Fourier Opacity Optimization for Scalable Exploration

Irene Baeza Rojo, Markus Gross and Tobias Günther

Abstract—Over the past decades, scientific visualization became a fundamental aspect of modern scientific data analysis. Across all data-intensive research fields, ranging from structural biology to cosmology, data sizes increase rapidly. Dealing with the growing large-scale data is one of the top research challenges of this century. For the visual exploratory data analysis, interactivity, a view-dependent visibility optimization and frame coherence are indispensable. In this work, we extend the recent decoupled opacity optimization framework to enable a navigation without occlusion of important features through large geometric data. By expressing the accumulation of importance and optical depth in Fourier basis, the computation, evaluation and rendering of optimized transparent geometry become not only order-independent, but also operate within a fixed memory bound. We study the quality of our Fourier approximation in terms of accuracy, memory requirements and efficiency for both the opacity computation, as well as the order-independent compositing. We apply the method to different point, line and surface data sets originating from various research fields, including meteorology, health science, astrophysics and organic chemistry.

Index Terms—Scientific visualization, opacity optimization, Fourier approximation.

1 Introduction

Scientific visualization is an essential instrument of modern scientific computing, as it helps to discover new insights, which include the role of dark matter in the formation of galaxies [50], oceanic currents that drive the global climate [60], the neurological pathways in our brain [5], as well as the treatment of impaired blood flow in our veins [48], to name a few. Due to the increasing availability of powerful computational resources and the improvements in measurement technologies, the available data grows rapidly in all scientific disciplines. This lead to a serious scalability problem that becomes even more challenging in the future, making this one of the top research challenges of the 21^{st} century. Thus, in this paper, we focus on the scalable visual data exploration of general scientific data. When it comes to 3D data, an inherent visualization problem that is shared among all visual primitives (points, lines, surfaces and volumes) is the *occlusion problem* [20], [23]. That is, highly relevant information might be hidden behind unimportant geometry, which is determined by a user-defined importance value. In its severity, this ranges from small details being missed to the screen being fully cluttered, making a navigation through the data impossible. An appropriate selection of the visual primitives to display has to fulfill a number of goals to adequately convey the information in the data [20]. First, it needs to be interactive to enable a free exploratory navigation, which is challenging with increasing data set sizes. Second, the occlusions need to be removed in order to provide a view onto the relevant aspects of the domain, while still preserving the context. Choosing the visibility of the geometric primitives is a view-dependent problem and it should thus give frame-coherent results during the camera navigation.

- *Irene Baeza Rojo and Tobias Günther are with the Computer Graphics Laboratory, ETH Zürich.*
  E-mails: irene.baeza@inf.ethz.ch, tobias.guenther@inf.ethz.ch.
- *Markus Gross is head of the Computer Graphics Laboratory at ETH Zürich and director of Disney Research Zürich.*
  E-mail: grossm@inf.ethz.ch.

For line geometry, Günther et al. [20] formulated the adjustment of the line opacity as a global optimization problem with bounded variables. Their approach was later extended to animated lines [21], opacity optimization for surfaces [22] and extinction optimization for volumes [1]. Ament et al. [1] approximated the energy and reformulated the minimization in ray space, which enabled a significant acceleration for volume data, which was later carried into point, line and surface geometry by Günther et al. [23] in their decoupled opacity optimization. However, the decoupled technique still does not scale well for large amounts of geometric primitives (points, lines and surfaces), due to an order-dependence that involves the construction and sorting of fragment linked lists [62], which is ultimately bounded by the available memory. The order-dependence is required in two steps of the pipeline: first, for the determination of how much important geometry is in front or behind a given fragment, and second, for the correct front-to-back compositing of the final transparent fragments.

In this paper, we reformulate the decoupled opacity optimization into an order-independent algorithm that operates on a fixed and small amount of memory. This is achieved by approximating both the accumulated importance (to determine the amount of important geometry in front and behind) and the optical depth (to obtain the weights for the front-to-back compositing) along the view ray by a Fourier series. The superiority of Fourier approximations over simple binning approaches has been demonstrated by Jansen and Bavoil [27] for the shadowing of transparent geometry. In order to apply their Fourier approximation along light rays to the color compositing along view rays, we introduce a necessary normalization scheme that avoids the severe compositing artifacts that could otherwise be introduced by potential overshooting of the approximated signal. The main benefit of our approach is that for each pixel, the Fourier series can be constructed, evaluated and integrated efficiently in an order-independent way. With this, our Fourier opacity optimization is applicable to large data sets, as shown throughout the paper. Fig. 1 gives the first examples from a DT-MRI fiber tracking and a cosmological galaxy formation

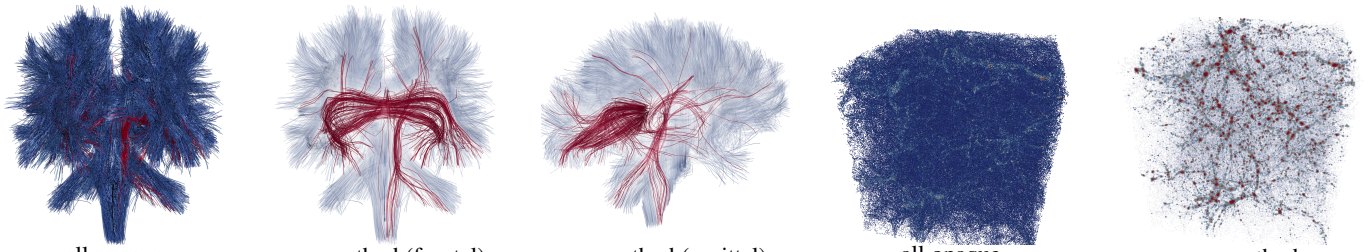


Fig. 1: We present a novel visualization technique that optimizes the visibility of scene elements based on a user-defined importance. In contrast to previous work, our formulation uses Fourier approximations to compute the visibility and to render the transparent geometry, both in an order-independent and memory-bound way. With this, opacity optimization becomes applicable to large scenes, as demonstrated here for a DT-MRI fiber tracking visualization of white-matter connectivity through the corpus callosum (left) and the cosmic web of 16 million stars in the DARK SKY simulation (right). Note that the DARK SKY simulation could not be rendered with decoupled opacity optimization [23] alone, because of its unbound memory consumption.

simulation. For the DARK SKY data set in Fig. 1 (right), the previous decoupled opacity optimization [23] is running out of memory, making it impossible to navigate through this data set. Even with a screen resolution small enough to render it with decoupled opacity optimization, our method needs only 4% of the memory and is 10× faster. In summary, our main contributions are:

- the first order-independent opacity optimization algorithm which utilizes a Fourier approximation to accumulate importance in front and behind a given fragment,
- an order-independent transparency (OIT) algorithm that extends a volumetric shadow mapping method named Fourier opacity mapping [27] to the blending of colored fragments along the view ray,
- and an unbiased normalization step that uses weighted averaging to correctly compose the fragments and blend them with the background.

With this, we obtain a scalable opacity optimization algorithm that operates within a fixed memory bound on all types of geometric primitives, i.e., points, lines and surfaces. By selecting the number of frequency bands in the Fourier approximation, the user can trade performance for accuracy. We carefully study the approximation quality of the Fourier-based method for both the importance accumulation and the order-independent compositing in detail by analyzing them independently. As we will show later, a small number of frequency bands is sufficient for the view-dependent assignment of transparency and its correct front-to-back compositing, for which we consider the decoupled opacity optimization [23] and the fragment linked-list compositing of Yang et al. [62] as reference solutions throughout the paper.

Section 2 outlines related work on visibility optimization in scientific visualization and introduces into the Fourier approximation. Section 3 presents our novel approach, for which implementation details are shared in Section 4. Our method is applied and evaluated in Section 5, followed by a conclusion in Section 6.

2 RELATED WORK

In this section, we summarize the recent visualization work on the occlusion problem and briefly introduce the Fourier foundations.

2.1 Visibility Optimization in Visualization

In visualization, data is often encoded by geometric primitives that are placed in the scene. The placement follows certain objectives. On the one hand, the geometric primitives should represent the underlying data well and on the other hand, they should not introduce too much occlusion. Next, we summarize recent work on line and surface placement, and discuss the state of the art in the visibility optimization of line, surface and volume data.

2.1.1 Line Placement and Selection

When it comes to the placement of flow geometry, we can distinguish between seeding algorithms and selection algorithms. Seeding algorithms determine the seed points of flow geometry, which typically aims for a certain density, the representation of features or the avoidance of redundancy. Selection algorithms, on the other hand, pick a subset of precomputed geometry, which is more suitable for interactive techniques. The first view-independent 3D seeding algorithms were extensions from 2D to 3D, including density-based [42], feature-based [63], [64], and similarity-based techniques [12], [45]. The first view-dependent seeding strategy was developed by Li and Shen [35], which was an extension of the seminal work of Jobard and Lefer's iterative streamline placement [28]. Annen et al. [2] computed streamlines that resemble surface contours. To maximize the information on the screen, an entropy-related strategy was used by Xu et al. [61]. Lee et al. [34] guided both line and camera selection by a maximum intensity projection of the entropy. Marchesin et al. [37] iteratively added lines based on the per-pixel fill rate and line entropy measures, which is not coherent during camera movement, i.e., small camera movements might lead to large changes in the chosen line set. To obtain frame coherence Günther et al. [19] used visibility statistics, which, however, favor lines closer to the camera. Using information theory, Tao et al. [55] selected lines and the viewpoint simultaneously by treating them as interrelated information channels. Ma et al. [36] merged lines from a view-independent and view-dependent candidate set by considering the coherence between local views and the last frame. For hemodynamics applications, Lawonn et al. [33] used a curvature-based segmentation into foreground and background flow in blood vessels. Recently, Kanzler et al. [29] selected lines from a precomputed line hierarchy, based on a maximum intensity projection of importance, depth and directional variation. Further, Kanzler et al. [30] proposed a voxel-based rendering method for large 3D line sets that encodes direction-preserving lines into a regular voxel grid and uses parallel GPU ray-casting to determine visible fragments in correct visibility order, which allows for interactive navigation with transparency and global illumination effects. In this paper, we extend the opacity optimization of Günther et al. [20], [21], which is a selection-based approach, i.e., it

operates on a precomputed line set. We examine their method later in Section 2.2. In principle, any of the above seeding methods (density-based, feature-based and similarity-based) can generate the geometry that is shown using our Fourier opacity optimization.

2.1.2 Surface placement and selection

The optimal placement of stream surfaces follows similar objectives as the streamline-based methods above, though with additional degrees of freedom: the orientation and shape of the seeding curve. Edmunds et al. [15], [16] proposed a dense evenly-spacing of stream surfaces. Martinez Esturo et al. [39] searched for a single best stream surface, such that principal surface curvature directions align with the underlying flow. The method was extended by Schulze et al. [51] to find multiple best surfaces. Other optimization criteria include the stretch-minimization by Barton et al. [3] and the flow alignment or orthogonality by Martinez Esturo et al. [40]. The above methods did not yet consider the potential occlusion that might be introduced by the geometry. Günther et al. [22] adjusted the transparency of surfaces to balance occlusion and visibility. Illustrative flow visualization [7] contributed methods to improve the perception of transparent surfaces, including the angle-based and normal variation mapping of Hummel et al. [26], the display of surface slabs and contours by Born et al. [6] and the diffusion of silhouettes and halos by Carnecky et al. [9].

2.1.3 Volume Visibility

In volume visualization, occlusion is typically treated by adjusting the transparency transfer function. To avoid occlusions, Viola and Gröller [56] used ghosting and cutaways. A number of approaches have been proposed for direct volume rendering. For instance, Viola et al. [57] suggested importance-based opacity mappings. Chan et al. [11] adjusted the opacity based on psychological principles. Marchesin et al. [38] included a relevance function into the volume rendering equation. Ament et al. [1] extended opacity optimization to volume data, which is detailed later. They not only optimized the extinction along view rays, but also along shadow rays.

2.2 Opacity Optimization

In this paper, we accelerate the decoupled opacity optimization of Günther et al. [23], which is a selection-based approach that adjusts the opacity of the scene geometry by an energy minimization that balances occlusion avoidance and the visibility of relevant parts of the data. The method is view-dependent, frame-coherent and interactive for moderate problem sizes. The following section describes the method in detail, since we build up on this approach.

2.2.1 Background

The energy that we minimize originates from the object-space optimization of Günther et al. [20], which was originally built for line geometry. Their method discretizes a given line set into equally-sized segments, each equipped with a user-defined importance. From fragment linked lists [62] that are obtained by rasterization of all lines, the mutual occlusion degrees are inferred to measure how much individual line segments occlude each other. Both the lines' importance and the occlusion degrees are input to a minimization that removes occluding lines, emphasizes relevant structures and ensures spatial smoothness. Later, Günther et al. [21] extended the method to time-dependent line geometry by adding a temporal smoothness term and by selecting the line discretization view-dependently from a pre-computed split tree. Afterwards, Günther et al. [22] extended the original method to surfaces. The minimization was still formulated as a global optimization problem. With their extension to an extinction optimization in volumetric data and with their reformulation into a local pixel-based minimization, Ament et al. [1] simplified the optimization significantly by spatially smoothing the input importance instead of the opacity. Günther et al. [23] applied this idea to points, lines and surfaces, which accelerated the opacity computation for the geometric primitives as well, but they have shown that the smoothing still has to be carried out in object space for line and surface geometry to avoid image-space discontinuities. A central building block remained: the construction and sorting of fragment linked lists, which are used to accumulate the amount of relevant information in front or behind a given fragment, and to blend the transparent scene geometry. In order to make the approach scalable for large scenes, we replace this former bottleneck with an order-independent approximation.

2.2.2 Energy Formulation

In decoupled opacity optimization [23], the optimal opacity $\alpha_i \in [0,1]$ of fragment i minimizes a quadratic energy E. As shown by Ament et al. [1], the optimal fragment opacities α_i are independent of each other. Thus, if the n fragments of a pixel are sorted from front to back, the quadratic energy becomes $E = \sum_{i=1}^{n} E_i(\alpha_i)$ with

$$E_i(\alpha_i) = \frac{p}{2}(\alpha_i - 1)^2 + \alpha_i^2\,(1-g_i)^{2\lambda}\left(\frac{q}{2}\sum_{j=1}^{i-1} g_j^2 + \frac{r}{2}\sum_{j=i+1}^{n} g_j^2\right). \tag{1}$$

The p-term is a regularization that penalizes an α_i unequal to 1 (opaque). Using the importance $g_i \in [0,1]$ of a fragment, the q-term penalizes the opacity of unimportant segments i (i.e., $1-g_i$ is large) that occlude an important segment j (i.e., g_j is large). Thereby, exponent λ steers the fall-off of g_i from 1, which allows the user to emphasize important structures. The r-term removes background clutter by fading out unimportant segments behind important ones. The energy weights $q \geq 0$, $r \geq 0$, and the emphasis exponent $\lambda > 0$ are set by the user, while we set $p = 1$ for normalization.

2.2.3 Analytic Solution

The optimal $\alpha_i \in [0,1]$ is found analytically by setting $\frac{\mathrm{d}E_i(\alpha_i)}{\mathrm{d}\alpha_i} = 0$ and rearranging for the fragment opacity α_i, cf. Ament et al. [1]:

$$\alpha_i = \frac{p}{p + (1-g_i)^{2\lambda}\left(r\sum_{j=1}^{i-1} g_j^2 + q\sum_{j=i+1}^{n} g_j^2\right)}. \tag{2}$$

Computing the sums $\sum_{j=1}^{i-1} g_j^2$ and $\sum_{j=i+1}^{n} g_j^2$ for geometric data [23] requires the construction and sorting of fragment linked lists, which is the bottleneck of this method. The construction is not only slow due to the atomic operations that are required to manage the access to the fragment memory pool, but also due to the complexity on the viewport, which is ultimately bounded by the available memory that is conservatively allocated to the memory pool. The visualization of large scenes with high depth complexity is thus prohibitively slow and memory intensive. In this paper, we lift these limitations by approximating the sums with Fourier series, which can be constructed and evaluated order-independently.

2.2.4 Object-Space Smoothing of Opacities

For line and surface geometry, the optimal fragment opacities α_i are smoothed across the geometry to ensure smooth transitions. We follow Günther et al. [23], who discretized lines into segments and surfaces into patches. First, the fragment opacities are mapped into

object-space, i.e., onto the segments and patches. As segment or patch opacity, they chose the smallest α_i of its rasterized fragments, which is a conservative choice to make a segment or patch disappear if at least one fragment needs to fade out in order to clear the view on something more important. Once the opacities arrived in object-space, they are smoothed with iterative Laplacian smoothing. Finally, the per-segment opacities are interpolated onto the vertices of the mesh. For this, the vertices of the lines and surfaces fade their opacity value towards the weighted average of the opacity values of the neighboring line segments or surface patches. The discretization into line segments and surface patches, as well as the interpolation weights of the vertices, are precomputed as described by Günther et al. [23]. In this paper, we are only concerned with the computation of the fragment opacities.

2.2.5 Transparent Rendering

The output of the opacity optimization is a transparency value for each vertex in the scene. For the final visualization, the transparent geometry has to be rendered for which several methods exist, cf. Maule et al. [43]. The correct blending methods require fragment sorting [10], [62] or many render passes [4], [17], which does not scale well for large scenes with high depth complexity. Several fast approximations were developed for real-time rendering to avoid the sorting, including weighted average [4], weighted sum [46] and the weighting with a scene-dependent analytic function [44]. These methods, however, do not consider the actual transmittance of light and therefore lead to an impaired perception of the surface order. For this reason, we follow a different strategy that is based on a volumetric shadow mapping approach by Jansen and Bavoil [27], named Fourier opacity mapping. They proposed to approximate the transmittance in participating media with a Fourier series, which they used to efficiently compute shadows. In this paper, we extend the method to the blending of colored fragments. The extension is not straightforward, since errors in the approximation lead to a noticeable saturation of the colors. To avoid these artifacts, we propose a normalized weighting scheme that keeps the colors in their convex hull. We prove that the weighting is identical to the correct blending methods in the case of zero approximation error. By adjusting the number of frequency bands in the Fourier approximation, our method is a scalable approach in-between the fast approximations and the slow ground truth methods.

2.3 Fourier Approximation of a General Function

In the following, we briefly introduce the Fourier approximation, which is the central building block of our method. The Fourier analysis found numerous applications in computer graphics, e.g., for ocean waves [41], depth of field [53] or shadows [27]. Next, we revisit the approximation of continuous and discrete sums.

2.3.1 Continuous Fourier Approximation

A continuous function $f(z)$ in the depth range $z \in [0,1]$ can be approximated by a Fourier series with m frequency bands, as shown by Jansen and Bavoil [27]:

$$f(z) \approx \frac{a_0}{2} + \sum_{k=1}^{m} a_k \cos(2\pi k z) + \sum_{k=1}^{m} b_k \sin(2\pi k z), \tag{3}$$

$$a_k = 2\int_0^1 f(z)\cos(2\pi k z)\,dz\,,\ b_k = 2\int_0^1 f(z)\sin(2\pi k z)\,dz, \tag{4}$$

where a_0 and a_k, b_k with $k \in \{1,\ldots,m\}$ are the coefficients of the Fourier basis functions. Once described in Fourier basis, the integral of the function $f(z)$ in Eq. (3) is conveniently calculated:

$$F(d) = \int_0^d f(z)\,dz \approx \frac{a_0}{2}\,d + \sum_{k=1}^{m} \frac{a_k}{2\pi k}\sin(2\pi k d) + \sum_{k=1}^{m} \frac{b_k}{2\pi k}\left(1-\cos(2\pi k d)\right). \tag{5}$$

Thereby, the bounds of the domain are reconstructed perfectly:

$$F(0) = 0 \qquad F(1) = \int_0^1 f(z)\,dz. \tag{6}$$

Eq. (5) shows the rate at which the Fourier terms decay when increasing m. The magnitude of the terms with coefficients a_k and b_k decays at a rate of $\frac{1}{\pi k}$ compared to a_0 [49]. Thus, terms for $k > 5$ have a magnitude of about 5% or less compared to a_0, and have thus a small impact on the reconstructed signal. An example of this decay is later shown visually in Fig. 9 for two pixels.

2.3.2 Discrete Fourier Approximation

In our application, we are interested in summing up a finite set of discrete values in a certain depth range. Thus, given n discrete values f_i at distance $d_i \in [0,1]$, our function $f(z)$ consists of Dirac deltas: $f(z) = \sum_{i=1}^{n} f_i \cdot \delta(z-d_i)$. Integrating $f(z)$ over the whole domain exactly recovers the sum of all values:

$$F(1) = \int_0^1 f(z)\,dz = \sum_{i=1}^{n} f_i, \tag{7}$$

since $\int_{-\infty}^{\infty} f(x)\delta(x-c)\,dx = f(c)$. Inserting Eq. (7) into Eq. (4) gives the Fourier coefficients in the discrete case:

$$a_k = 2\sum_{i=1}^{n} f_i \cos(2\pi k d_i)\quad,\quad b_k = 2\sum_{i=1}^{n} f_i \sin(2\pi k d_i). \tag{8}$$

Note that the approximation of $F(d)$ using Eq. (8) is independent of the sampling order. The integral from zero to a certain distance d_i recovers the sum of the fragments until f_i. Assuming they are sorted from front to back, i.e., $d_i < d_{i+1}$, this is formally:

$$F(d_i) = \int_0^{d_i} f(z)\,dz = \sum_{j=1}^{i} f_j. \tag{9}$$

Thus, after constructing $F(d)$ in Eq. (5) order-independently, the sum of f_i up until d_i can be evaluated without the need to store and sort the values f_i. The construction of $F(d)$ is memory-bound, as it requires only m frequency bands. The selection of m is a trade-off between performance and approximation quality, which is analyzed later in Section 5.5.

3 Fourier Opacity Optimization

3.1 Algorithm Overview

Our order-independent algorithm consists of two main steps, as illustrated in Fig. 2. First, for each pixel a Fourier approximation of the accumulated squared importance $G(d)$ is formed along the view ray. With this, objects can determine whether they occlude relevant information, which in turn steers the adjustment of their own opacity. Second, all transparent objects are rendered, for which we approximate the optical depth $\tau(d)$ along the view ray in Fourier basis. In the following, we elaborate on the two steps and the subsequent normalization and compositing with the background.

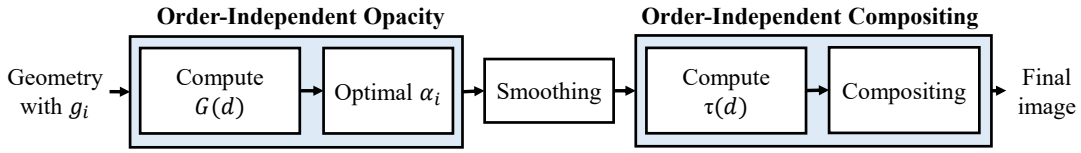


Fig. 2: Method overview: Input to our method is point, line and surface geometry, equipped with an importance value g_i. Using a Fourier series, we order-independently compute a representation of the accumulated squared importance $G(d)$ along the view ray, which is used to calculate the optimal opacity α_i of each fragment i. Using the object-space smoothing of Günther et al. [23], the opacity is smoothed across the line and surface geometry. Using another Fourier series, we order-independently approximate the optical depth $\tau(d)$ along the view ray, which is used for our order-independent transparency rendering to produce the final image.

3.2 Order-independent Opacity

To compute the optimal opacity, Günther et al. [23] constructed and sorted fragment linked lists, which is not scalable, as it becomes relatively slow and exhausts memory in large scenes. To overcome this, we rephrase the optimal fragment opacity α_i, given in Eq. (2). By denoting importance g_i at distance d_i, we obtain:

$$\alpha_i = \frac{p}{p + (1-g_i)^{2\lambda}\left(r\left[G(d_i) - g_i^2\right] + q\left[G(1) - G(d_i)\right]\right)}, \quad (10)$$

where $G(d_i) = \sum_{j=1}^{i} g_j^2$ follows from Eq. (9) and $G(1) = \sum_{i=1}^{n} g_i^2$ follows from Eq. (7). We approximate the importance sum $G(d)$ order-independently using

$$G(d) \approx \frac{a_0'}{2}\, d + \sum_{k=1}^{m} \frac{a_k'}{2\pi k}\, \sin(2\pi k d) + \sum_{k=1}^{m} \frac{b_k'}{2\pi k}\, (1 - \cos(2\pi k d))$$

$$a_k' = 2\sum_{i=1}^{n} g_i^2\, \cos(2\pi k d_i)\ , \quad b_k' = 2\sum_{i=1}^{n} g_i^2\, \sin(2\pi k d_i). \quad (11)$$

The coefficients a_k' and b_k' are calculated first by rendering all scene geometry with additive blending to sum up their contributions. The resulting coefficients are stored in multiple render targets, which are read in the subsequent pass to sample $G(d)$ in Eq. (10).

3.3 Order-independent Compositing

Once the optimal opacities are computed using Eq. (10) and smoothed in object-space using the method of Günther et al. [23], the transparent geometry has to be blended in the correct order. The blending equation for a front-to-back compositing is, cf. [24]:

$$C = \sum_{i=1}^{n} C_i\, \alpha_i \prod_{j=1}^{i-1}(1-\alpha_j)\ +\ C_{bg} \prod_{i=1}^{n}(1-\alpha_i) \quad (12)$$

where the n fragments with color C_i and alpha value α_i are sorted from front-to-back, and C_{bg} is the opaque background color. Note that the weight of color C_i depends on the alpha values of all fragments in front, i.e., the blending is order-dependent. Further note that the colors are *not* pre-multiplied with their alpha value. First, we apply an index shift from $i-1$ to i so that the left product ends at element i:

$$C = \sum_{i=1}^{n} C_i\, \frac{\alpha_i}{1-\alpha_i} \prod_{j=1}^{i}(1-\alpha_j)\ +\ C_{bg} \prod_{i=1}^{n}(1-\alpha_i). \quad (13)$$

By introducing the optical depth $\tau(d_i)$

$$\tau(d_i) = \sum_{j=1}^{i} -\ln(1-\alpha_j), \quad (14)$$

the products in Eq. (13) are reformulated into sums:

$$C = \sum_{i=1}^{n} C_i\, \frac{\alpha_i}{1-\alpha_i}\, e^{-\tau(d_i)}\ +\ C_{bg} \cdot e^{-\tau(1)}. \quad (15)$$

Then, optical depth $\tau(d)$ is approximated order-independently:

$$\tau(d) \approx \frac{a_0''}{2}\, d + \sum_{k=1}^{m} \frac{a_k''}{2\pi k}\, \sin(2\pi k d) + \sum_{k=1}^{m} \frac{b_k''}{2\pi k}\, (1 - \cos(2\pi k d)),$$

$$a_k'' = -2\sum_{i=1}^{n} \ln(1-\alpha_i)\, \cos(2\pi k d_i), \quad (16)$$

$$b_k'' = -2\sum_{i=1}^{n} \ln(1-\alpha_i)\, \sin(2\pi k d_i), \quad (17)$$

and can then be inserted into Eq. (15). Recalling Eq. (7), the optical depth $\tau(1)$ in Eq. (15) is always perfectly reconstructed, since

$$e^{-\tau(1)} = e^{-a_0''/2} = \prod_{i=1}^{n}(1-\alpha_i). \quad (18)$$

Because of the index shift in Eq. (13), we can evaluate the optical depth $\tau(d_i)$ directly at the location of fragment i, which means that we do not need to know the location of the fragment in front, i.e., d_{i-1}, as it would be the case if the Fourier approximation was directly applied to Eq. (12). The division by $(1-\alpha_i)$ in Eq. (13), however, is undefined for $\alpha_i = 1$, which is why we clamp all alpha values in practice to 0.9999. With our Fourier-based approach, the storing and sorting of fragments is avoided completely, since both the importance sum $G(d)$ and the optical depth $\tau(d)$ can be approximated order-independently.

3.4 Normalization and Background

The compositing described in Eq. (15) is correct under the assumption that the optical depth $\tau(d)$ is perfectly represented by the Fourier approximation. Since this is usually not the case, overshooting and undershooting of $\tau(d)$ may result in brightened or darkened colors, see Fig. 3. In order to avoid these artifacts, we introduce a normalization step. From Eq. (18), we know that the blending with the background is always correct, because $e^{-\tau(1)}$ is the *exact* transmittance of the background. For the fragments, we then introduce the normalization via weighted averaging which effectively composes the fragments into a single layer that is linearly blended with the background C_{bg}:

$$C = \frac{\sum_{i=1}^{n} C_i \frac{\alpha_i}{1-\alpha_i} e^{-\tau(d_i)}}{\sum_{i=1}^{n} \frac{\alpha_i}{1-\alpha_i} e^{-\tau(d_i)}} \cdot (1 - e^{-\tau(1)}) + C_{bg} \cdot e^{-\tau(1)}. \quad (19)$$

If $\tau(d)$ is perfectly approximated, then Eq. (19) is equivalent to Eq. (15), since $\sum_{i=1}^{n} \frac{\alpha_i}{1-\alpha_i} e^{-\tau(d_i)} = 1 - e^{-\tau(1)}$, which is proven in the additional material by induction. Thus, with an increasing number of frequency bands m, Eq. (19) converges to the correct solution.

4 Implementation

4.1 Preprocess

As shown by Günther et al. [23], discontinuities inevitably occur with pixel-based optimizations whenever lines or surfaces cross

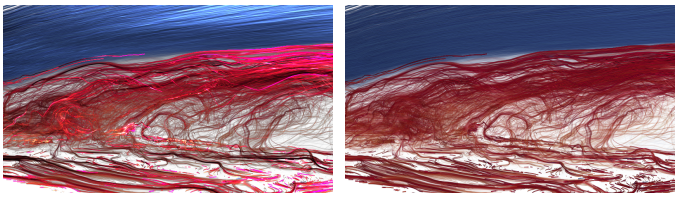


Fig. 3: Without normalization (left), colors might saturate due to overshooting. With normalization (right), the resulting color stays in the convex hull of the input colors due to a weighted averaging.

in screen-space. To remove this artifact, they smoothed the optimal opacity along the lines and across the surfaces. Following their approach, we thus form an object-space discretization in a preprocess. The lines are subdivided into equally-sized segments by using prefix sums [21]. Surfaces are subdivided into patches in two steps. First, points are uniformly distributed on the surfaces using geodesic farthest point sampling, which is based on geodesic distances [13]. Second, we use the corresponding Voronoi cells as the surface discretization [22]. With this, each vertex belongs to one segment or patch. In addition to the segment or patch ID, each vertex stores the distance to the adjacent line segments or surface patches, which is used at runtime for the interpolation of opacities from the line segments and surface patches onto the individual vertices of the lines and surfaces. The resulting per-vertex opacities are sent as vertex attributes into the rendering pipeline for the final blending of the result. Points do not require a preprocess, since each point is treated as independent element with its own opacity.

4.2 Runtime

At runtime, depth writing is disabled to process every fragment, while depth tests are enabled to discard fragments hidden behind context geometry. Each frame, we repeat the following steps:

1) Render all points, lines and surfaces and project their squared importance g_i^2 in the fragment shader at their depth d_i into the Fourier basis in Eq. (11). The Fourier coefficients a'_k and b'_k are summed up using additive blending and are stored in multiple render targets (two bands per RGBA float texture). Since the resulting opacities are smoothed later, this pass can be computed at half resolution to speed up the computation.
2) Render all geometries again and compute the optimal opacity α_i of each fragment using Eq. (10) by sampling the Fourier coefficients a'_k and b'_k from textures and computing $G(d_i)$ at fragment depth d_i. By passing the precomputed segment ID to the fragment shader, a segment opacity is computed via an atomic min operation of its rasterized fragment opacities.
3) Smooth the per-segment opacities using Laplacian smoothing and interpolate per-vertex opacities, cf. [23].
4) To render the transparent geometry, pass the vertex opacity to the fragment shader and project the opacities using Eqs. (16)–(17) into Fourier basis. Sum up the coefficients a''_k and b''_k with additive blending in multiple render targets to obtain a representation of the optical depth $\tau(d)$.
5) In the final pass, use the coefficients a''_k and b''_k to sample the optical depth $\tau(d)$ and compute the sums in Eq. (19) using additive blending. Then, draw a full-screen quad to perform the normalization and blending with the background. In this pass, we employ temporal anti-aliasing by jittering the camera to obtain higher quality.

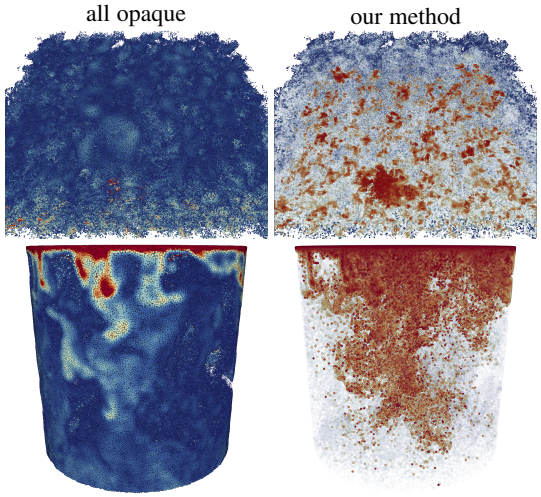


Fig. 4: Cumulonimbus clouds in the CTBL data set ($q = 100$, $r = 500$, $\lambda = 3$) and VISCOUS FINGERS in a liquid container ($q = 200$, $r = 50$, $\lambda = 2$). Left: all points are rendered opaque, which entails significant occlusion. Right: our method reveals the important structures.

When rendering very dense geometry, such as large point clouds, the Fourier approximations of $G(d)$ and $\tau(d)$ in steps (1) and (4) can be efficiently approximated with a $\frac{1}{n}$ subset of the full geometric data set, as shown later in Fig. 11. Note that for the final rendering, still all elements are visualized.

5 RESULTS

Next, we demonstrate the scalability of our Fourier opacity optimization for points, lines and surfaces in a wide range of applications. We compare our method with previous work on opacity optimization and order-independent transparency approximation, and evaluate the performance of our approach. The colors depict the importance from blue (low = 0) to red (high = 1). We refer to the video for interactive navigations through all data sets.

5.1 Applications

This section shows several application examples where we are able to navigate through the data interactively and visualize the important structures therein. The large number of fragments per pixel that need to be stored and sorted made previous methods [23] prohibitively slow or even unable to process the necessary geometry due to lack of memory. Fig. 1 (right) and Fig. 4 show our opacity optimization for dense point sets. In astrophysics, cosmological simulations of dark matter in the universe are typically discretized by particles that interact only gravitationally. In order to achieve adequate mass resolution, simulations require large numbers of particles, as in the $16.3M$ points DARK SKY data set. Here, star density is mapped to importance, revealing the cosmic web. In the CTBL data set, $3.7M$ particles were traced in a cloud resolving boundary layer simulation, which is used in atmospheric research to derive cloud statistics that parameterize and improve large-scale climate simulations. A key aerodynamic process in the development of cumulonimbus clouds is updraft, which is mapped to importance to reveal thunderstorm cells. The VISCOUS FINGERS data set is the result of a finite $1.7M$ point-based ensemble simulation of a salt layer dissolving in a liquid container. Using our optimization framework, the internal viscous fingers are made visible.

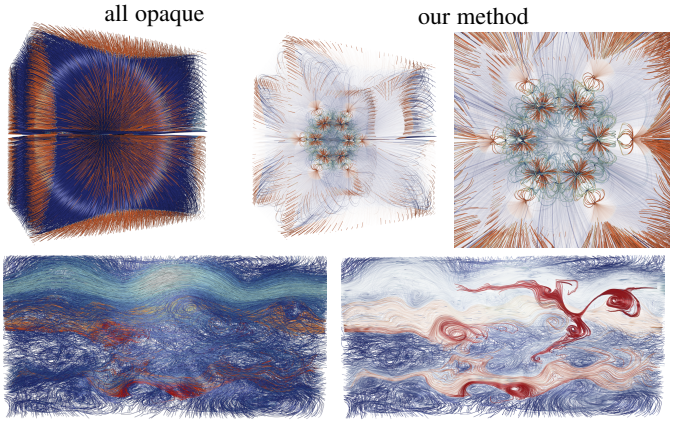


Fig. 5: Two large line data sets where streamlines show the magnetic vector potential of a BENZENE molecule (top, $q = 20$, $r = 50$, $\lambda = 10$) and a meteorological air flow simulation on ICON grids (bottom, $q = 400$, $r = 200$, $\lambda = 1.3$).

Fig. 1 (left) shows a DT-MRI data set, containing $18k$ fiber tracks, among which connections through the corpus callosum are mapped to importance, showing communication paths between the left and right hemispheres. In medical imaging, the visualization of diffusion tensor tractography allows to infer white-matter connectivity of the brain. Fig. 5 (top) shows an analytic approximation of a 3D potential field of a benzene molecule, containing $32k$ stream lines. In this visualization, the focus is set onto stream lines close to critical points, which represent the six carbon atoms (orange structures in the middle) and the six surrounding hydrogen atoms (orange structures around the teal field lines). Fig. 5 (bottom) depicts $11.5k$ atmospheric tracer trajectories in a meteorological reanalysis simulation. Our opacity optimization reveals the user-selected flow structures in red, while showing context geometry in blue. Figs. 3 and 9c use $40k$ streamlines to visualize the shear-induced turbulence that is created behind a backward facing step in the BFS data set, which is a common test case in fluid mechanics.

5.2 Comparison of Importance Approximation

Next, we compare the quality of the Fourier-based importance approximation $G(d)$ in Section 3.2 with the ground truth computation of the importance using fragment linked lists as done by Günther et al. [23]. To clearly separate the importance computation from all other subsequent steps (especially the smoothing), we directly visualize the resulting fragment opacities in Eq. (10) using fragment linked lists. This means that no subsequent smoothing via segment binning, iterative Laplacian smoothing or per-vertex interpolation take place. The results in Fig. 6 show that the Fourier approximation of the importance sum $G(d)$ leads to slightly brighter or darker fragments than in the linked list approach. This error, however, is small, indicating a sufficiently good approximation while achieving a performance speed-up. We refer to the additional material for more comparison examples.

5.3 Comparison with Blending Approximations

The rendering of transparent geometry with methods based on fragment linked lists [62] or ones that require many render passes [4] are not practical for scenes with large depth complexity. Common real-time approximations such as weighted average by Bavoil and Myers [4] and its extension with a correct background transmittance by McGuire and Bavoil [44] are fast, but lack visual

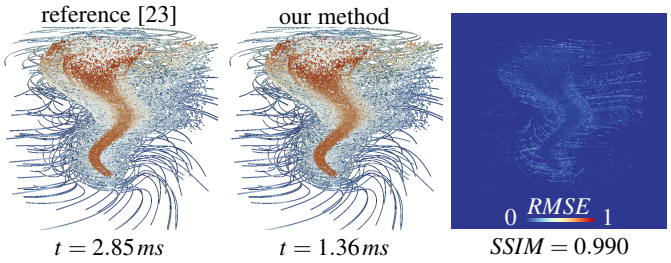


Fig. 6: Comparison of the importance approximation using our method with decoupled opacity optimization [23], showing the RMSE, the SSIM index and the GPU render time (t) for the same energy weights in the TORNADO ($q = 60$, $r = 500$, $\lambda = 2$) data set. Segment opacity computation via atomic min and per-segment smoothing (end of step 2 and step 3 of Section 4.2) are skipped and transmittance is computed using linked lists in both cases.

quality. We propose a blending based on Fourier approximations of the optical depth, which is a scalable middle ground between the coarse (and fast) approximations and the exact (and slow) calculations. For an equal-memory consumption test, we also compare our method with a binning approach that approximates $\tau(d)$ for each pixel with 11 bins along the view ray. When sampling $\tau(d)$, we linearly interpolate between the bins. See the additional material for more results and a comparison with piecewise constant bins. In Fig. 7, we show from left to right a reference solution using fragment linked lists by Yang et al. [62], the weighted averaging of Bavoil and Meyers [4], the extension of McGuire and Bavoil [44], the binning approach and our Fourier-based method. The top row depicts magnetic field lines of a topological decay simulation. The bottom row shows stream surfaces winding around a synthetic tornado. In both cases, our method preserves the layer order better than the other approximate techniques. The binning approach exhibits large errors at bin boundaries and cannot resolve the order within bins. Jansen and Bavoil [27] similarly observed that Fourier approximations achieve better quality than binning methods in the context of shadow computations. See the additional material for a repetition of Fig. 7 with $m = 2$ bands.

5.4 Comparison with Decoupled Opacity Optimization

Finally, we perform an end-to-end comparison of our complete method, using both Fourier approximations, compared with the linked list-based decoupled opacity optimization of Günther et al. [23], which is the reference solution that our method aims to approximate. Fig. 8 left compares our approximated images qualitatively and quantitatively, using the root-mean-square error (RMSE) and the structural similarity index (SSIM) [58], using surfaces in the wake turbulence of a DELTA WING. Further comparisons between the two methods using point and line geometry are provided in the additional material. We approximate the reference with almost imperceptible differences, which is also shown by the SSIM metric for all examples.

5.5 Parameter Study

5.5.1 Number of Coefficients

The number of frequency bands m used in the function approximation directly affects the quality of the reconstruction. Fig. 9 shows the approximation of function $G(d)$ at two different pixels of the BFS data set for varying number of bands m. As the number of bands increases, the approximation gets closer to the reference

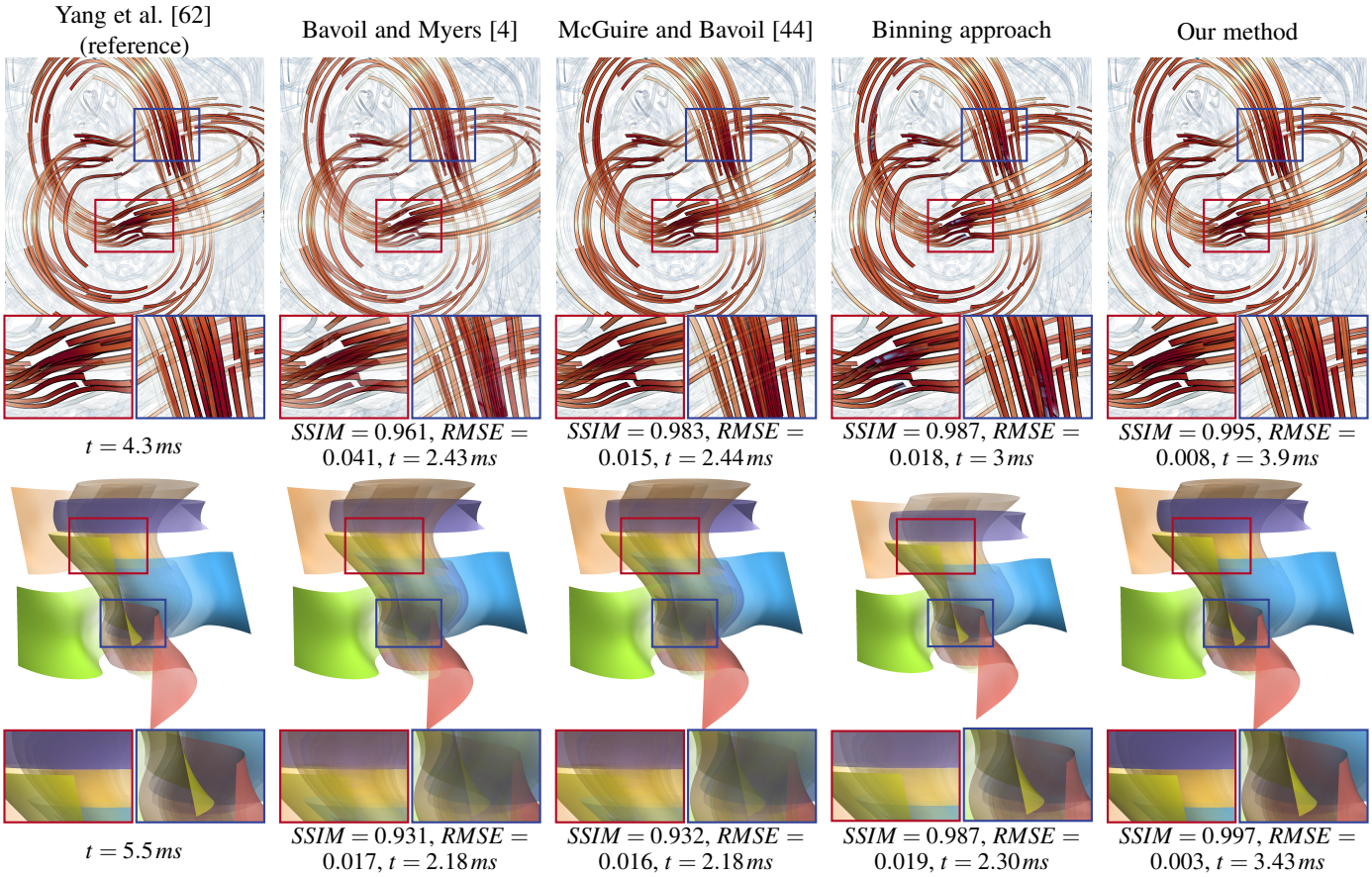


Fig. 7: Comparison of multiple blending approximations, from left to right: fragment linked lists (reference solution) by Yang et al. [62], weighted averaging by Bavoil and Myers [4], the extension with exact background term by McGuire and Bavoil [44], binning of $\tau(d)$ into 11 bins, and our Fourier-based method. The rows show the TREFOIL (top, $q = 50$, $r = 500$, $\lambda = 2$) and the TORNADO (bottom, $q = 10$, $r = 500$, $\lambda = 2$) data sets. We list the structural similarity index (SSIM), the root-mean-square error (RMSE) and the GPU render time (t) in *ms* for each data set. Bavoil and Myers [4] always require 19 MB of storage on a 1000×1000 pixel resolution and McGuire and Bavoil [44] need 23 MB. While all are faster than our method, they cannot retain the order of fragments, which makes a correct perception of the layer order impossible. This is especially noticeable with colored fragments. The binning method shows artifacts at bin boundaries that are specially noticeable for large depth domains (see additional material). Our method, on the other hand, approximates the depth-dependent transmittance better and resembles the reference more closely, as shown in the zoom-in images.

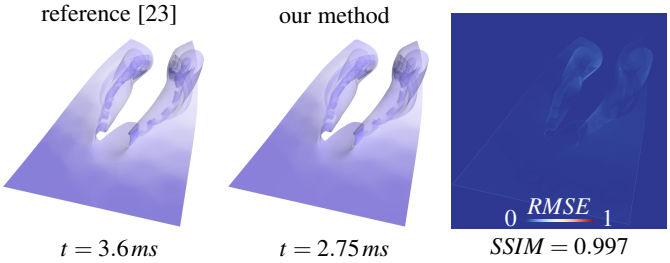


Fig. 8: Comparison with decoupled opacity optimization [23], showing RMSE, SSIM index and GPU render time (t) for the same energy weights in the DELTA WING ($q = 50$, $r = 10$, $\lambda = 2$).

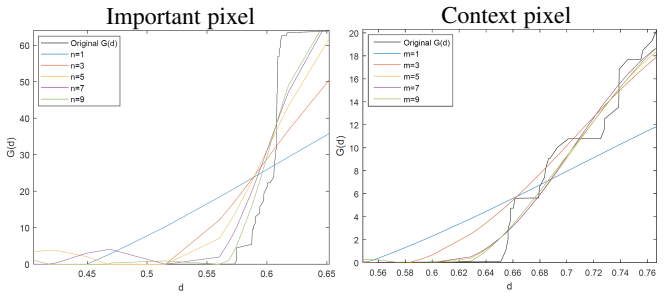
solution. We observed that using more than 5 bands does not significantly improve the quality, which is also evident in the coefficient plots in Fig. 9b by the small contribution of higher frequency bands. Thus in all examples, we used $m = 5$ frequency bands to approximate $G(d)$ and $\tau(d)$. This means we calculate the coefficients a_0, a_i and b_i for $i \in \{1, ..., 5\}$, which we store in three RGBA float render targets. See Fig. 10 for a visual comparison of different numbers of bands in the TREFOIL data set.

5.5.2 Discretization

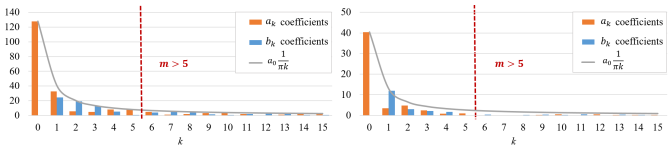
For the object-space smoothing of opacities, we discretize the lines into $20k - 150k$ equally-sized segments and the surfaces into roughly 500 equally-sized patches. We refer to Günther et al. [20], [22] for the implementation details of the discretization and also for visualizations using varying numbers of segments and patches. Note that in the object-space optimizations for lines [20], [21] and surfaces [22], the number of elements had a direct impact on the runtime of the optimization. With the pixel-based decoupled optimization [23] and in our Fourier-based approach, this is no longer the case, since the discretization only affects the subsequent smoothing. If more line segments or surface patches are used, the opacity can adapt more locally.

5.5.3 Subset for Fourier Approximation

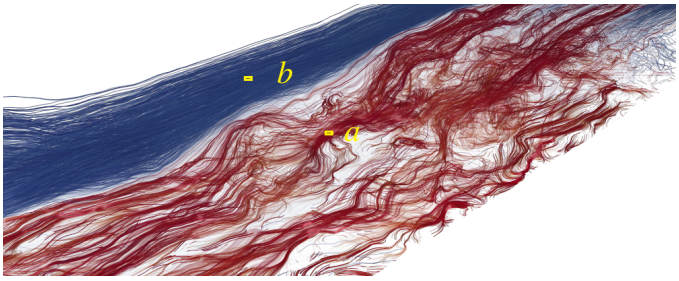
As shown in Section 5.6, our method is mainly bounded by overdraw. In dense scenes, such as the DARK SKY simulation, the Fourier approximation maps can be constructed by using only a fraction $\frac{1}{n}$ of the full data set, as long as it is representative for the importance and opacity distribution in the full scene. Note that in the final rendering pass, still the entire geometry is shown,



(a) Approximation of importance for different number of bands m.



(b) Convergence rate for increasing m. After $m = 5$ (red line), coefficients contribute less than 5% to the reconstructed signal.



(c) Analyzed pixels (a) and (b) in the BFS with $q = 20$, $r = 7$, $\lambda = 0.8$.

Fig. 9: Fourier approximations of the squared importance sum $G(d)$ for different numbers of Fourier frequency bands m at two pixels (important and context) of the BFS data set.

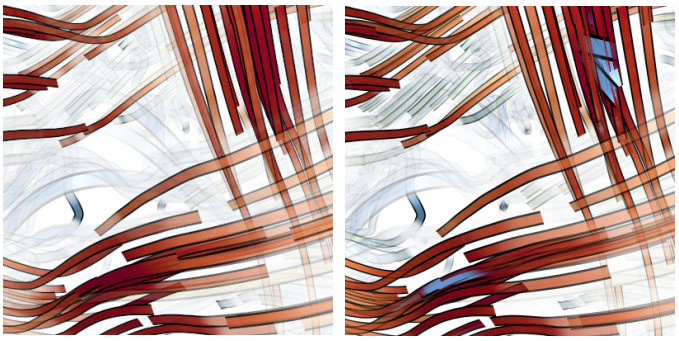
thus relevant information is not skipped. Fig. 11 shows the speed-up that is obtained when using a point subset for the Fourier approximations. The important structures (the star clusters of the cosmic web) remain visible, while only the visibility of the blue context is gradually reduced. Thus, when skipping too much (large n), unimportant parts are lost. We refer to the additional material for zoom images and additional results on another data set.

5.5.4 Energy Weights

The weights q, r, and λ of the energy in Eq. (10), as well as the smoothing iterations s, determine how clear the important structures are displayed. Fig. 12 shows a typical workflow of the parameter adjustment, which is consistent with the decoupled opacity optimization [23]. Given is a cluttered view with fully opaque geometry. When increasing the q-term in Eq. (1), unimportant lines (in blue) that are directly in front of important lines (red) become transparent. The λ exponent emphasizes the important lines. The r-term removes background clutter and finally, the opacities are iteratively smoothed, as in the decoupled opacity optimization [23]. We always use $s = 15$ smoothing iterations by default. Throughout the paper, the energy weights are listed in the captions. For more examples of varying parameters, we refer to Günther et al. [23].

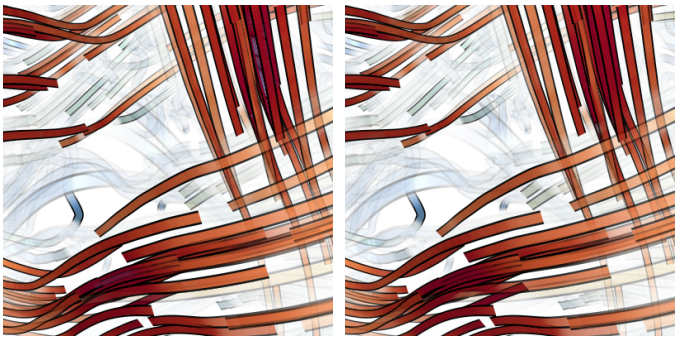
5.5.5 Importance

The importance is the main tool to declare which structures are relevant and should be visible. What may be considered important depends on the application and the analysis task. Thus, we do not prescribe any particular measure. The user-defined importance can be based on geometric properties (size, curvature), information theory (linear and angular entropy), statistics (directional variance) or application-specific properties (vorticity, distance to domain boundary). It can also prioritize certain types of geometry as in Fig. 6, where points have a higher importance than lines.



Yang et al. [62] (reference), $t = 7.5\,ms$

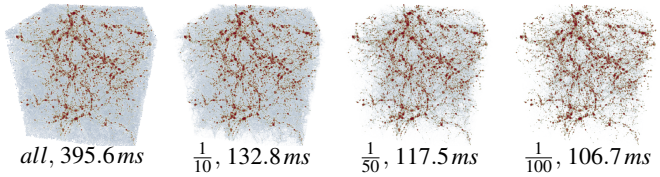
$SSIM = 0.959$, $RMSE = 0.036$, $m = 1$, $t = 3.0\,ms$



$SSIM = 0.975$, $RMSE = 0.032$, $m = 3$, $t = 3.1\,ms$

$SSIM = 0.984$, $RMSE = 0.027$, $m = 5$, $t = 4.3\,ms$

Fig. 10: Results for varying number of bands m in the TREFOIL data set ($q = 50$, $r = 500$, $\lambda = 2$, $s = 0$). When decreasing the number of bands, the approximated functions get smoother and are unable to represent strong importance changes as shown in the middle of the magnetic rings, where unimportant lines (blue) that are very close to important lines (red) are not transparent enough.



all, 395.6 ms $\frac{1}{10}$, 132.8 ms $\frac{1}{50}$, 117.5 ms $\frac{1}{100}$, 106.7 ms

Fig. 11: Results and total time when using varying subset sizes for the construction of $G(d)$ and $\tau(d)$ in the DARK SKY, for $q = 100$, $r = 20$, $\lambda = 1.3$. While the importance of the context is increasingly underestimated, the important structures remain visible.

5.6 Performance

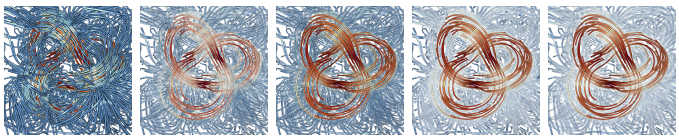
In this section, we measure and discuss the performance, memory consumption and complexity of our method and compare it with the decoupled opacity optimization [23]. All timings were measured on an Intel Core i7-4790K CPU with 4 GHz and 32 GB RAM, and an NVIDIA Quadro P6000 GPU.

5.6.1 Runtime and Memory

Table 1 shows the comparison of the total GPU time needed with the decoupled approach and with our method. In previous work, the optimal opacity and the final compositing were computed

Data set	Figure	Resolution	Decoupled OO [23]		Fourier Opacity Optimization						
			VRAM (bytes)	Total (*ms*)	Total (*ms*)	1	2	3	4	5.1	5.2
DARK SKY	Figs. 1 (right), 11	1000×1000	1'345'156'752	1065	106.71	5.5	28.2	1	6.17	**65.8**	0.04
		1500×1500	–	–	192.2	5.75	50.6	1.17	6.08	**127**	1.6
BENZENE	Fig. 5 (top)	1000×1000	552'823'360	145	16.94	1.51	2.15	0.14	**7.6**	5.43	0.11
		1500×1500	–	–	54.98	8.61	6.7	0.14	**25.6**	13.75	0.18
VISCOUS	Fig. 4 (bottom)	1000×1000	440'842'192	346	12.62	1.26	3.31	0.06	1.15	**6.8**	0.04
CBTL	Fig. 4 (top)	1000×1000	239'874'688	245	14.55	1.6	3.1	0.12	1.28	**8.4**	0.05
BFS	Figs. 3, 9	1000×1000	149'219'232	65.05	16.58	2.18	1.6	0.14	**7.8**	4.8	0.06
DT-MRI	Fig. 1 (left)	1000×1000	99'636'592	43.5	10.13	1.7	0.65	0.09	**5.12**	2.5	0.07
HELICOPTER	Additional material	1000×1000	36'725'664	6	2.62	0.32	0.31	0.03	1.15	**4.24**	0.11
ICON	Fig. 5 (bottom)	1000×1000	44'363'840	60	14.19	2.63	2.82	0.37	**4.02**	0.75	0.06
TREFOIL	Figs. 7 (top), 12	1000×1000	34'122'416	4.3	3.9	0.58	0.25	0.09	**1.91**	0.95	0.12
TORNADO	Fig. 7 (bottom)	1000×1000	24'201'104	5.5	1.46	0.22	0.11	0.12	**0.58**	0.36	0.07
ECMWF	Additional material	1000×1000	20590384	22.8	2.03	0.3	0.15	0.05	**0.96**	0.48	0.09
DELTA	Fig. 8	1000×1000	11570864	3.6	2.75	0.6	0.7	0.15	0.6	**0.6**	0.1

TABLE 1: Runtime in *ms* and memory consumption for the decoupled opacity optimization [23] and our method. Our method always operates with 46 MB to approximate a function with 5 bands for a resolution of 1000×1000 pixels, and with 103 MB at 1500×1500 pixels. The steps are: (1) projection of importance to Fourier, (2) optimal opacity by sampling Fourier maps, (3) per-vertex opacity smoothing and interpolation, (4) projection of opacity to Fourier, (5.1) accumulate colors using optical depth from Fourier maps, (5.2) normalization and compositing with background. The bottleneck is printed **bold**.



(a) Opaque geometry (b) Reduce occlusion $q = 50$ (c) Important lines $\lambda = 2$ (d) Reduce clutter $r = 500$ (e) Smoothing $s = 15$

Fig. 12: Examples for varying energy terms in the TREFOIL data set. From left to right, we subsequently add one term to the energy minimization with final weights $q = 50$, $r = 500$, $\lambda = 2$ and $s = 15$.

by sorting all rasterized fragments by depth. The storage and sorting of all fragments introduced a bottleneck in the render time and exhausted memory on large data. For instance, the DARK SKY simulation used $1.25\,GB$ of the allocated memory pool to store the rasterized fragments. By using the two Fourier approximations, the computations are up to $10\times$ faster and avoid memory problems, since we only need to store the coefficients of the approximated functions, which are bound to $46\,MB$ for $m = 5$ bands at 1000×1000 pixels. For two data sets, we also show the runtime at higher resolution (1500×1500), where the previous method [23] runs out of memory. For our approach, we list detailed measurements for each pipeline step, described in Section 4, indicating the bottleneck in bold letters, which was in almost all cases either the approximation of $\tau(d)$ or the final compositing of all fragments. This is because these render passes are done at full resolution, while the importance approximation is computed at half resolution. The video displays the performance statistics during camera navigation, showing that the performance depends as expected on the pixel fill rate due to the rasterization.

5.6.2 Time and Space Complexity

Both the approximation of the importance sum $G(d)$ (Section 3.2) and the optical depth $\tau(d)$ (Section 3.3) have a time complexity of $O(nm)$, where n is the number of fragments per pixel and m is the number of bands used in the Fourier series. No serialization and no unordered memory accesses are required and the approximation can be accelerated by using subsets of the geometry, as done for the DARK SKY ($1/100$), the VISCOUS FINGERS ($1/30$) and the cloud topped boundary layer CTBL ($1/100$). Existing methods for the rendering of transparent geometry are either very approximate [4], [46] (they do not preserve depth information, which is a strong limitation for visualization purposes) and therefore require only $O(n)$ in time, or they compute a ground truth solution which requires n render passes [4], [17] or the creation and full sorting of fragment linked lists [10], [62], which is at $O(n \log n)$. The latter requires serialized memory accesses during the linked list construction and a large number of unordered memory accesses during the sorting, which is not optimal for a GPU implementation. Further, the ground truth solution has an unbound space complexity of $O(n)$, whereas we only need $O(m)$ for a small constant m, making it possible to render large scenes. By adjusting the number of frequency bands m, our method is a scalable middle ground between existing approximate solutions and ground truth solutions.

5.7 Discussion

5.7.1 Approximation Artifacts

The Fourier approximation of a high-frequency signal with only few bands results in a coarse approximation with potential ringing artifacts. This might become noticeable whenever sudden importance changes are exhibited along the view ray. In these cases, our method can underestimate or overestimate the values surrounding the change in the approximated function. Fig. 10 shows an example of the possible artifacts introduced by the approximation. A poor approximation of the importance can lead to visible unimportant lines (in blue) that are too close to important lines (in red) therefore receiving an incorrectly high opacity. These artifacts are often reduced by the subsequent object-space smoothing, as can be seen in the additional material for a version of Fig. 7 with only $m = 2$ bands. Overall, for a sufficiently high m our results are visually indistinguishable from the reference solution [23], since the human perception is not sensitive to subtle changes in transparency, which comes to our advantage as it allows for approximations. When using only a subset of the data (Section 5.5.3), artifacts in the form of faded context geometry can also occur. Close-ups of this effect can be seen in the additional material.

5.7.2 Decay Rate of Coefficients

In general, the rate at which the magnitude of the Fourier coefficients decays is related to the smoothness of the function to approximate [59]. Our changes in the transparency, modeled

as Dirac functions, become a piece-wise constant function after integration. Since $F(d)$ has discontinuities, the rate of decay in the k^{th}-Fourier coefficient is proportional to $\frac{1}{k}$ [49]. As shown in Section 2.3.1, it can be seen by Eq. (5) that the terms a_k and b_k even decay at a rate of $\frac{1}{\pi k}$ and thus, components for $k > 5$ have a magnitude of about 5% or less compared to a_0. Throughout our experiments, we found that it is safe to set $n \gg m$. As shown in Fig. 9, increasing m does not noticeably improve the quality.

5.7.3 Transparency Issues

Our optimization is image-based and therefore only adjusts the opacity of the content that is visible on the screen, keeping the invisible parts opaque. During camera navigation, the additional temporal smoothing can cause the previously clipped geometry to appear opaque before its transparency is adjusted. Alternatively, it would be possible to initialize the geometry as invisible or to internally enlarge the viewport and thus conservatively adjust the opacity of nearby off-screen elements as well. Another known visual issue with faded transparencies is that it can be confused with depth cueing. Possible ways to side step the problem include the addition of visual cues such as halos [9] or to modulate the line width [29]. In our work, we implemented depth-dependent halos of Everts et al. [18] to indicate line crossings.

6 Conclusions

The interactive exploration of large scientific data is essential to advance research in numerous scientific disciplines, including meteorology, health science, astrophysics and many more. We introduced a general opacity optimization algorithm that enables an interactive, view-dependent and frame coherent navigation through large data sets, containing points, lines and surfaces. For this, we computed Fourier approximations of the accumulated squared importance and the optical depth in an order-independent and memory-bound way, which are needed to compute the optimal fragment opacity and to blend the transparent layers. With our method, occlusion is avoided and relevant objects are made visible.

In the future, we extend our method to time-dependent data, which requires fast geometry subdivision for object-space smoothing. In addition, we would like to explore the automatic placement of geometry during the navigation, which will allow for an unlimited level of detail, instead of using precomputed geometry.

Acknowledgments

The DARK SKY simulation was made available by Skillman et al. [52] and was subject of the IEEE SciVis Contest 2015. The BENZENE molecule data set was kindly provided by Zöckler et al. [65]. Kuhnert et al. [31] provided the VISCOUS FINGERS ensemble simulation for the The IEEE SciVis Contest 2016. The CTBL simulation of a cloud resolving boundary layer was provided by Stevens et al. [54]. The backward facing step flow BFS was kindly provided by Niemann and Fröhlich [47]. The DT-MRI data set was acquired at the Central Hospital of Bremen. The ICON simulation is part of the HD(CP)^2 project and was kindly provided by Heinze et al. [25]. The flow around a HELICOPTER in slow forward flight close to the ground was provided by Kutz et al. [32]. The TREFOIL data set was kindly provided by Candelaresi and Brandenburg [8]. The ECMWF data set is part of the ERA-interim reanalysis of Dee et al. [14]. The DELTA WING flow was provided by Markus Rütten. This work was supported by the Swiss National Science Foundation (SNSF) Ambizione grant no. PZ00P2_180114.

References

[1] M. Ament, T. Zirr, and C. Dachsbacher. Extinction-optimized volume illumination. *IEEE Transactions on Visualization and Computer Graphics*, 23(7):1767–1781, July 2017.

[2] T. Annen, H. Theisel, C. Rössl, G. Ziegler, and H.-P. Seidel. Vector field contours. In *Proc. Graphics Interface*, pages 97–105, 2008.

[3] M. Bartoň, J. Kosinka, and V. M. Calo. Stretch-minimising stream surfaces. *Graphical Models*, 79:12 – 22, 2015.

[4] L. Bavoil and K. Myers. Order independent transparency with dual depth peeling. Technical report, NVIDIA Research, 2008.

[5] J. Beyer, M. Hadwiger, A. Al-Awami, W.-K. Jeong, N. Kasthuri, J. W. Lichtman, and H. Pfister. Exploring the connectome: Petascale volume visualization of microscopy data streams. *IEEE Computer Graphics and Applications*, 33(4):50–61, July 2013.

[6] S. Born, A. Wiebel, J. Friedrich, G. Scheuermann, and D. Bartz. Illustrative stream surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1329–1338, 2010.

[7] A. Brambilla, R. Carnecky, R. Peikert, I. Viola, and H. Hauser. Illustrative flow visualization: State of the art, trends and challenges. In *EuroGraphics 2012 State of the Art Reports (STARs)*, pages 75–94, 2012.

[8] S. Candelaresi and A. Brandenburg. Decay of helical and nonhelical magnetic knots. *Phys. Rev. E*, 84:016406, 2011.

[9] R. Carnecky, R. Fuchs, S. Mehl, Y. Jang, and R. Peikert. Smart transparency for illustrative visualization of complex flow surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 19(5):838–851, 2013.

[10] L. Carpenter. The A-buffer, an antialiased hidden surface method. *SIGGRAPH Comput. Graph.*, 18(3):103–108, Jan. 1984.

[11] M.-Y. Chan, Y. Wu, W.-H. Mak, W. Chen, and H. Qu. Perception-based transparency optimization for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1283–1290, 2009.

[12] Y. Chen, J. Cohen, and J. Krolik. Similarity-guided streamline placement with error evaluation. *IEEE Transactions on Visualization and Computer Graphics*, 13:1448–1455, 2007.

[13] K. Crane, C. Weischedel, and M. Wardetzky. Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 32(5):152:1–152:11, 2013.

[14] D. P. Dee, S. M. Uppala, A. J. Simmons, P. Berrisford, P. Poli, et al. The ERA-Interim reanalysis: configuration and performance of the data assimilation system. *Quarterly Journal of the Royal Meteorological Society*, 137(656):553–597, 2011.

[15] M. Edmunds, R. Laramee, R. Malki, I. Masters, T. Croft, G. Chen, and E. Zhang. Automatic stream surface seeding: A feature centered approach. *Computer Graphics Forum (Proc. EuroVis)*, 31(3):1095–1104, 2012.

[16] M. Edmunds, R. S. Laramee, G. Chen, E. Zhang, and N. Max. Advanced, automatic stream surface seeding and filtering. In *Theory and Practice of Computer Graphics*, pages 53–60, 2012.

[17] C. Everitt. Interactive order-independent transparency. Technical report, NVIDIA Corporation, 2001.

[18] M. H. Everts, H. Bekker, J. B. T. M. Roerdink, and T. Isenberg. Depth-dependent halos: Illustrative rendering of dense line data. *IEEE Transactions on Visualization and Computer Graphics*, 15:1299–1306, 2009.

[19] T. Günther, K. Bürger, R. Westermann, and H. Theisel. A view-dependent and inter-frame coherent visualization of integral lines using screen contribution. *Proc. Vision, Modeling, and Visualization (VMV)*, pages 215–222, 2011.

[20] T. Günther, C. Rössl, and H. Theisel. Opacity optimization for 3D line fields. *ACM Transaction on Graphics (Proc. SIGGRAPH)*, 32(4):120:1–120:8, 2013.

[21] T. Günther, C. Rössl, and H. Theisel. Hierarchical opacity optimization for sets of 3D line fields. *Computer Graphics Forum (Proc. Eurographics)*, 33(2):507–516, 2014.

[22] T. Günther, M. Schulze, J. Martinez Esturo, C. Rössl, and H. Theisel. Opacity optimization for surfaces. *Computer Graphics Forum (Proc. EuroVis)*, 33(3):11–20, 2014.

[23] T. Günther, H. Theisel, and M. Gross. Decoupled opacity optimization for points, lines and surfaces. *Computer Graphics Forum (Proc. Eurographics)*, 36(2):153–162, 2017.

[24] M. Hadwiger, P. Ljung, C. R. Salama, and T. Ropinski. Advanced illumination techniques for GPU-based volume raycasting. In *ACM SIGGRAPH 2009 Courses*, SIGGRAPH '09, pages 2:1–2:166, New York, NY, USA, 2009. ACM.

[25] R. Heinze, A. Dipankar, C. C. Henken, C. Moseley, O. Sourdeval, et al. Large-eddy simulations over germany using ICON: a comprehensive evaluation. *Quarterly Journal of the Royal Meteorological Society*, 143(702):69–100, 2017.

[26] M. Hummel, C. Garth, B. Hamann, H. Hagen, and K. I. Joy. IRIS: Illustrative rendering for integral surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1319–1328, 2010.

[27] J. Jansen and L. Bavoil. Fourier opacity mapping. In *Proc. Symposium on Interactive 3D Graphics and Games*, pages 165–172, New York, NY, USA, 2010. ACM.

[28] B. Jobard and W. Lefer. Creating evenly-spaced streamlines of arbitrary density. *Proc. Eurographics Workshop on Visualization in Scientific Computing*, 7:45–55, 1997.

[29] M. Kanzler, F. Ferstl, and R. Westermann. Line density control in screen-space via balanced line hierarchies. *Computers & Graphics*, 61:29–39, 2016.

[30] M. Kanzler, M. Rautenhaus, and R. Westermann. A voxel-based rendering pipeline for large 3D line sets. *IEEE Transactions on Visualization and Computer Graphics*, page to appear, 2018.

[31] J. Kuhnert. *Meshfree Numerical Schemes for time dependent Problems in Fluid and Continuum Mechanics*, pages 119–136. Ane Books, 2014.

[32] B. M. Kutz, U. Kowarsch, M. Kessler, and E. Krämer. Numerical investigation of helicopter rotors in ground effect. In *30th AIAA Applied Aerodynamics Conference*, 2012.

[33] K. Lawonn, T. Günther, and B. Preim. Coherent view-dependent streamlines for understanding blood flow. In *EuroVis - Short Papers*, pages 19–23, 2014.

[34] T.-Y. Lee, O. Mishchenko, H.-W. Shen, and R. Crawfis. View point evaluation and streamline filtering for flow visualization. In *Proc. IEEE Pacific Visualization*, pages 83–90, 2011.

[35] L. Li and H.-W. Shen. Image-based streamline generation and rendering. *IEEE Transactions on Visualization and Computer Graphics*, 13:630–640, 2007.

[36] J. Ma, C. Wang, and C.-K. Shene. Coherent view-dependent streamline selection for importance-driven flow visualization. *Proc. SPIE 8654, Visualization and Data Analysis*, 2013.

[37] S. Marchesin, C.-K. Chen, C. Ho, and K.-L. Ma. View-dependent streamlines for 3D vector fields. *IEEE Transactions on Visualization and Computer Graphics*, 16:1578–1586, 2010.

[38] S. Marchesin, J. M. Dischler, and C. Mongenet. Per-pixel opacity modulation for feature enhancement in volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 16(4):560–570, 2010.

[39] J. Martinez Esturo, M. Schulze, C. Rössl, and H. Theisel. Global selection of stream surfaces. *Computer Graphics Forum (Proc. Eurographics)*, 32(2):113–122, 2013.

[40] J. Martinez Esturo, M. Schulze, C. Rössl, and H. Theisel. Poisson-based tools for flow visualization. In *IEEE PacificVis*, pages 241–248, 2013.

[41] G. A. Mastin, P. A. Watterberg, and J. F. Mareda. Fourier synthesis of ocean scenes. *IEEE Computer Graphics and Applications*, 7(3):16–23, March 1987.

[42] O. Mattausch, T. Theußl, H. Hauser, and E. Gröller. Strategies for interactive exploration of 3D flow using evenly-spaced illuminated streamlines. In *Proc. Spring Conference on Computer Graphics (SSCG)*, pages 213–222. ACM, 2003.

[43] M. Maule, J. L. Comba, R. P. Torchelsen, and R. Bastos. A survey of raster-based transparency techniques. *Computers & Graphics*, 35(6):1023–1034, 2011.

[44] M. McGuire and L. Bavoil. Weighted blended order-independent transparency. *Journal of Computer Graphics Techniques (JCGT)*, 2(2):122–141, 2013.

[45] T. McLoughlin, M. W. Jones, R. S. Laramee, R. Malki, I. Masters, and C. D. Hansen. Similarity measures for enhancing interactive streamline seeding. *IEEE Transactions on Visualization and Computer Graphics*, 19(8):1342–1353, 2013.

[46] H. Meshkin. Sort-independent alpha blending. In *Perpetual Entertainment, GDC Session*, 2007.

[47] M. Niemann and J. Fröhlich. Direct numerical simulation of turbulent heat transfer behind a backward-facing step at low Prandtl number. *Proceedings in Applied Mathematics and Mechanics*, 14(1):659–660, 2014.

[48] S. Oeltze-Jafra, J. R. Cebral, G. Janiga, and B. Preim. Cluster analysis of vortical flow in simulations of cerebral aneurysm hemodynamics. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):757–766, Jan 2016.

[49] G. Raisbeck. The order of magnitude of the Fourier coefficients in functions having isolated singularities. *The American Mathematical Monthly*, 62(3):149–154, 1955.

[50] A. Scherzinger, T. Brix, D. Drees, A. Völker, K. Radkov, N. Santalidis, A. Fieguth, and K. H. Hinrichs. Interactive exploration of cosmological dark-matter simulation data. *IEEE Computer Graphics and Applications*, 37(2):80–89, Mar 2017.

[51] M. Schulze, J. Martinez Esturo, T. Günther, C. Rössl, H.-P. Seidel, T. Weinkauf, and H. Theisel. Sets of globally optimal stream surfaces for flow visualization. *Computer Graphics Forum (Proc. EuroVis)*, 33(3):1–10, 2014.

[52] S. W. Skillman, M. S. Warren, M. J. Turk, R. H. Wechsler, D. E. Holz, and P. M. Sutter. Dark Sky Simulations: Early Data Release. *ArXiv e-prints*, July 2014.

[53] C. Soler, K. Subr, F. Durand, N. Holzschuch, and F. Sillion. Fourier depth of field. *ACM Trans. Graph.*, 28(2):18:1–18:12, May 2009.

[54] B. Stevens. Introduction to UCLA-LES, 2013.

[55] J. Tao, J. Ma, C. Wang, and C. Shene. A unified approach to streamline selection and viewpoint selection for 3D flow visualization. *IEEE Transactions on Visualization and Computer Graphics*, 19:393–406, 2013.

[56] I. Viola and E. Gröller. Smart visibility in visualization. In *Proc. Computational Aesthetics*, pages 209–216, 2005.

[57] I. Viola, A. Kanitsar, and E. Gröller. Importance-driven volume rendering. In *Proc. Visualization*, pages 139–146, 2004.

[58] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[59] E. T. Whittaker and G. N. Watson. *A course of modern analysis*. Cambridge university press, 1996.

[60] J. Woodring, M. Petersen, A. Schmeißer, J. Patchett, J. Ahrens, and H. Hagen. In situ eddy analysis in a high-resolution ocean climate model. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):857–866, Jan 2016.

[61] L. Xu, T.-Y. Lee, and H.-W. Shen. An information-theoretic framework for flow visualization. In *IEEE Transactions on Visualization and Computer Graphics*, pages 1216–1224, 2010.

[62] J. C. Yang, J. Hensley, H. Grün, and N. Thibieroz. Real-time concurrent linked list construction on the GPU. *Computer Graphics Forum*, 29(4):1297–1304, 2010.

[63] X. Ye, D. Kao, and A. Pang. Strategy for seeding 3D streamlines. *IEEE Visualization Conference*, pages 471–478, 2005.

[64] H. Yu, C. Wang, C.-K. Shene, and J. Chen. Hierarchical streamline bundles. *IEEE Transactions on Visualization and Computer Graphics*, 18(8):1353–1367, 2012.

[65] M. Zöckler, D. Stalling, and H.-C. Hege. Interactive visualization of 3D vector fields using illuminated stream lines. In *IEEE Visualization*, pages 107–113, 1996.

Irene Baeza Rojo is a PhD student at the Computer Graphics Laboratory of ETH, Zürich. She received her B.Sc. degree in Electrical Engineering at UPC Barcelona in 2014 and her M.Sc. degree in Computer Science at ETH Zürich in 2017. Her research interests lie in computer graphics, scientific visualization, as well as photorealistic and real-time rendering.

Markus Gross is a Professor of Computer Science at ETH Zürich, head of the Computer Graphics Laboratory and director of Disney Research Zürich. His research interests include physically based modeling, computer animation, immersive displays, and video technology. He received a master of science in electrical and computer engineering and a PhD in computer graphics and image analysis, both from Saarland University in Germany in 1986 and 1989.

Tobias Günther joined the Computer Graphics Laboratory (CGL) at the ETH Zürich as a postdoctoral researcher in 2016. He received his M.Sc. in Computer Science in 2013 and his Dr.-Ing. (Ph.D.) in 2016 both from the Otto-von-Guericke University of Magdeburg. His research interests include scientific visualization, progressive light transport and real-time rendering.