the white matter connectivity patterns in the human brain.[6,7,20–22]

3D distance transform: Maurer et al. proposed a very efficient algorithm for computing a 3D distance transform in Ref. 23 which, however, relies on frequent concurrent read/write accesses—an operation that significantly limits the performance of recent GPUs. The use of geometric primitives for hardware-accelerated distance field computation was proposed in Ref. 24–26. A slice-based GPU framework for computing 3D distance transforms was proposed in Ref. 27. Rong and Tan[28] proposed the jump flooding paradigm, which presents a communication pattern to quickly propagate information in highly SIMD parallel computing environments such as GPUs. Recently, Schneider et al.[29] introduced a discrete distance transform using vector propagation on the GPU. For Cartesian volumes as large as 256^3, the 3D distance transform can be computed in less than 200 milliseconds on recent GPUs.

3. VISITATION VOLUME CONSTRUCTION

Visitation maps (in combination with probabilistic tractography approaches) are extensively used in Diffusion Tensor Magnetic Resonance Imaging (DT-MRI) to construct a percentage visitation index for each voxel in a 3D Cartesian voxel grid.[6,7,20–22] The general idea is to perform Monte Carlo (MC) particle tracing, i.e., to start a particle multiple times from the same seed position, each time computing it's trajectory in an uncertain vector field. In the following we assume that the uncertainty—more precisely, a local error—can be modeled by an uncorrelated probability distribution function, which gives at every point in the domain a random variation of the vector at this point. Thus, even though all particles start at the same point, the random variation will cause the trajectories to vary.

Before we proceed, let us stress that we do not intend to introduce any new uncertainty model for a particular application in this work. Rather than that, we are aiming to present a general approach for the interactive construction of visitation maps, regardless of the underlying error model. Therefore, in the following we will assume the existence of a random-vector generator that can be parameterized to any meaningful probability distribution. This generator takes as input a 3D vector and a tuple of random values, and it outputs the randomly deflected vector. In Section 5.1 we will present some experiments demonstrating specific generators in the context of 3D flow visualization and DTI fiber tractography.

After trajectory extraction, a 3D visitation percentage map is constructed by counting for each voxel the number of trajectories that have passed through it. These indices are then employed to visualize visitation envelopes, e.g., to accentuate those voxels that have been visited by a certain percentage of particles.

To provide interactivity, our approach constructs the visitation map iteratively by interleaving Monte Carlo particle tracing, visitation map construction, and the visualization of visitation envelopes. In each iteration only as many particle pathways are processed as can be interactively traced and accumulated into the visitation map. We harness the massive processing power of modern GPUs to perform Monte Carlo particle tracing for a large number of trajectories in parallel, and we exploit their rasterization capabilities to interactively update the visitation map. By doing so, the user can be given immediate visual feedback that becomes more and more reliable the more trajectories are computed. Moreover, this gives rise to the possibility to position the seed point in the domain or change system specific parameters (both of which require to restart visitation map construction) at interactive rates and instant visual feedback.

3.1 GPU-based Monte Carlo Particle Tracing

Monte Carlo particle tracing is performed in i iterations, each constructing n (perturbed) particle trajectories of length len in parallel. This gives a total of $i \times n$ particle traces. Instead of computing all trajectories at once, after each iteration the current content of the visitation map is visualized. The reason therefore is that n can be made small enough so that each iteration can be carried out at very high speed, enabling the user to continually interact with the emerging map or to interactively change the selected seed position. With more and more iterations, the reliability of the visualized visitation envelopes is increased.

To construct the particle trajectories on the GPU, we build on the "ping-pong" stream line advection technique introduced in Ref. 11. Generally speaking, this technique employs two texture resources, each large enough to store the positions for n particles, and one additional buffer of size $n \times len$—the trajectory buffer—to hold the set of particle traces.

Particle integration is performed in len passes, in each of which n traces are progressed in parallel. Each pass receives the results from the last pass as input, advects the particles to their new positions, and subsequently employs an API specific copy operation to transfer the results as a contiguous data block from the output position buffer to the trajectory buffer. Since GPUs do not provide a "global" random number generator, we have to employ a specific strategy to realize the random-vector generator that is required to calculate a randomly disturbed direction vector. First of all we use an additional buffer resource holding $i \times n \times r$ random number seeds, each assuring to generate a unique sequence of len random valued r-tuples along all particle traces. Furthermore, an additional set of ping-pong buffers is created, each large enough to hold a sequence of $n \times r$ unique random seed values required during particle advection.

At the beginning of every Monte Carlo iteration, the corresponding block of $n \times r$ random values is copied to the advection random input buffer, and every entry in the input position buffer is set to the single seed position. During particle advection, the vector at the current particle position and r random seed values are fed into the generator, which returns a perturbed traversal direction as well as updated random seed values. Particles are then advected to their new positions, and together with the updated random seed values they are written to the ping-pong output buffers. In Section 5.1 we present some specific parameterizations of the random number generator to show the effect of different error sources on the trajectories.

Our technique always extracts n complete trajectories in each Monte Carlo iteration before transferring them into the visitation map to avoid frequent kernel switches between particle tracing and the subsequent visitation map update. By doing so, unnecessary stalls in the graphics pipeline can be omitted and a large number of trajectories can be extracted in each frame.

3.2 Trajectory Voxelization

After the execution of each Monte Carlo iteration the extracted particle trajectories are voxelized into the visitation map, which means to find for a particular trajectory all voxels in the visitation map this trajectory is passing through. Similar to the technique proposed in Ref. 30, we employ the graphics API to render directly into a 3D texture in GPU memory, and we use geometry shader functionality to route primitives to their target positions in this map. In the following we describe the voxelization of trajectories given by a sequence of 3D points in the local object space of the visitation map, which requires to find all voxels covered by the line segments between two adjacent points.

3.2.1 Line voxelization

Let us note first, that we cannot utilize GPU line rasterization for the voxelization of trajectories into slices of the target volume, as line rasterization does not guarantee to generate a fragment for every intersected texel. The reason is that the GPU rasterizer uses the diamond exit rule to determine whether a fragment should be generated for a texel covered by a line segment (see Fig. 2), so that some texels will not be determined even though the trajectory is passing through.

GPU voxelization of the line segments that are given by adjacent points in the trajectory buffer is performed by sending *len* vertices *n* times into the rendering pipeline. We employ an instanced draw call and let the input assembly generate all necessary values (namely a *vertex id* and an *instance id*) to access the particle positions in the trajectory buffer. A vertex shader then uses the *instance id* $\in [0,\ldots,n-1]$ to access one particle trace in the trajectory buffer and the *vertex id* $\in [0,\ldots,len-1]$ to fetch the particle positions along a trace. By initializing the draw call with a line strip topology, the rendering pipeline then issues a line segment for each pair of adjacent particle positions to the geometry shader stage.

Voxelization of the line segment spanned by the two particles is performed within the geometry shader by generating a point primitive for each intersected voxel. To compute these voxels we employ the discrete grid traversal algorithm proposed in[31]. More precisely, for each line segment, the geometry shader starts a grid traversal in the visitation volume and issues a point primitive to the rasterizer for each voxel the ray intersects, except for the voxel that contains the line segment's end position. By doing so, we do not only avoid accumulating duplicate entries within a voxel where two adjacent line segments intersect, but also exclude segments from voxelization if they do not intersect a voxel boundary. Finally, a point primitive is sent into the respective voxel of the visitation map, and additive blending is used to increment the visitation count at this voxel by one.

3.2.2 Spherical footprints

Voxelizing the particle traces as a set of voxel-sized points results in typical staircase patterns in the visitation map. This in turn can lead to highly fragmented isosurfaces in the subsequent visualization, especially if regions of low visitation percentage are to be visualized.

To alleviate this problem, we propose to not only update the voxels a trajectory is passing through, but also to slightly increase the visitation count in their surrounding neighborhood. This can be achieved by splatting spherical footprints along a trajectories, where one splat per control point or per overran voxel is issued. Here, the line voxelization technique as presented above can be used in conjunction with the stream output stage to find and store all intersected voxels in an intermediate vertex buffer before voxelizing spherical footprints into the visitation map.

For the voxelization of the spherical footprints we employ the GPU particle slicing technique presented in Ref. 32. Here, a geometry shader kernel calculates the first and last slice of the visitation map covered by a footprint according to a global support radius of *h* voxels. For each covered slice, the shader then generates one equilateral triangle oriented orthogonal to the z-axis of the visitation map and centered about the coordinates of a control point.

Before sending a triangle to the rasterizer, the geometry shader scales the triangle so that it just covers the cross-section between the sphere and the respective target volume slice. After rescaling, the shader determines for each triangle vertex the distance vector to the particle center and issues the primitive to the rasterization stage. A pixel shader kernel then tests for each generated fragment whether the length of the interpolated distance vector is outside the spherical kernel support and discards the fragment in that case. For all fragments surviving the test a density value is computed and added to the target voxel value via additive blending.

To achieve a smooth transition of density values between overlapping spherical footprints, we determine the density value at a given position according to the *poly*6-kernel (with constant support radius *h*), which is generally applied in the field of particle based fluid dynamics. This kernel is given as follows:

$$W_{poly6}(d,h) = \frac{315}{64\pi h^9} \begin{cases} (h^2-d^2)^3, & 0 \le d \le h \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

While this technique blurs the content of the visitation map and also leads to an increasing visitation percentage in voxels where multiple spherical footprints overlap, isosurfaces in the resulting scalar fields have a much smoother shape and give a much more pleasant visual impression.

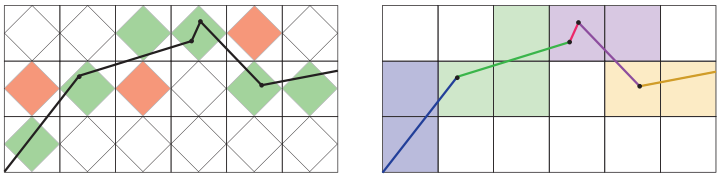

Figure 2. Trajectory voxelization. Left: The standard hardware rasterization cannot be applied due to the diamond exit rule. Voxels colored red indicate missing entries in the visitation map. Right: Our voxelization approach. Voxel colors correspond to the line segment that triggers the voxelization of a point primitive into the respective grid cell.

4. SAFETY REGIONS

One possible application of our proposed framework is the interactive visualization of uncertainty in white matter tractography. Probabilistic fiber tractography in combination with visitation maps can be used in surgical planning routines to estimate fiber bundles and, thus, to support surgeons in their preoperative planning phase. In such a scenario, next to regions with a high probability of presence for fiber bundles, safety margins are of special interest.

To facilitate the visualization of such safety regions, we employ the possibility to generate Euclidean distance transform volumes on the GPU at very high speed.[29] The distance transform is computed to particular visitation envelopes in the visitation map. The particular implementation we use is a GPU-optimized variant of the vector propagation approach presented in Ref. 33. It is close to exact, with an intrinsic error of 0.12 voxels.

An Euclidean distance transform volume for a given feature volume is generated by sweeping planes along the orthogonal axes of the volume. The technique presented in Ref. 29, however, is restricted to volumes of equal size in either dimension, as the authors only perform sweeps along the z-axis and rotate the volume twice to sweep along the x- and y-axis. Rotations are required because current GPUs only allow rasterizing primitives into z-aligned volume slices.

To be able to handle arbitrary volumes and to omit the additional fragment load introduced by rotating the volume between consecutive sweeps, we adapt the technique as follows. The initial sweep along the z-axis is performed by rasterizing a quadrilateral perpendicular to the sweep direction sequentially into the slices of the distance volume. Now, instead of rotating the volume to sweep a plane along another direction, we represent the sweep plane by a set of individual line primitives and raster one line in each of the z-aligned slices (see Fig. 3 for a graphical illustration).

Computing the distance transform starts by initializing a feature volume to which the transform is computed. Therefore, all voxels whose value in the visitation map exceeds a selected confidence value are marked. The shortest distances to these features are then computed for every voxel in the distance transform volume as described in Ref. 29. In Fig. 1 (right), Fig. 3 (right) and Fig. 7 a number of examples demonstrating the use of safety margins are shown.

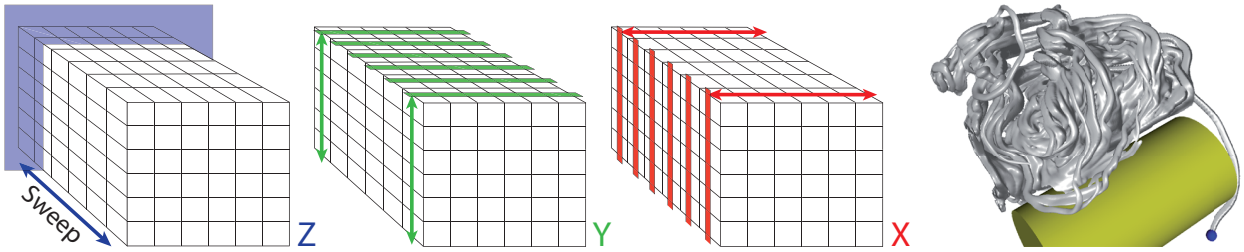

Figure 3. Modification to the GPU-based Euclidean distance transformation. Left: To avoid rotating the volume between consecutive passes, sweeps along the x- and y-axis are realized by splitting the sweep plane into individual line segments. Right: An exemplary safety region with one voxel distance to a particle pathway of low visitation percentage is shown.

5. VISUALIZATION

To visualize visitation envelopes and safety margins interactively, we have integrated a GPU-based volume ray-caster that supports direct volume rendering as well as the scale invariant rendering of isosurfaces as proposed in Ref. 4.

Since the proposed technique intertwines the incremental visitation map construction and visualization, we do not normalize the content of the visitation map. Instead, normalization (according to the total amount of trajectories extracted so far) is performed whenever a sample is queried from the data structure during rendering.

5.1 Visual Experiments

To validate the effectiveness of our proposed technique, we have conducted a number of experiments on different data sets from two application areas. All data sets employed during our tests were given on 3D Cartesian grids. In the following we will describe the data sets and techniques used to extract trajectories in detail.

Uncertain 3D flow trajectories

Extracting characteristic trajectories on the basis of lagrangian particle tracing is a well-established technique in the field of flow visualization. However, errors introduced during the execution of the visualization pipeline are often neglected as the user is presented only one of many possible realizations of a particle path in the data. Yet, a computed particle trajectory can significantly differ from the path a particle might take in reality and, thus, an analysis of individual pathways can lead to inaccurate or incorrect conclusions. In the following we present two exemplary experiments modeling errors introduced in the visualization stage as normal distributed directional probability for the Monte Carlo particle tracing stage. By doing so, the viewer is presented a range of possible variations of a particle pathway. Analyzing this range can be employed to optimize visualization specific parameters, in turn, increasing the reliability of the extracted features.

The following 3D flow field data set was used in two experiments:

- *Flow around a short, wall mounted round cylinder:* An LES simulation of incompressible turbulent flow around a short wall-mounted cylinder[34]. The size of the data grid is $256 \times 128 \times 128$. From 22 simulated time-steps we selected several snapshots with highly developed turbulence to visualize uncertain stream lines.

Exp1: The first experimental setup focuses on errors arising from the application of numerical integrators of specific order to compute particle traces in a fluid flow. Especially in interactive flow visualization environments, explicit low-order integrators are often used in conjunction with a fixed global step size to extract particle trajectories in real time.

To visualize errors introduced by numerical integration wrt. neglecting terms of higher order from the taylor series, we start particle traces from a single seed point and calculate new particle positions in the Monte Carlo particle tracing stage as follows; For every particle $\mathbf{p}_j$ in each advection step j, we employ two explicit Runge-Kutta integrators of different order to calculate two intermediate particle positions $\mathbf{p}_a$ and $\mathbf{p}_b$. Here, $\mathbf{p}_a$ corresponds to the new position obtained with an integrator of order a and $\mathbf{p}_b$ to a "ground truth" positions obtained with an integrator of higher order b. We now employ the integrator error vector $\mathbf{v}_{err} = \mathbf{p}_b - \mathbf{p}_a$ and a normal distributed input random value $r \in [0,\ldots,1]$ to choose a random position particle $\mathbf{p}_j$ moves to as

$$\mathbf{p}_{j+1} = \mathbf{p}_a + r \cdot \mathbf{v}_{err}.$$

With increasing order a $(a \leq b)$ the error vector diminishes and both integrators converge to the same solution. Orders a and b can be flexibly changed during interactive flow exploration, thus, allowing the user to find an appropriate integrator with respect to a given fixed step size Δt and the desired reliability regarding a ground truth integrator. Since our technique visualizes surfaces of certain visitation percentage in real-time, users can explore the field and find a suitable integration order for the whole flow domain by placing the seed position interactively close to regions of interest—e.g. critical points.

In Fig. 4 (left) isosurfaces of low (transparent grey) and high (opaque green) visitation percentages are shown. The three images were generated by starting Monte Carlo particle tracing from a fixed seed position, while the integration order a was increasingly set to 2,3, and 4 (from top to bottom). 256K trajectories were extracted in each image, b was set to 6. As can be seen in the lower image, for this specific seed position and a given global step size, the reliability of a fourth order integrator closely matches the soundness of the "ground truth" trajectory. To obtain the right image in Fig. 4, a was set to 3, b to 4, and a total of one million particle pathways was computed. The outer visitation envelope (transparent grey) corresponds to regions where at least one per mil of particles has passed through, the inner red surface to regions where at least forty percent of all trajectories reside. As can be seen, even while a wide range of possible stream lines was extracted, our approach allows to identify a pathway of high probability intuitively. This example demonstrates how the presented technique can be employed to detect pathways of high probability in uncertain flow fields, where a locally occurring directional variations is given by an application specific model.

Exp2: Our second experiment focuses on another source of uncertainty in the visualization pipeline. Errors introduced due to data transformation operations, such as resampling, interpolation or quantization either prior to or during the feature extraction process can drastically change the geometry and topology of extracted features. To demonstrate how the precision

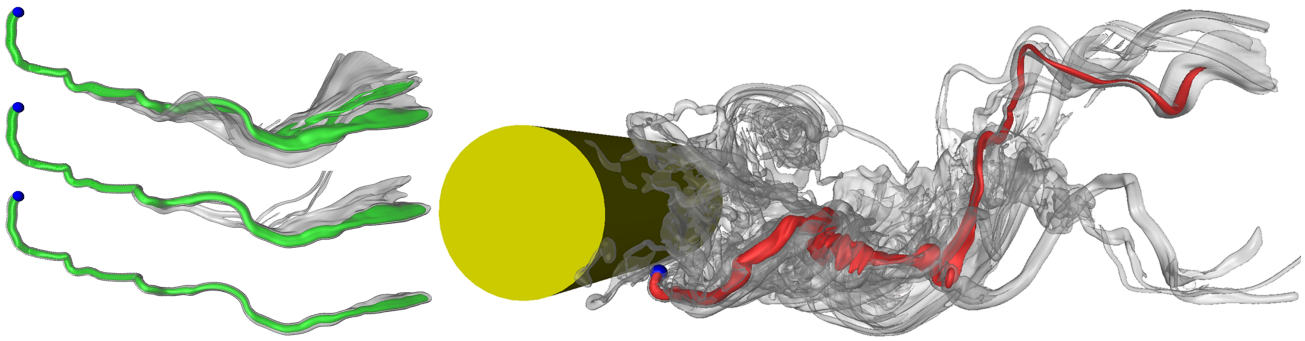

Figure 4. Visualization result for the experiment **Exp1**. Ranges of possible trajectories were generated during MC particle tracing by randomly positioning a particle along the integration error vector between two intermediate positions calculated with two integrators of varying order. Left: Three different ranges were extracted by combining integrators of increasing order 2,3 and 4 (from top to bottom) with a "ground truth" integrator of sixth order. As can be seen in the lower image, the reliability of the extracted feature merely increases by using a higher order integration scheme. Right: By visualizing isosurfaces of high visitation frequency (red), the regions a particle pathway is more likely to pass through can be depicted intuitively.

at which the vector field data or variables employed during particle tracing are stored can influence the visualization result, we proceed as follows; During MC particle tracing, whenever a velocity vector is fetched from the flow field, we employ a 3-tuple r of random variables to manipulate each of the vector components by replacing the last s bits of the significand by an arbitrary bit pattern. The data set used for this test is given at 32bit single floating point precision (23 fraction bits) and we employ an explicit Runge-Kutta scheme of fourth-order for numerical particle integration.

Both images in Figure 5 were generated by extracting 500K stream line trajectories and depict two visitation envelopes in the visitation map. The opaque red isosurfaces correspond to the locations where at least thirty percent of all trajectories have passed through, while the outer isosurfaces depict regions where at least five per mil of particles have passed through. As can be seen, even for a modeled small loss in precision (here, s was set to 7, which is less than the loss in precision introduced by switching from single to half precision), analyzing only a single trajectory might significantly mislead the viewer. Whereas the visualization of visitation envelopes allows to narrow down the probable location of the exact trajectory in an intuitive way.

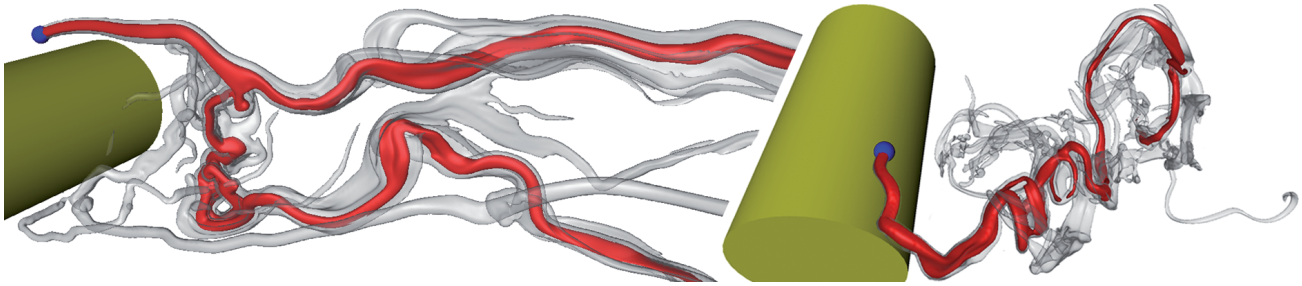

Figure 5. Visualization result for the experiment **Exp2**. To simulate errors introduced due to loss in precision through data transformation operations, we generate ranges of possible trajectories by randomly changing the last s bits of the significand of the floating point velocity vector components. Visualizing visitation envelopes of possible particle pathways shed light on the probable location of the exact trajectory, whereas analyzing a single trajectory might mislead the viewer during data exploration.

Uncertain 3D diffusion tensor fields

Visitation maps are often used in combination with probabilistic white matter tractography techniques to identify fiber bundles in the human brain. To the best of our knowledge, here visitation maps have not been used in an interactive context and are only visualized by a maximum intensity projection onto axis aligned clip planes. In the following experiment we will demonstrate how our approach can support experts to understand the spatial relation of extracted features by exploring visitation envelopes in 3D. The following data sets were used in our experiment:

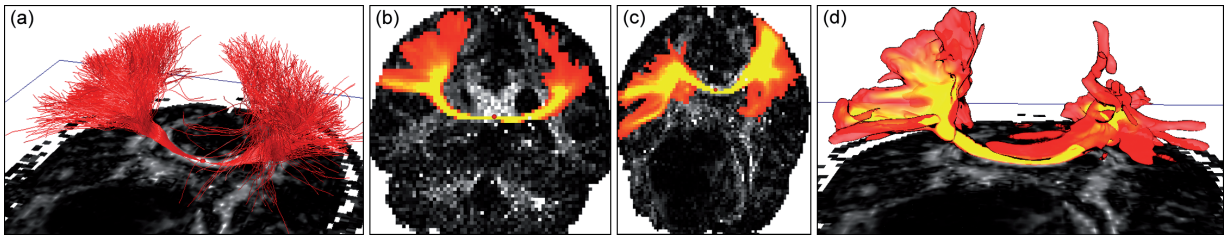

Figure 6. (a) 2K fiber pathways from a total of 1M were extracted in every frame. (b) Coronal and (c) axial clip planes with maximum intensity projections in logarithmic probability color scale. In (d) an isosurface of low visitation percentage is shown, inside the surface direct volume rendering was applied.

- *IEEE Visualization Contest 2010:* The Case1 and Case2 DTI data sets, each containing 30 diffusion-weighted gradients on a regular lattice at resolution $128 \times 128 \times 72$, were used to reconstruct a Diffusion Tensor field. Least Squares Fitting was applied to calculate the diffusion tensor data. Several of the supplementary data sets (CT, brain and tumor masks) were used during our visualization to provide contextual information. The data sets are courtesy of Prof. B. Terwey, Klinikum Mitte, Bremen, Germany.

Exp3: To extract pathways in the tensor fields, an eigen-decomposition of trilinear interpolated tensors was performed on the GPU to compute eigenvalues, eigenvectors and anisotropy coefficients. Our implementation uses the eigensolver algorithm proposed by Hasan et al. in Ref. 35. The tensorline diffusion-deflection technique as proposed by Weinstein et al. in Ref. 36 was applied to determine the traversal direction at a given point in the field. Furthermore, particle positions along the pathways were computed by traversing the grid from voxel to voxel.

To create different pathways emanating from a single seed point, in each advection step the traversal direction calculated with the tensorlines technique was randomly jittered by an angle of $0^\circ \leq d \leq 10^\circ$. We used a normal distributed importance sampling on the unit hemisphere around the traversal direction. Let us note that this setup was only used as a test case to demonstrate the usefulness of our technique in this application area and does not present a sound model for the probability distribution function of the underlying tensor data. Hence, the reader should not interpret the extracted fiber bundles as meaningful features in the data sets.

In Fig. 6 (d) the visualization of a transparent visitation envelope is shown. The inside of this region depicts direct volume rendering with a logarithmic probability transfer function. Images (b+c) show maximum intensity projection clip planes, which are commonly used in the field of probabilistic tractography. As can be seen, the exploration in 3D vastly improves the spatial relations between extracted features. Figures 1 (right) and 7 depict additional isosurface renderings of probabilistic trajectories in combination with safety margins.

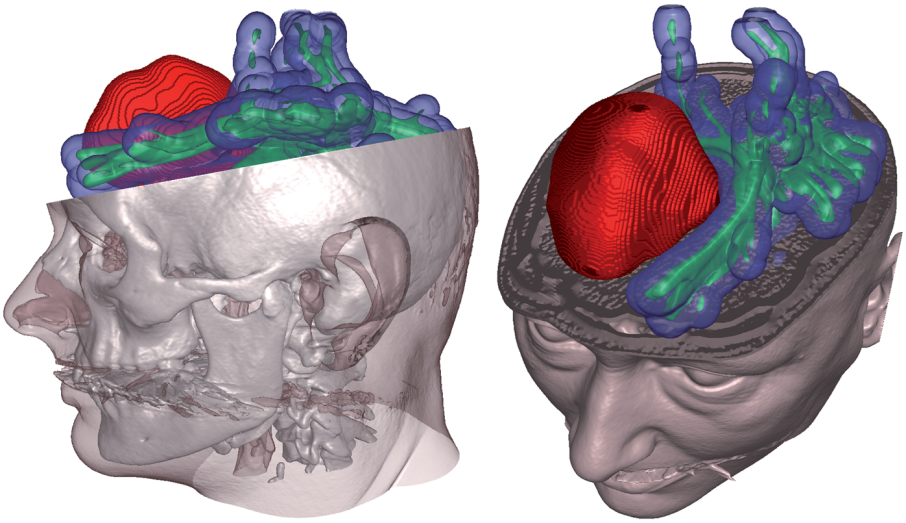

Figure 7. A visitation probability isosurface with an enclosing safety region is shown. Here, a distance transform surface is used to measure the distance from the fiber bundle to the tumor tissue.

6. PERFORMANCE

Performance statistics were measured on a 3.0 GHz Core 2 Quad processor, equipped with a NVIDIA Geforce GTX480 with 1536 MB local video memory. Results were rendered into a viewport at 1600×1200 resolution.

Performance measures for the Monte Carlo particle tracing are presented in Table 1. Representative timings in milliseconds (ms) are given in column *Time* for varying amounts (*Traces*) and lengths (*TraceLength*) of particle traces and the three discussed case studies (*Experiment*). We refer the reader to Section 5.1 for more detail on the employed integration schemes and Monte Carlo random walk models.

Table 1. Performance statistics for GPU-based MC particle tracing.

Experiment	Traces	TraceLength	Time (ms)
Exp1	512	512	6.9
Exp1	2048	512	9.4
Exp1	512	2048	23.8
Exp2	512	512	6.7
Exp2	2048	512	8.6
Exp2	512	2048	21.2
Exp3	512	128	2.7
Exp3	2048	128	3.3
Exp3	2048	256	4.8

Performance measures for the voxelization of particle traces into the visitation map are given in Table 2. Representative timings in milliseconds are given in column (*VoxelizationTime*) for visitation maps at varying spatial resolutions (*GridRes*) and different amounts of particle traces (*Traces*) at varying length (*TraceLength*). Timings are given for three different voxelization techniques, namely the voxelization of control points—i.e., particle positions—along a trajectory (*CP*), line segments (*LS*) and the splatting of spherical footprints at each control point (*SF*). Column labeled *EDT* lists timings (in ms) for the computation of Euclidean distance transform volumes.

Table 2. Performance statistics for particle voxelization and the computation of euclidean distance volumes.

Traces	Trace Length	GridRes	VoxelizationTime			EDT
			CP	LS	SF	
256	1024	$256 \times 128 \times 128$	0.7	6.7	8.0	70.3
1024	256	$256 \times 128 \times 128$	0.7	9.6	18.1	
1024	1024	$256 \times 128 \times 128$	1.7	24.2	32.0	
256	1024	$512 \times 256 \times 256$	2.1	8.6	8.6	383.8
1024	256	$512 \times 256 \times 256$	2.2	12.5	17.8	
1024	1024	$512 \times 256 \times 256$	2.9	31.3	34.2	
256	1024	$1024 \times 512 \times 512$	13.2	23.2	19.2	N/A
1024	256	$1024 \times 512 \times 512$	13.3	27.2	27.8	
1024	1024	$1024 \times 512 \times 512$	14.2	44.0	39.6	

Detailed performance measures for the presented experiments are given in Table 3. Column *Trajectories per MC iteration* lists the amount and length of trajectories extracted in every frame. Columns labeled *Visitation map update* and *Euclidean distance transform* give timings (in milliseconds) for the voxelization of particle traces and the computation of safety margins (when applied). Column *SpatialRes* lists the spatial resolution of the visitation map and the distance volume, respectively. Column (*Rendering*) lists timings for the isosurface rendering of visitation envelopes, safety margins and optional contextual information. Average frame rates (in frames per second) achieved during the experimentation sessions are given in Column (*Total average frame rate*).

Table 3. Performance statistics for the overall system performance for the presented experiments. Timings marked with an • correspond to the rendering time for visitation envelopes, safety margins and two additional context volumes.

Experiment	Trajectories per MC iteration (amount / length)	SpatialRes	Visitation map update (ms)	Euclidean distance transform (ms)	Rendering (ms)	Total average frame rate (FPS)
Exp1	1024 / 1024	$256 \times 128 \times 128$	31	70	48	6.3
Exp2	1024 / 1024	$256 \times 128 \times 128$	32	-	46	11
Exp3	2048 / 256	$128 \times 128 \times 72$	7	27	7 / 28•	22 / 14•
Exp3	2048 / 512	$256 \times 256 \times 144$	13	126	12 / 35•	6.6 / 5.3•

7. CONCLUSION

We have presented a novel approach for visualizing the positional variability of uncertain particle trajectories in 3D flow fields. Here we understand by uncertainty the mean deviation of the vector samples from a given value, and we have assumed that these deviations can be modeled via known probability functions. We have shown that in applications where these functions are known, and can be evaluated on the GPU, extremely efficient approaches for visualizing the uncertainty in particle pathways in the given fields are possible.

In particular, we have demonstrated interactive methods on the GPU for generating 3D visitation volumes by using Monte Carlo particle tracing in 3D flow fields. By combining the capabilities of recent GPUs to stream millions of particles through a flow field and to rasterize their footprints into a 3D texture, an instant generation of visitation volumes has been made possible. Hardware-accelerated volume rendering and distance transform computation allows visualizing trajectory envelopes for particle pathways as well as safety margins to these regions.

In the future we will extend our research into two different directions. Firstly, we will investigate appropriate models for uncertainty in flow fields, for instance, based on wild bootstrapping or stochastic models for the uncertainty in numerical simulations. Secondly, we will apply our techniques to reveal the possible variations of flow features that are derived from particle trajectories, such as Lagrangian Coherent Structures or streak lines.

REFERENCES

[1] Pang, A. T., Wittenbrink, C. M., and Lodha, S. K., "Approaches to uncertainty visualization," *The Visual Computer* **13**, 370–390 (1997).

[2] Johnson, C. and Sanderson, A., "A next step: Visualizing errors and uncertainty," *Computer Graphics and Applications, IEEE* **23**, 6–10 (sept.-oct. 2003).

[3] Johnson, C., "Top scientific visualization research problems," *IEEE Comput. Graph. Appl.* **24**(4), 13–17 (2004).

[4] Kraus, M., "Scale-invariant volume rendering," in [*In Proc. of IEEE Visualization*], 295–302, Academic Press (2005).

[5] Post, F. H., Vrolijk, B., Hauser, H., Laramee, R. S., and Doleisch, H., "The state of the art in flow visualisation: Feature extraction and tracking," *Computer Graphics Forum* **22**(4), 775–792 (2003).

[6] Jones, D. K. and Pierpaoli, C., "Confidence mapping in diffusion tensor magnetic resonance imaging tractography using a bootstrap approach," *Magn. Reson. Med* , 2005.

[7] Jones, D., "Tractography gone wild: Probabilistic fibre tracking using the wild bootstrap with diffusion tensor MRI," *Medical Imaging, IEEE Transactions on* **27**(9), 1268–1274 (2008).

[8] Schirski, M., Gerndt, A., van Reimersdahl, T., Kuhlen, T., Adomeit, P., Lang, O., Pischinger, S., and Bischof, C. H., "ViSTA Flowlib: A framework for interactive visualization and exploration of unsteady flows in virtual environments," in [*7th International Workshop on Immersive Projection Technology, 9th Eurographics Workshop on Virtual Enviroments*], 77–86 (2003).

[9] Shen, H.-W., Li, G.-S., and Bordoloi, U. D., "Interactive visualization of three-dimensional vector fields with flexible appearance control," *IEEE Transactions on Visualization and Computer Graphics* **10**(4), 434–445 (2004).

[10] Krüger, J., Kipfer, P., Kondratieva, P., and Westermann, R., "A particle system for interactive visualization of 3D flows," *IEEE Transactions on Visualization and Computer Graphics* **11**(6), 744–756 (2005).

[11] Bürger, K., Schneider, J., Kondratieva, P., Krüger, J., and Westermann, R., "Interactive visual exploration of instationary 3D-flows," in [*Eurographics/IEEE VGTC Symposium on Visualization (EuroVis)*], 251–258 (2007).

[12] Guthe, S., Gumhold, S., and Strasser, W., "Interactive visualization of volumetric vector fields using texture based particles," in [*Proceedings of WSCG*], **10**, 33–41 (2002).

[13] Schirski, M., Kuhlen, T., Hopp, M., Adomeit, P., Pischinger, S., and Bischof, C., "Efficient visualization of large amounts of particle trajectories in virtual environments using virtual tubelets," in [*VRCAI '04: Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry*], 141–147 (2004).

[14] Kondratieva, P., Krüger, J., and Westermann, R., "The application of gpu particle tracing to diffusion tensor field visualization," in [*IEEE Transactions on Visualization and Computer Graphics*], **0**, 10 (2005).

[15] Merhof, D., Sonntag, M., Enders, F., Nimsky, C., and Greiner, G., "Hybrid visualization for white matter tracts using triangle strips and point sprites," *IEEE Transactions on Visualization and Computer Graphics* **12**(5), 1181–1188 (2006).

[16] Lodha, S. K., Pang, A., Sheehan, R. E., and Wittenbrink, C. M., "Uflow: Visualizing uncertainty in fluid flow," *Visualization Conference, IEEE* **0**, 249 (1996).

[17] Otto, M., Germer, T., Hege, H.-C., and Theisel, H., "Uncertain 2D Vector Field Topology," *Computer Graphics Forum* **29**(2), 347–356 (2010).

[18] Otto, M., Germer, T., and Theisel, H., "Uncertain topology of 3D vector fields," in [*Pacific Visualization Symposium (PacificVis), 2011 IEEE*], 67–74 (march 2011).

[19] Otto, M., Germer, T., and Theisel, H., "Closed stream lines in uncertain vector fields," in [*SCCG 2011: Spring Conference on Computer Graphics*], (2011).

[20] Jeurissen, B., A., L., K., J. D., D., T. J., and J., S., "Probabilistic fiber tracking using the residual bootstrap with constrained spherical deconvolution," *Human Brain Mapping* **32**, 467–479 (March 2011).

[21] Lazar, M. and Alexander, A. L., "Bootstrap white matter tractography (BOOT-TRAC)," *NeuroImage* **24**(2), 524–532 (2005).

[22] Berman, J. I., Chung, S., Mukherjee, P., Hess, C. P., Han, E. T., and Henry, R. G., "Probabilistic streamline q-ball tractography using the residual bootstrap," *NeuroImage* **39**(1), 215–222 (2008).

[23] Maurer, C., Qi, R., and Raghavan, V., "A linear time algorithm for computing exact euclidean distance transforms of binary images in arbitrary dimensions," *IEEE Trans. Pattern Analysis and Machine Intelligence* **25**(2), 265–270 (2003).

[24] Hoff, K., T. Culver, J. K., Lin, M., and Manocha, D., "Fast computation of generalized Voronoi diagrams using graphics hardware," *ACM Trans. on Graphics* **18**(3), 277–286 (1999).

[25] Mauch, S., *Efficient algorithms for solving static Hamilton-Jacobi equations*, PhD thesis, California Institute of Technology, Pasadena, CA (Mar. 2003).

[26] Sigg, C., Peikert, R., and Gross., M., "Signed distance transform using graphics hardware," in [*IEEE Visualization*], 83–90 (2003).

[27] Sud, A., Otaduy, M., and Manocha, D., "DiFi: Fast 3D distance field computation using graphics hardware," *EG Computer Graphics Forum* **23**(3), 557–566 (2004).

[28] Rong, G. and Tan, T.-S., "Jump flooding in GPU with applications to Voronoi diagram and distance transform," in [*ACM Symp. Interactive 3D Graphics and Games*], 109–116 (2006).

[29] Schneider, J., Kraus, M., and Westermann, R., "GPU-based euclidean distance transforms and their application to volume rendering," in [*Selected papers of VISIGRAPP 2009, Communications in Computer and Information Science (CCIS) 68*], 215–228, Springer-Verlag Berlin Heidelberg (2010).

[30] Bürger, K., Krüger, J., and Westermann, R., "Direct volume editing," *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization / Information Visualization 2008)* **14**, 1388–1395 (November-December 2008).

[31] Amanatides, J. and Woo, A., "A fast voxel traversal algorithm for ray tracing," in [*In Eurographics 87*], 3–10 (1987).

[32] Fraedrich, R., Auer, S., and Westermann, R., "Efficient high-quality volume rendering of SPH data," *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization / Information Visualization 2010)* **16**, 1533–1540 (November-December 2010).

[33] Danielsson, P., "Euclidean distance mapping," *Computer Graphics and Image Processing* **14**, 227–248 (1980).

[34] Frederich, O., Wassen, E., and Thiele, F., "Prediction of the flow around a short wall-mounted cylinder using LES and DES," *Journal of Numerical Analysis, Industrial and Applied Mathematics (JNAIAM)* **3**(3-4), 231–247 (2008).

[35] Hasan, K. M., Basser, P. J., Parker, D. L., and Alexander, A. L., "Analytical computation of the eigenvalues and eigenvectors in DT-MRI," *Journal of Magnetic Resonance* **152**(1), 41–47 (2001).

[36] Weinstein, D., Kindlmann, G., and Lundberg, E., "Tensorlines: advection-diffusion based propagation through diffusion tensor fields," in [*Proceedings of the conference on Visualization '99: celebrating ten years*], *VIS '99*, 249–253, IEEE Computer Society Press, Los Alamitos, CA, USA (1999).