Efficient Stochastic Rendering of Static and Animated Volumes using Visibility Sweeps

von Radziewsky, Philipp; Kroes, Thomas; Eisemann, Martin; Eisemann, Elmar

DOI
10.1109/TVCG.2016.2606498

Publication date
2017

Document Version
Final published version

Published in
IEEE Transactions on Visualization and Computer Graphics

Efficient Stochastic Rendering of Static and Animated Volumes Using Visibility Sweeps

Philipp von Radziewsky, Thomas Kroes, *Member, IEEE*, Martin Eisemann, and Elmar Eisemann, *Member, IEEE*

Abstract—Stochastically solving the rendering integral (particularly visibility) is the de-facto standard for physically-based light transport but it is computationally expensive, especially when displaying heterogeneous volumetric data. In this work, we present efficient techniques to speed-up the rendering process via a novel visibility-estimation method in concert with an unbiased importance sampling (involving environmental lighting and visibility inside the volume), filtering, and update techniques for both static and animated scenes. Our major contributions include a progressive estimate of partial occlusions based on a fast sweeping-plane algorithm. These occlusions are stored in an octahedral representation, which can be conveniently transformed into a quadtree-based hierarchy suited for a joint importance sampling. Further, we propose sweep-space filtering, which suppresses the occurrence of fireflies and investigate different update schemes for animated scenes. Our technique is unbiased, requires little precomputation, is highly parallelizable, and is applicable to a various volume data sets, dynamic transfer functions, animated volumes and changing environmental lighting.

Index Terms—Visibility, raytracing, volume rendering, stochastic rendering, importance sampling

1 Introduction

In stochastic volume rendering the rendering equation is evaluated using Monte Carlo (MC) techniques by taking many point-samples (shooting rays) to compute the light distribution within a volume. Starting at the camera, for each sample a ray traverses the volume until a scattering event occurs along this ray based on the current transfer function, which maps the volume's density to material properties. Evaluating the rendering equation at the position of the scattering event requires generating and traversing one or more sample rays which are ultimately either absorbed within the volume or hit a light source, e.g., the environmental light. Without incorporating knowledge about the light characteristics or volume absorption this process can be highly inefficient as many rays might contribute little or nothing to the final image.

Importance-sampling techniques [1], [2], [3] incorporate knowledge about scene content, e.g., material properties, and light distribution to improve convergence during rendering. Visibility, however, is usually not taken into account as its direct computation is almost as costly as solving the rendering integral directly. Another drawback of a brute-force visibility precomputation is that any change in the transfer function, lighting or motion within the scene would require a complete reevaluation.

The problem we tackle in this work is the evaluation of direct lighting for an (animated) volume data set in the context of an unbiased MC-based stochastic volume renderer. We support arbitrary and interactively changing transfer functions to define varying diffuse materials. The volume is lit by natural illumination in form of environmental lighting.

The main idea of our approach is to estimate a joint probability density function (pdf) combining lighting and approximate visibility information, used to steer the sampling for any scattering event within the volume, which is more efficient than in previous work and still unbiased. While generalization is possible, we illustrate the application to single scattering only.

This work is an improved and extended version of [4]. We include all aspects and components of the previous paper for the sake of completeness and point out noteworthy differences where appropriate. In addition to the contributions, which are:

- An efficient sweeping-plane algorithm to compute approximate visibility within a 3D volume;
- A product importance sampling solution based on joint environmental light and visibility information;
- A GPU-adapted and highly-parallel implementation

we extend the previous version by:

- Several implementation details;
- An efficient filtering technique to avoid rendering artifacts stemming from (potentially) insufficient sampling of the visibility domain;
- Treatment of animated volumes via three (lazy) update schemes to avoid costly visibility recomputations.

Our technique is useful for any volumetric renderer with dynamically changing content, such as environmental light, transfer functions, volume data, etc., making it an interesting addition to visualization and rendering systems aiming for unbiased results.

- *P. von Radziewsky, T. Kroes, and E. Eisemann are with Delft University of Technology, CD Delft 2628, Netherlands. E-mail: {p.vonradziewsky, t.kroes, e.eisemann}@tudelft.nl.*
- *M. Eisemann is with TH Köln, Köln 50678, Germany. E-mail: martin.eisemann@th-koeln.de.*

Manuscript received 31 Dec. 2015; revised 9 Aug. 2016; accepted 29 Aug. 2016. Date of publication 7 Sept. 2016; date of current version 2 Aug. 2017. Recommended for acceptance by X. Tong.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TVCG.2016.2606498

Fig. 1. *Overview:* We compute the product of approximated visibility and environment map lighting in a stochastic Monte Carlo volume renderer to steer a joint importance sampling of the direct lighting. Our proposed approach is well suited for dynamic changes in visibility and lighting functions due to a fast sweeping-plane algorithm to estimate visibility. The insets show how our technique (magenta) achieves faster convergence with fewer samples compared to a uniform sampling (red) and importance sampling of the environment map (yellow). Here, 4 samples per pixel have been used. The Manix data set consists of $512 \times 512 \times 460$ voxels.

2 Related Work

The literature on volumetric-illumination techniques is vast and a recent survey on this topic can be found in [5]. We will focus only on certain aspects to put our approach in perspective.

Ambient Occlusion helps to better perceive certain shapes and their relative positions by measuring the light accessibility for each scene point. In these approaches, luminance is linked to the degree of local occlusion [6]. Multi-resolution variants [7], and even dynamic ambient occlusion variants [8], which allow changes to the transfer function, have been considered. Nonetheless, ambient occlusion computes only a statistical scalar value to approximate the ambient light, which means that directional information is lost. We incorporate full directional support for high-quality unbiased physically-based rendering.

Visibility Approximation for Semi-Transparent Structures are most common in physically-based volume rendering. Opacity shadow maps [9] are an extension of shadow maps [10] using a stack that stores alpha values instead of depth values to support shadow computation for complex, potentially semi-transparent structures. Deep shadow maps [11] are a more compact representation, which store a shadow-function approximation per pixel. They have quickly been adopted for volume rendering [12], [13]. Recently, other approaches showed links to filtering [14]. They approximate occlusion in a very efficient way but at the expense of precision.

All such techniques are fast but inapplicable in our scenario of stochastic MC volume rendering. First, using approximate visibility directly for shading introduces a bias. Second, these techniques support only point and directional light sources, whereas we aim for environmental lighting. Third, visibility is costly to compute and even approximating it can usually involve many rays, although not all locations might ultimately contribute to the image. Our approach computes visibility in a coarse 5D grid and uses it only to carefully steer the sample generation. In this way, our approach remains unbiased, and supports arbitrary environmental lighting.

Basis-Function Techniques decouple light-source radiance and visibility, which allows for dynamically changing the illumination. Spherical harmonics (SH) are prominent basis functions, used for example for pre-computed radiance transfer [15], and were first used in the context of volume rendering to pre-compute and store isosurface illumination [16]. They have also been used to store visibility for volume rendering under natural illumination [17]. Other research in this area mostly aimed at generalizations to support advanced material properties [18] or reduce memory costs [19].

While SH are well suited to represent low-frequency functions, their direct use for visibility is a strong approximation and introduces bias. Further, only low-frequency illumination is supported, in contrast to our solution.

Image Plane-Sweep Volume Illumination Approaches move a virtual plane through a scene to invoke the shading computations for all positions within the plane in parallel. The parallelism makes these approaches highly applicable to modern parallel architectures, such as the GPU. Using carefully-chosen approximations (e.g., a forward-peaked phase function, single-point or directional light source), single and forward multiple-scattering effects can be simulated at interactive frame rates [20]. We decouple the plane sweep from a particular light source to enable general illumination and drive the sampling process in stochastic MC volume rendering.

Light propagation maps [21] are a plane sweep approach to solve the radiative transport equation for a volume by alternatingly propagating light along 6 sweeping planes until convergence.

Recently, iterative convolutions on volume slices have also been used to approximate direct lighting [22]. The results are approximate, some parameter settings have to be carefully chosen, and only particular light-source configurations are efficiently supported (e.g., usually Gaussian and behind the observer).

Monte Carlo Ray Tracing for volume rendering gained attention with the advances of modern GPUs, which made interactive progressive rendering possible. First attempts sacrificed generality for performance [23] and did not support translucent materials. New approaches, such as Exposure Render [24] achieve images of very high realism. They employ all the benefits of physically-based MC techniques: arbitrary

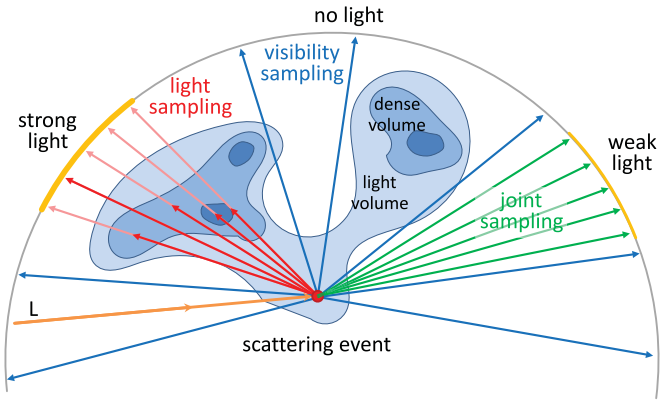

Fig. 2. *Problem statement:* For efficient sampling, samples with both strong light and strong visibility need to be found. Sampling according to the lighting only (red) may give bad results as samples may get absorbed within the volume before reaching the light source, sampling only according to the visibility (blue) might miss important lights. Product sampling (green) solves the problem. Unfortunately, the required visibility is usually unknown beforehand.

natural illumination, real-world cameras with lens and aperture (e.g., for depth-of-field effects). We implemented our approach building upon this open-source solution. Only recently, specialized algorithms have been developed to efficiently handle participating media by splitting the evaluation into an analytically and a numerically evaluated part [25].

Importance Sampling is a powerful sampling technique to render objects illuminated by natural or complex environmental illumination [1]. An efficient method for non-specular materials is to place pre-integrated directional lights at the brightest locations [26], [27], [28]. These methods work extremely well in the absence of occlusion, but shadowed regions may appear noisy. When materials are increasingly specular, a large number of lights is needed to adequately represent the environment map. Consequently, many physically-based MC techniques sample the environment map directly to avoid such artifacts. The intensity of the environment map can even be used as a pdf to steer the sampling [3].

If also visibility or material properties are to be included, the pdfs can be combined in a single MC estimator via multiple importance sampling (MIS) [29]. MIS is most efficient if only one of the sampled functions is complex and will choose accordingly. If both are complex, MIS provides little advantage and is likely to waste samples in regions with little influence. Visibility and lighting can both be complex and only a joint sampling of both functions can be efficient (Fig. 2). A first step towards this direction was taken in [30]. Their technique importance samples the environment map to produce a candidate sample. Its probability is then evaluated again using a special pdf involving the BRDF to determine if an evaluation is triggered. Such a sampling can quickly become costly, due to potentially high rejections rates (in the order of 90 percent) [30].

More related to our sampling algorithm are techniques for joint importance sampling. They compute the BRDF/environment-map product [2], [31], [32] and BRDF/visibility/environment-map product [33] to steer sample placement. In the context of participating media, joint importance sampling can also be employed to optimize volumetric paths [34]. In this article, we focus on efficient visibility/environment-map sampling. Nonetheless, we also rely on a quadtree-based product to hierarchically warp samples [32].

3 Overview

Here, we describe our approach. First, we present the background knowledge necessary for the understanding of our method (Section 4). Our actual solution is described, starting with our data structures and data representations (Section 5). All elements have been designed with GPU-efficiency in mind. Our visibility-sweep algorithm (Section 6) allows us to derive an approximate visibility within the volume. This information is then used in conjunction with the scene illumination to yield a joint sampling technique to steer the MC evaluation (Section 7). We then extend the visibility sweeps with an on-the-fly filtering function (Section 8) to prevent potential undersampling artifacts. Further, we investigate three update schemes to support efficient re-usage of visibility in animated volumes (Section 9). The benefits for convergence behavior and the support of dynamic volumes will be demonstrated in Section 10.

4 Background and Goal

We use a single-scattering rendering equation [3] implemented in the framework of [24] for isotropic media:

$$L = \int_0^\infty T_r(\mathbf{x}_c, \mathbf{x}(u)) L_o(\mathbf{x}(u)) \mathrm{d}u. \tag{1}$$

It describes the radiance L aggregated along ray positions $\mathbf{x}(u)$ towards the camera at $\mathbf{x}_c = \mathbf{x}(0)$, where

$$T_r(\mathbf{x}(t_1), \mathbf{x}(t_2)) = \exp\left(-\int_{t_1}^{t_2} \tau_e(D(\mathbf{x}(t)))\mathrm{d}t\right) \tag{2}$$

$$L_o(\mathbf{x}(u)) = \tau_\rho(D(\mathbf{x}(u))) \int_\Omega V(\mathbf{x}(u), \omega) L_i(\omega) \mathrm{d}\omega. \tag{3}$$

T_r is the *transmittance* between two points and L_o the outgoing radiance towards the camera. The negated exponent of T_r is called the *optical thickness*. The volume *density* $D(\mathbf{x}(u))$ is mapped to an *extinction coefficient* τ_e and a *scattering coefficient* τ_ρ by a transfer function. The environmental light L_i is assumed to be independent of $\mathbf{x}(u)$. Consequently, the *visibility* V of the light in direction ω from $\mathbf{x}(u)$ is the transmittance to a point at infinity:

$$V(\mathbf{x}(u), \omega) = \lim_{h \to \infty} T_r(\mathbf{x}(u), \mathbf{x}(u) + h\omega), \tag{4}$$

where the integration domain is, in practice, limited by the volume's bounding box. For brevity, we will mostly omit the ray parameter u and write only $\mathbf{x}$ to denote a certain location.

We use stochastic ray marching to solve the integral in Eqs. (1) and (2). To solve Eq. (3) stochastically, we rely on MC integration:

$$L_o(\mathbf{x}) \approx \frac{1}{N} \sum_{j=1}^{N} \frac{\tau_\rho(D(\mathbf{x})) V(\mathbf{x}, \omega_j) L_i(\omega_j)}{p(\mathbf{x}, \omega_j)}.$$

Here, p is a pdf, which is used to weigh and generate random sample vectors ω_j. The efficiency of the MC integration

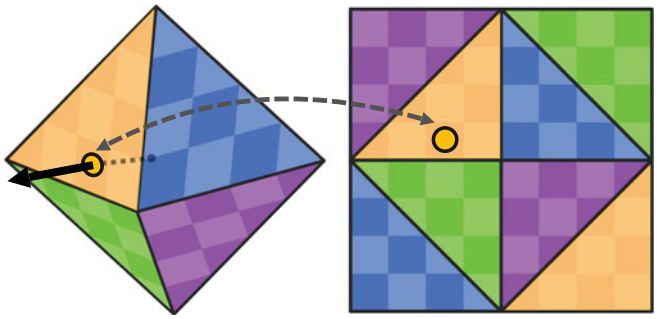

Fig. 3. *Octahedral representation:* We encode spherical functions using an octahedral representation. Left: 3D representation, right: unfolded 2D representation.

depends highly on the pdf p, as the variance and hence the convergence of the estimator depends on how closely p approximates the integrand (Fig. 2).

This paper addresses the question of how to approximate this optimal pdf efficiently. To this extent, we split p into two components p_V and p_{L_i}:

$$p(\mathbf{x},\omega) = \frac{1}{W(\mathbf{x})} p_V(\mathbf{x},\omega) p_{L_i}(\omega). \tag{5}$$

p_V is a pdf resulting from visibility information, which changes locally throughout the volume based on the location $\mathbf{x}$. p_{L_i} is a pdf based on the position-independent environmental lighting. Finally, $W(\mathbf{x}) = \int p_V(\mathbf{x},\omega) p_{L_i}(\omega) \mathrm{d}\omega$ is a normalization factor to produce a valid pdf.

p_{L_i} is known and can be derived directly from the intensity of the environmental lighting, normalized by its overall intensity. The representation of these functions, the computation and update of $p_V(\mathbf{x},\omega)$ and $p(\mathbf{x},\omega)$, and how to draw samples from $p(\mathbf{x},\omega)$ are the core of our method and explained in the following sections. We describe the data structures, then the visibility approximation, which will be used to derive p_V, before combining all the elements. Then we will extend the basic approach and adapt p_V to prevent visually disturbing high-energy samples and investigate three update schemes for p_V for animated volumes.

5 Octahedral Representation

Before explaining the algorithmic part of our approach, we will focus on the chosen data structures. All representations were developed with efficiency on modern graphics hardware in mind. We opt for simplifying generation, sampling, and product computation, which will be necessary to drive the MC sampling process.

As we are dealing with potentially semi-transparent media in volume rendering, we will assume for the moment that V is locally smooth with respect to $\mathbf{x}$ and ω. In consequence, it can be considered sufficient to estimate V at discrete positions $\mathbf{x}_d$ and using a few discrete directions ω_d.

We represent visibility by discrete values arranged on a regular five-dimensional grid, the *visibility grid* $\mathbf{V}$. The first two dimensions encode directions, the remaining ones spatial positions. These positions encompass the density volume and the values stored at the grid centers are interpolated during rendering to obtain the visibility estimates at each location within the volume. Each entry represents the average probability to hit the environmental light source for all rays originating in the cube in physical space represented by the spatial component in the visbility grid and whose directional components are in the corresponding interval.

In our approach, we internally save several 3D textures to store the visibility. Each texture saves the visibility for a single direction ω_d throughout the 3D volume. Each pass of our sweeping-plane algorithm (Section 6) updates one of these textures. When sampling a direction during a scattering event (Section 7) we opt for an octahedral representation generated on the fly, which is a discrete image-based area-preserving representation [35] and can be saved/accessed as a 2D texture (Fig. 3). Each texel in this map is associated with one direction ω_d and indicates the accumulated volumetric visibility in direction ω_d from the respective grid cell's location.

6 Visibility Approximation

In this section, we describe how to compute the entries of the visibility grid via our sweeping-plane algorithm. Visibility is computed for one direction ω_d at a time. In each step, one slice of $\mathbf{V}$ is evaluated in parallel, reusing results from the previous slice. This insight makes it possible to only use a few value lookups per slice, instead of accumulating visibility along a ray throughout the whole volume. In consequence, the amortized cost over all grid cells is comparatively low. After all directions have been treated, the resulting $\mathbf{V}$ is used to derive the pdf p_V, which will guide the MC sampling process. An illustration of the entire process is given in Fig. 4. Hereafter, we describe the algorithm for a fixed direction ω_d.

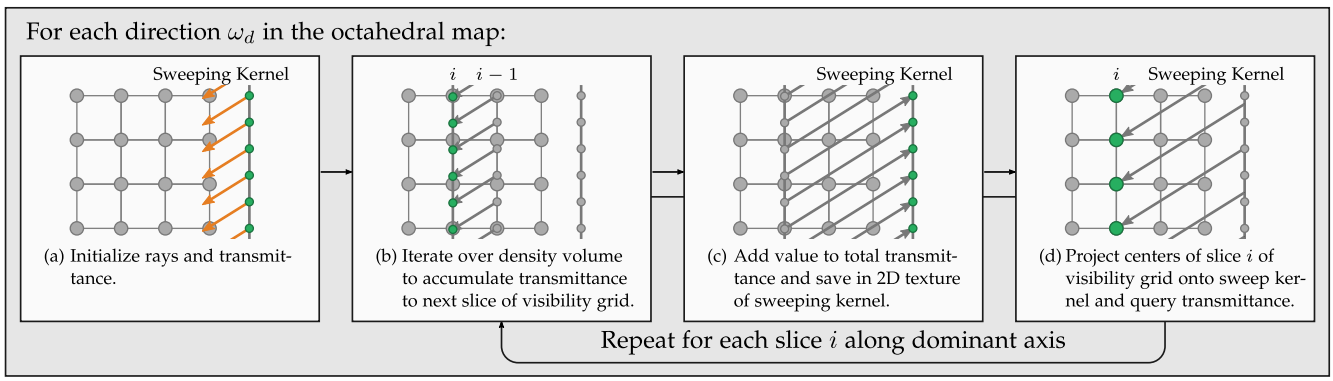

Fig. 4. *Precomputation algorithm overview:* We compute the transmittance along sample rays starting at an axis-aligned plane outside of the density volume. The accumulation of transmittance reuses results needed at the previous slice which are cached in the sweeping kernel. Using a 3D texture to store the respective part of the visibility grid for each direction ω allows us to exploit memory locality on the GPU in this step.

In the following, we let $\mathbf{x}_{min}, \mathbf{x}_{max} \in \mathbb{R}^3$ be the 6 values that describe $\mathbf{V}$'s bounding box to ease notation. First, let a plane $\mathcal{P}_d$ be given by normal

$$\mathbf{n} = \underset{\tilde{\mathbf{n}} \in \{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}}{\arg\max} |\omega_d^T \tilde{\mathbf{n}}|, \tag{6}$$

where $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ is the standard basis, and distance from origin

$$a = \begin{cases} \mathbf{n}^T \mathbf{x}_{min} & \text{if } \mathbf{n}^T \omega_d < 0, \\ \%\mathbf{n}^T \mathbf{x}_{max} & \text{otherwise.} \end{cases} \tag{7}$$

Starting from this plane, we accumulate transmittance along rays whose origins are arranged on a regular 2D grid within $\mathcal{P}_d$. This grid is associated with a 2D texture, which stores intermediate values during the sweeping-plane algorithm and we'll refer to it as the *sweeping kernel*. It is noteworthy that visibility grid, input volume, and sweeping kernel do not necessarily share the same resolution along corresponding axes. It is sensible to choose the resolution of the sweeping kernel to be similar to the resolution of the input voxel grid to ensure not to miss small features of the volume. In our experiments, we choose it to be $s \times s$, with $s = \max\{r_x, r_y, r_z\}$, where r_x, r_y, r_z are the number of voxels in the input grid along the x, y, z axes. However, doing the same for the five-dimensional visibility grid may lead to prohibitive memory requirements, which is why we typically choose a lower resolution for it.

To coordinate ray traversal, we introduce a sweeping plane $\mathcal{S}_d$ parallel to $\mathcal{P}_d$. The main loop moves $\mathcal{S}_d$ along $\mathbf{n}$ by one visibility-map slice in $\mathbf{V}$ at a time until the entire volume is traversed. The traversal is done reversely to ω_d in positive ($\mathbf{n}^T \omega_d \leq 0$) or negative direction (otherwise). At iteration i, $\mathcal{S}_d$ is placed such that the centers of slice i (or $r-1-i$, resp.) coincide with $\mathcal{S}_d$. In one iteration, we only need to compute a change from the previous slice by

$$V(\mathbf{x}(t_i), \omega_d) = V(\mathbf{x}(t_{i-1}), \omega_d) \cdot T_r(\mathbf{x}(t_{i-1}), \mathbf{x}(t_i)), \tag{8}$$

where $t_i = t_{end} \cdot \frac{i+1/2}{r}$, with r the resolution of $\mathbf{V}$ along the sweeping axis and $t_{end} = \mathbf{n}^T(\mathbf{x}_{max} - \mathbf{x}_{min})/|\mathbf{n}^T \omega_d|$. The value of $T_r(\mathbf{x}(t_{i-1}), \mathbf{x}(t_i))$ is computed via ray marching on the input volume; then, we update the visibility according to Eq. (8) and store it in the sweeping kernel. As the final step of an iteration, we project the visibility grid centers of slice i onto $\mathcal{P}_d$ and look up the visibility values via bilinear interpolation of the neighboring grid cells in the sweeping kernel. We will refine this step in Section 8 to account for subsampled visibility grids.

After the algorithm finishes and all directions have been processed, $\mathbf{V}$ encodes a discrete approximation of the visibility within the volume, which, if normalized, results in the pdf p_V. We add a small $\epsilon = 0.01$ beforehand to prevent zero probabilities.

The iterative update is significantly more efficient than individual visibility computations per visibility grid cell. Additionally, one can implement a further optimization which exploits that the visibility is multiplicatively accumulated over $[0, t_{end}]$. The accumulation towards the opposite direction $-\omega_d$ can be computed from the final entries in the sweeping kernel and the values in the visibility grid. Specifically, given a ray $\mathbf{y}(.)$ with $\mathbf{y}(0) = \mathbf{x}(t_{end})$, $\mathbf{y}(t_{end}) = \mathbf{x}(0)$, the visibility along $\mathbf{y}$ to slice $r-1-i$ is

$$\begin{aligned} V(\mathbf{y}(t_{r-1-i}), -\omega_d) &= T_r(\mathbf{y}(0), \mathbf{y}(t_{r-1-i})) \\ &= T_r(\mathbf{x}(t_i), \mathbf{x}(t_{end})) \\ &= \frac{T_r(\mathbf{x}(0), \mathbf{x}(t_{end}))}{V(\mathbf{x}(t_i), \omega_d)}, \end{aligned}$$

where $T_r(\mathbf{x}(0), \mathbf{x}(t_{end}))$ is just the value stored in the sweeping kernel at the end of a sweep. However, due to filtering issues discussed in Section 8, we compute visibility for the reversed directions explicitly using the original sweeping-plane algorithm explained earlier.

7 Joint Importance Sampling

At a scatter event during rendering, we want to make use of a joint importance sampling combining visibility and environmental lighting. We have explained how to produce the pdfs for p_V and p_{L_i}. Here, we explain how to combine both. The computation is divided into a preprocess, taking place whenever the environment map, the data, or the transfer function changes, and an online process, taking place whenever a scattering event occurs during rendering.

Preprocess. For the preprocess, we assume that the environment map is given as an octahedral map, otherwise we convert it first. As a reminder, p_{L_i} is defined as the normalized intensity value of the environmental lighting, giving higher importance to brighter parts. Being derived directly from the environment map, the resolution of the octahedral map of p_{L_i} is usually higher than for p_V. To combine both, we first adapt the resolution of p_{L_i}. To simplify explanations, we assume that the resolution in width and height is chosen to be a power of two.

Similar to [32], we create a multiresolution pdf from p_{L_i} in form of a quadtree, i.e., each node saves the average of its four child nodes, with the leafs being the individual pixels. To match the resolution between lighting and visibility, we choose a level l in p_{L_i} whose resolution is equal to the angular resolution of a single spatial position within the visibility grid We then multiply all entries in $\mathbf{V}$ with the respective information in p_{L_i} at level l. The result is an unnormalized joint pdf of the combined visibility/lighting product.

Rendering. In the rendering phase, we create a final combined pdf p for each scatter event at location $\mathbf{x}$ on the fly. This pdf is then used to draw a single sample, as this strategy is often more efficient than drawing multiple samples at once in a stochastic volume renderer with semi-transparent media [24]. Nonetheless, the sampling algorithm naturally extends to any number and distribution of initial samples, including quasi-MC methods [36].

To derive the pdf p, we first linearly interpolate the neighboring visibility grid cells. After our preprocess, these carry the information of visibility and lighting. Initially, the interpolated result is not a pdf. Nevertheless, we do not normalize it right away, but compute a multiresolution representation in the form of a quadtree, where each node is the average of its child nodes. Following the hierarchical warping technique [2], we can then transform a uniformly distributed $[0,1)^2$-variable into one that is distributed according to p by passing the sample down in the quadtree

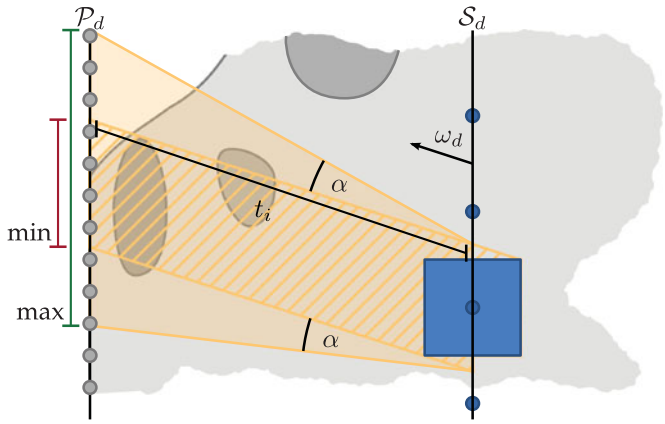

Fig. 5. *Calculation of the filter size* Hatched Area: We account for the oblique projection when decimating the sampling rate before transferring the values to the visibility grid slice. Yellow Area: We widen the filter by a maximal amount that depends on the opening angle α and distance t_i along ω_d.

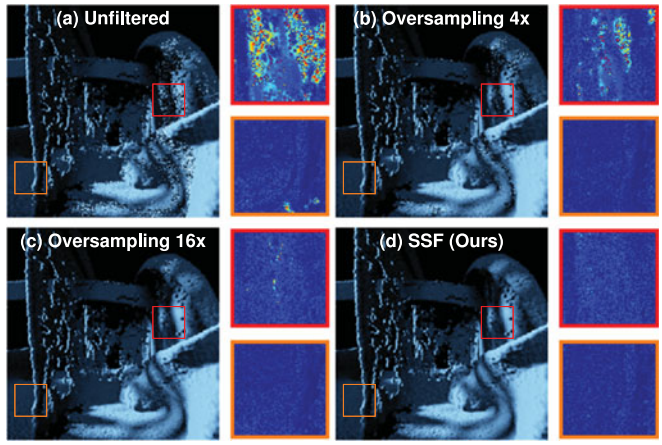

Fig. 6. *Sweep-space filtering*: Comparison of (a) the unfiltered result from [4], (b) $4\times$, and (c) $16\times$ oversampling in the angular domain, and (d) our filtering algorithm. The insets visualize the absolute differences to the reference image. For all methods we use 64 samples per pixel (8,192 SPP for the reference).

following the local probabilities. In contrast to [2], we need to normalize each 2×2 tile that we encounter during the quadtree sampling to determine the actual probabilities. Nonetheless, as we only draw a single sample per scatter event, the effort is only $O(\log n)$, where n is the number of texels in the lowest level of the quadtree. In comparison, it would be $O(n)$, if we created a complete pdf for the interpolated octahedral map.

In case the environment map has a high resolution, we propose an additional *two-step approach*, which continues to descent in the remaining quadtree of the higher-resolved environment map [31], [32]. This step is especially beneficial in the presence of complex high-frequency illumination, which is otherwise not well taken into account during the sampling.

8 Sweep-Space Filtering

The variance of the estimator for L_o sampled according to p from Eq. (5) depends on how closely the reconstructed visibility resembles the original one. Undersampling may increase variance as it may introduce aliasing which will lead to higher deviation from exact visibility. Sampling the visibility at a sufficiently high rate is difficult, not only because the visibility function is high-dimensional, but also because the projection of the data volume to the visibility grid's spatial sample positions make the choice of an appropriate angular grid resolution difficult.

Even though this can be alleviated, storing this visibility would lead to impractical memory requirements for high-resolution density grids. Likewise, standard oversampling in the angular and/or spatial domain scales precomputation time by the number of samples per grid cell. The sweeping-plane algorithm described in Section 6 suggests a more efficient approach which relocates the decimation to the iterations of the sweeping stage. It assumes coherency of visibility in both the angular and spatial domain and, particularly, that coherency is still significant if both position and angle are slightly varied.

As a first step, we apply an anti-aliasing filter along the visibility values stored in the sweeping kernel for the current iterate i before projecting them onto the visibility grid cells, i.e., it is applied between steps (3) and (4) in Fig. 4 to account for the 2D resampling when mapping from the sweeping plane to a visibility grid slice. We determine the filter size by computing a bounding rectangle of a 3D visibility grid cell that is projected along ω_d onto the sweep plane. A schematic view is presented in Fig. 5.

Following the coherency assumption, we then widen this filter along the axes of the sweeping plane to approximate the visibility along further rays in the beams corresponding to the grid cells. We account here for two quantities. The first is the area covered on the unit sphere by a mapped octahedral grid cell. We describe it by an opening angle α of the beam corresponding to one cell of the visibility grid. We assume α to be constant for simplicity, even though it slightly varies due to anisotropy of the octahedral representation. With an 8×8 angular resolution, α does not exceed $\pi/8 = 11.25°$. The second quantity is the value t_i (cf. Eq. (8)) which is the distance from slice i of the visibility grid to $\mathcal{P}_d$ along ω_d after which the transmittance does not change anymore. We compute the extension of the filter by

$$t_i \cdot \tan(\beta), \tag{9}$$

where $0 \leq \beta \leq \alpha$. A value of $\beta = \alpha/2$ was empirically found to yield good results. Estimating an optimal value for β is left for future work as it always results in a trade-off between precision and recall. Filtering with a box filter across this domain can be efficiently implemented on the GPU using a sliding window as its size is constant for all positions across the filtered slice of the visibility grid and is separable along its axes. We copy the visibility values to a second texture once per iteration before applying this filter.

By sampling the visibility at a higher resolution than the visibility grid's one and applying a lowpass filter we reduce the number of spurious minima which can locally improve convergence. A comparison to angular oversampling is given in Fig. 6. The depicted error insets show that the filtering, though approximate, produces less pronounced errors in the final image, even when compared to a $16\times$ oversampling. In Fig. 6a one can also see that undersampling leads to disturbing visual artifacts.

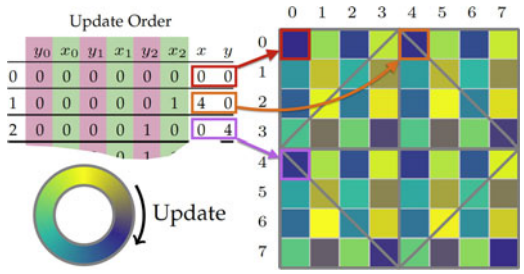

Fig. 7. *Round-robin lazy update:* Calculation of the update pattern for the round-robin lazy update algorithm. For a $2^n \times 2^n$ octahedral map (right), we consider the $2n$-bit binary numbers (upper left). The x coordinates are composed of the odd bits, the y coordinates of the even ones in reversed order. The obtained order of directions is cyclically traversed as the animation progresses (cf. Fig. 3).

9 Lazy Update for Animated Volumes

For static scenes the computation time required for the sweeping-plane algorithm is negligible, but seems unnecessarily high for dynamic data, which often exhibits unexploited coherence. Furthermore, in certain scenarios, such as simulations, it is not possible to precompute visibility and a direct evaluation in each frame would be required. In the following, we will investigate different sampling options for situations where data changes are not known in advance but exhibit temporal coherency.

Under the assumption that the computed approximate visibility does not change erratically from frame to frame, we want to postpone the recomputation of the importance function p for certain parts of the visibility map to reduce its computational cost per frame. In other words, we decouple the update frequency from the rendering frame rate. As shown for the static case, the massive parallelism inherent in the sweeping-plane algorithm, is crucial for the preprocess. Thus, we focus on strategies that determine subsets of directions ω_d to be updated per frame.

For animated data, the transmittance, visibility, volume density and environmental light become time-dependent quantities. Consequently, the joint probability $p = p(\mathbf{x}, \omega, t)$ is also time-dependent. Since p steers the rendering, it is a straight-forward assumption that update directions ω_i for p's discrete approximation should be chosen based on its change with respect to time t averaged over all positions $\mathbf{x}$ within the volume. Computing the changes is, unfortunately, as costly as computing visibility itself, which is why we opt for simpler ways to update p_V. In the following, we describe the various algorithms we tested in detail. The evaluation and comparison of these algorithms will be given in Section 10.

Round-Robin Update. One promising approach for animation is a round-robin scheme. Inspired by low-discrepancy sequences, we establish an update ordering for this approach by indexing each visibility direction in the octahedral visibility representation. The respective direction is derived from the binary representation of the update index (Fig. 7). Extracting and reversing the order of the even and odd bits results in the x- and y-position, respectively. Essentially, this is an inversion of the morton code [37] that changes the most significant bits first instead of last. This assures a well distributed sampling order over the octahedral domain.

Environment-Guided Update. While round-robin schemes ensure a good distribution over time, they do not directly exploit the changes in the scene. Because volume changes are not easy to analyze, we decided to investigate if the environment map can be better exploited. Our motivation was that visibility towards strong light sources should not be underestimated, as it can lead to fireflies, which are difficult to remove. Consequently, we choose the update directions by sampling the normalized intensity values of the environmental lighting p_L directly, using the appropriate level whose resolution is equal to the directional resolution in our visibility map.

Static Update: As a reference, we also implemented a static update scheme, which simply updates p_V for all directions in uniform intervals of n frames and uses this p_V to render the following n frames.

10 Results

We integrated our approach into a stochastic CUDA-based volume renderer [24]. We tested our approach with two different PCs, a 64bit Intel© Core i7 3820 with 3.60 GHz, 32 GB of RAM, and an NVIDIA GeForce GTX Titan as the standard machine for generating the images and experiments, and a 64bit Intel © Core i7 920 with 2.67 GHz, 12 GB of RAM, and an NVIDIA GeForce GTX 760. We compare performance and present a qualitative comparison between existing solutions and our approach. Further, we compute the mean-squared error (MSE) and compare to reference solutions using 8,192 samples per pixel with uniform sampling. Our environment maps all had a resolution of 2048×2048 pixels (in the octahedral representation). For all tests, the joint importance pdf p was constructed on the fly for each scattering event via interpolation of the eight neighboring spatial grid cells. As each visibility direction ω is represented as a single 3D texture, we can efficiently exploit hardware interpolation to create each entry in p from its eight neighbors with a single texture lookup.

We will first show results for the basic algorithm described in Sections 6 to 7 and then show the improvements achieved by our sweep-space filtering (Section 8), before discussing animated scenes (Section 9).

Timings and Parameters. The overhead during the rendering phase using our visibility sweeps is low compared to the gain in quality, especially as the sweeping-plane algorithm to update the visibility in $\mathbf{V}$ does not need to be invoked if only the viewing direction changes. As standard parameters, we use a 8^2 directional map for each visibility position and use one visibility position for each 4^3 voxels in the original data volume. Using these parameters the memory requirements of the visibility grid are equal to the original data volume. The overhead during rendering is only around 10 percent, compared to rendering the same number of samples per pixel using plain uniform sampling. This includes the interpolation, creating the multiresolution 2D pdf representation on the fly and the joint importance sampling itself. Please note that neither the filtering extension nor the lazy updates affect render time as they solely act in the preprocessing phase.

We compared our visibility sweeps approach to a brute-force computation of the visibility, where all directional

TABLE 1
Memory Requirements (MB) and Timings (Seconds) For the Sweeping-Plane Algorithm and Varying Input Parameters in Comparison to a Brute-Force Visibility Computation

Vis. Positions	$256^2 \times 230$	$128^2 \times 115$	$64^2 \times 57$	$32^2 \times 28$
Memory	964.7	120.6	15.0	1.8
Sweep (16^2)	3.76	1.93	0.76	0.53
Sweep (32^2)	3.89	1.97	0.79	0.54
Sweep (64^2)	4.05	2.03	0.79	0.55
Sweep (128^2)	4.31	2.40	1.04	0.59
Sweep (256^2)	6.36	3.23	1.63	1.41
Brute-Force	97.35	15.17	2.79	0.57

We shoot 16^2, 32^2, 64^2, 128^2 and 256^2 absorption rays per sweeping direction. All experiments are performed on the Manix data set ($512 \times 512 \times 460$ voxels) on an NVIDIA GeForce GTX 760. The variance of recorded timings of the CUDA implementation can be quite high due to scheduling with other render tasks of the OS. Hence, all depicted timings are the best of 10 runs.

entries for every discrete spatial position in $\mathbf{V}$ are computed exactly using ray marching. Table 1 shows a comparison of the timings for different parameters. For the aforementioned standard parameters and a reasonable number of absorption rays our approach is approximately $6\times$ faster than the brute-force computation. It is important to note that this factor becomes larger with an increasing number of evaluated visibility positions within the volume (up to a factor of 15 in our tests in Table 1). Further, the test scene (Manix) resembles an isosurface. It has a very steep transfer function, which means that the brute-force ray marching can stop as soon as a ray hits the isosurface. Our sweeping-plane algorithm needs to traverse the whole volume. So, we deliberately chose a difficult scenario—the benefit will be even bigger for more transparent volumes.

Filtering. While sweep-space filtering is highly beneficial for scenes with strong visibility discontinuities (Fig. 6), we still need to verify that our approach does not deteriorate convergence in scenes with low-frequency visibility. To this end, we choose a suitable scene and compare the filtered results with importance sampling of the environment map and the unfiltered two-step approach in Fig. 8. Even in

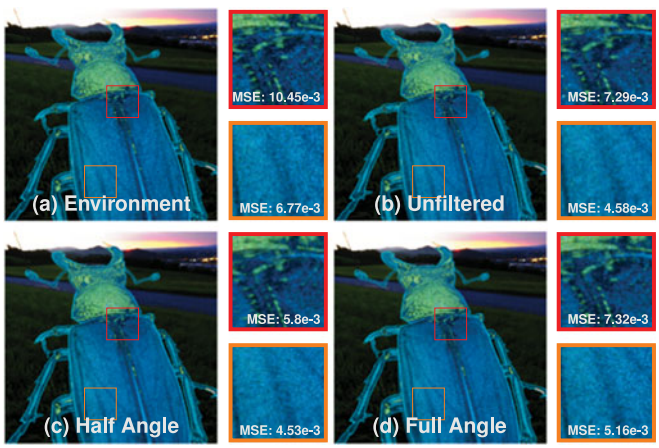

Fig. 8. *Influence of the sweep-space filter to regions with homogeneous visibility:* In addition to a result from environment map only (a) and the unfiltered result from [4] (b), we compare the filtered result in which we used the full angle α (d) to one of half the magnitude (c). All images have been created from 32 SPP.

TABLE 2
Timings in Seconds for the Sweeping-Plane Approach without Filtering: In Parentheses we List Overhead Factors of Our Sweep-Space Filtering for 256^2, 512^2, and $1,024^2$ Rays Per Sweeping Direction

	Resolution	256^2	512^2	$1,024^2$
Engine	$256^2 \times 128$	0.63 (2.05×)	0.98 (1.28×)	2.33 (0.55×)
Fluid	256^3	0.60 (2.36×)	1.03 (1.42×)	2.72 (0.52×)
Smoke	$384^2 \times 512$	1.05 (2.26×)	1.75 (1.40×)	4.56 (0.57×)
Backpack	$512^2 \times 373$	1.09 (3.08×)	1.81 (1.88×)	4.96 (0.81×)
Statue	$535 \times 342 \times 690$	1.29 (3.24×)	2.05 (2.17×)	5.47 (0.85×)
Manix	$512^2 \times 460$	1.46 (2.42×)	2.21 (1.66×)	5.33 (0.81×)

All depicted timings are the best of 10 runs on an NVIDIA GeForce GTX Titan.

particularly uniform regions, where we expected the benefit of filtering to be minor, the full-size filter is roughly on par with the two-step approach in terms of MSE. Further, recall that the filter approach approximates the mean visibility along a frustum by averaging visibility along parallel rays. We, therefore, additionally compare to a generated image for which the sweeping plane is filtered assuming half of the angle α, i.e., $\beta = \alpha/2$. We found this value to deliver the best tradeoff in our experiments and use it as the standard value in all experiments.

Our filtering approach is significantly faster than naïve oversampling. Table 2 lists runtimes of the basic approach along with overhead factors for filtering for various data sets. We used one visibility position for each 8^3 subset of the respective data volumes. Sweep-space filtering (SSF) is faster than $4\times$ angular oversampling, which yields an overhead factor of $3\times$, for most data and a sweep plane resolution of 256^2. The ratio improves with finer resolution of the sweep plane. We found the influence of the filter size insignificant due to the separable sliding window implementation. Note that oversampling and filtering can be mixed in a natural way.

Qualitative Evaluation for Static Scenes. In a static context, we compare our approach to uniform sampling (Uniform), importance sampling of the environment map only (Environment), importance sampling of the visibility only (Visibility), a combined approach, where the visibility pdf is

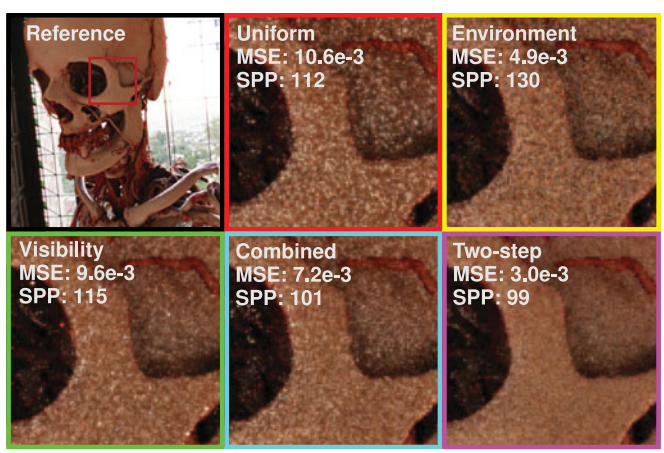

Fig. 9. *Equal time comparison:* All images, except the reference image, have been created using 10 seconds of rendering time on an NVIDIA GeForce GTX 760.

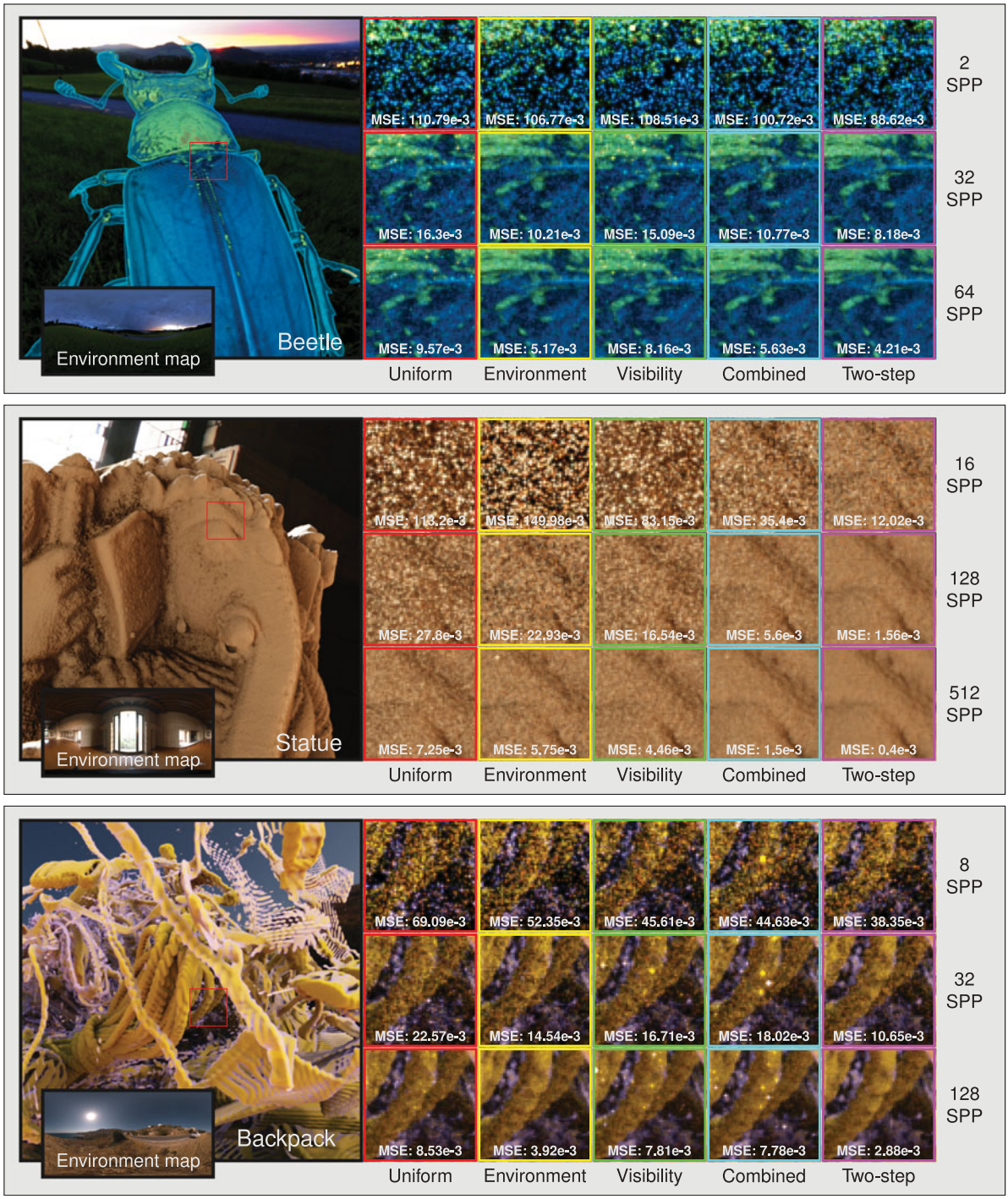

Fig. 10. *Equal sample comparison:* We compare our proposed two-step importance sampling technique (magenta) using different sample sizes to uniform sampling (red) and importance sampling of the environment map only (yellow), the visibility only (green), and the combined low-resolution product (cyan). All images are unbiased and a reference, as well as the environment map are shown on the left.

multiplied with the downsampled pdf from the environment map (Combined), as well as our combined two-step approach, which makes use of the combined sampling but switches to the full environment-map resolution as soon as a leaf in the combined pdf representation is reached (Two-step). In Fig. 9 we investigate the benefits of the basic approach [4]. For all the remaining test scenes sweep-space filtering is additionally enabled.

Fig. 9 shows an equal-time comparison of all the techniques after 10s render time, excluding the visibility precomputation. For comparison, we show the computed number of samples per pixel (SPP) and the Mean-Squared Error (MSE) for each approach. Though the number of samples is lower, due to the computational overhead induced by the joint sampling, the noise is significantly reduced with our approach. Due to a lower ray coherency the uniform sampling creates fewer samples per pixel in the same time than most of the other approaches.

Results for static scenes with an equal sample count are shown in Fig. 1 and 10. In all cases the proposed Two-step approach performs best.

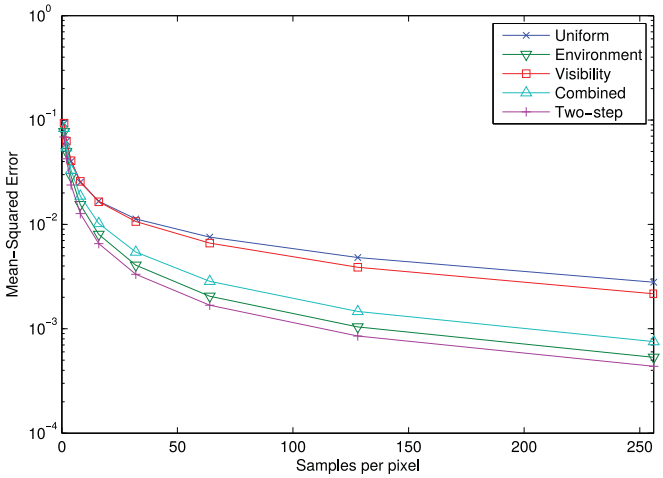

Fig. 11. Convergence graphs for the Beetle scene (Fig. 10).

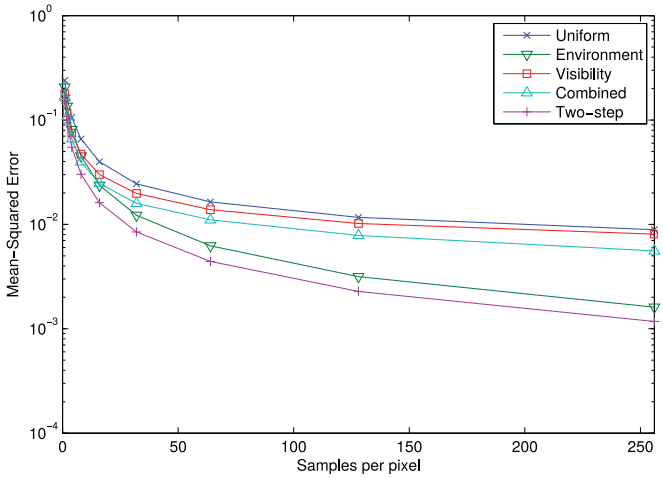

Fig. 13. Convergence graphs for the Backpack scene (Fig. 10).

Additionally, we provide error plots for the Beetle, Statue, and Backpack scenes with respect to the number of samples in Figs. 11, 12 and 13. The y-axis in all plots is in log scale. As expected, uniform sampling performs worst. Environment map and uniform sampling perform almost equally well on the Statue scene. Presumably, environmental importance sampling wastes a lot of samples that are absorbed within the volume. The combined sampling approach suffers to some extent from the low resolution of the visibility function and, therefore, the combined pdf is not able to capture the high frequency details in the environment map. This disadvantage is compensated by the two-step approach, which can make use of both the visibility and the high-frequency illumination information and shows better convergence rates even at high sampling rates. The results suggest that it is highly beneficial to incorporate the proposed visibility sweeps and joint sampling in the two-step approach for stochastic MC volume rendering.

Qualitative Evaluation for Dynamic Scenes. We tested the lazy update strategies on two four-second-long animated sequences at ~ 25 frames per second. The Smoke sequence (Fig. 14 top) has homogeneous visibility, while the Fluid sequence (Fig. 14 bottom) is one with discontinuous visibility. For environment-guided and round-robin update, we update four directions per frame, and for the static update we recompute the full visibility grid after every 16 frames. Hereby, we reduce the number of total directional updates by a factor of 16 in all cases. The update cost per frame settles on 100-150 and 150-200 ms in all our test settings for the Fluid and Smoke animations, respectively.

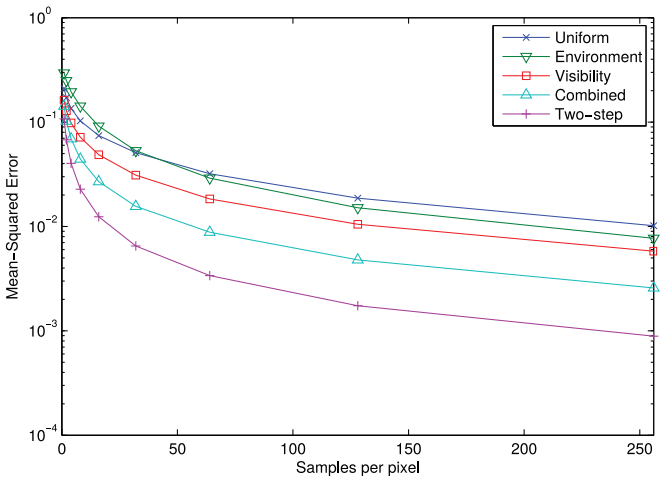

Fig. 12. Convergence graphs for the Statue scene (Fig. 10).

In Fig. 15, we provide equal-time comparisons which are adjusted to include render and update time for (a) the full animation, as well as the (b) first and (c) last 20 frames of each of the two. The update times for the static update are incorporated by averaging them over the frames from the current to the next visibility recomputation. As a reference and neglecting the computational overhead, we provide a *Full Update* plot, for which the full visibility grid is recomputed in each frame. Additionally, we also show standard environment importance sampling executed during the rendering phase, avoiding any preprocessing. Nevertheless, for all tested strategies the update cost already amortizes compared to environment importance sampling for a time budget of 250 ms.

Interestingly, the environment-guided update performs much worse than the other two tested strategies. The reason is that certain directions are almost never sampled when using a non-deterministic strategy. While the static update performs on average as well as the round-robin strategy, the error increases over the course of the animation, the more frames lie between the current frame and the last update. For the Smoke sequence in Fig. 14, the last update of the visibility data in Frame 62 (resp. 77) was 14 (resp. 13) frames ago. Here, the approach performs much worse than on the other two depicted frames. In contrast, the round-robin approach does not suffer from temporal artifacts. Further, in a practical scenario, the static approach would show a very inconsistent performance. Additionally, the two-step approach ensures that the round-robin scheme still integrates the environment map information very successfully, which makes it the most robust approach in our tests.

Overall, the results demonstrate that the stategies for exploiting the temporal coherence in animated data can easily be incorporated into the framework and that they are an attractive choice for streamed data.

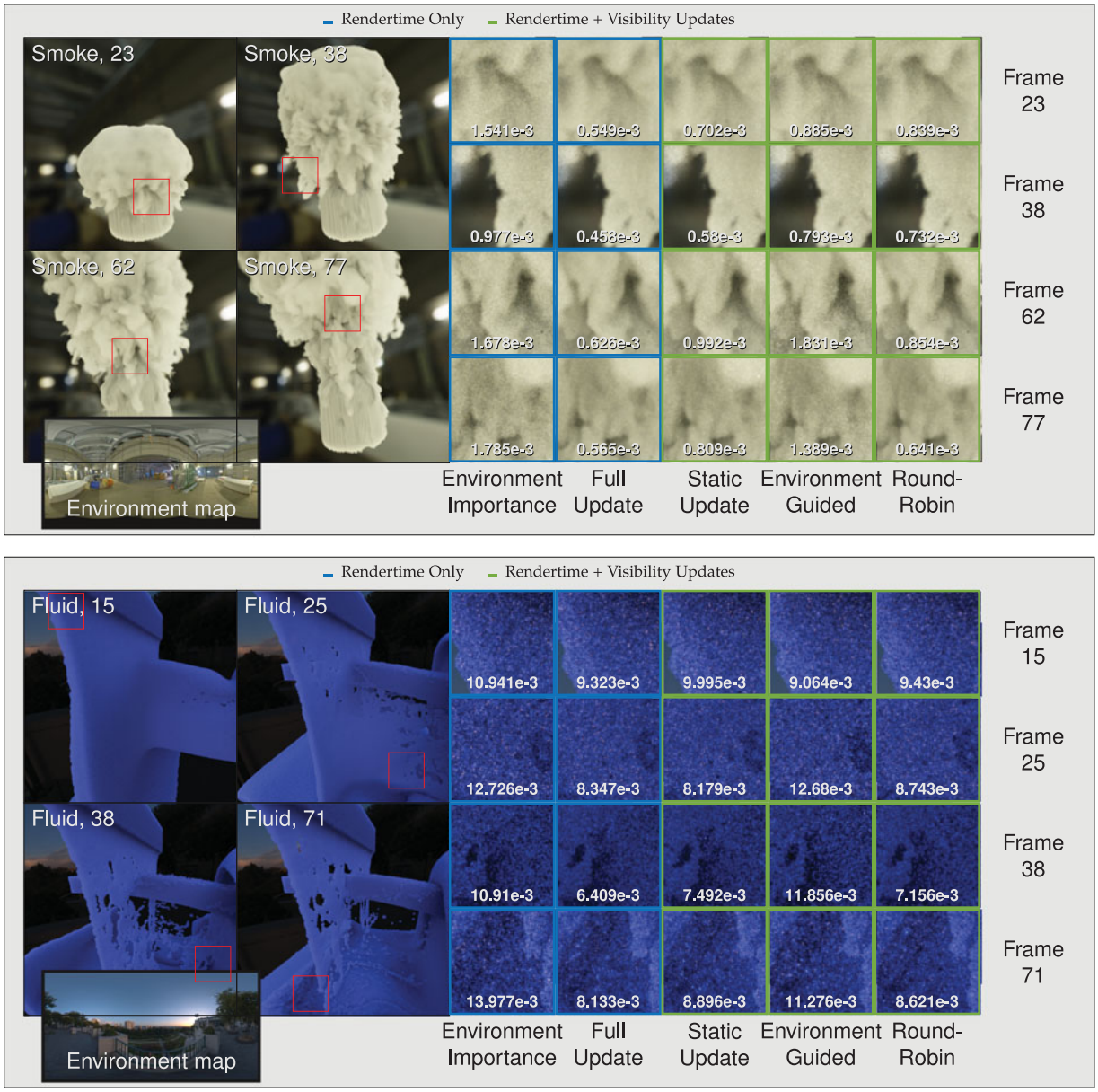

Fig. 14. *Adjusted equal time comparison of animation:* We compare our evaluated lazy update strategies (green) to importance sampling of the environment map only (first column). The *Full Update* column acts as a reference for the efficiency of the lazy updates; the full visibility grid is recomputed, but the computational effort neglected. All insets except environment importance sampling use the two-step approach in the render phase and a time budget of 4s (Smoke) / 0.5s (Fluid) on an NVIDIA GeForce GTX Titan.

11 Conclusion

We presented a joint sampling approach relying on visibility and lighting information within an interactive unbiased stochastic volume renderer. The core of our solution is an efficiently-computed visibility approximation based on a sweeping-plane algorithm. Its performance allows us to change environmental lighting and transfer functions dynamically. We carefully designed our algorithms for GPU execution and have demonstrated its applicability to different volume data sets, including animated representations.

Further, we found that visibility sweeps are even beneficial for traditional boundary representations, resulting from steep transfer functions. Our approach usually significantly lowers the amount of necessary samples when compared to previous solutions at equal quality. This result is important as sample evaluations are a very costly element in most production and rendering contexts.

Though not yet implemented, interactive clipping (slicing) of the volume is naturally supported in our approach, as it simply requires disregarding the intensity values in front of the slicing plane during the visibility computation. A remaining challenge is the extension to support also the triple product of lighting, visibility and phase-function. An interesting direction could be to investigate the usage of models such as the Henyey-Greenstein function [38], which can be efficiently integrated across solid angles [39].

The sparse visibility map proposed in this article could also be seen as an undersampled light field [40]. This would open up the possibility of using frequency analysis to derive appropriate filter kernels to reduce aliasing.

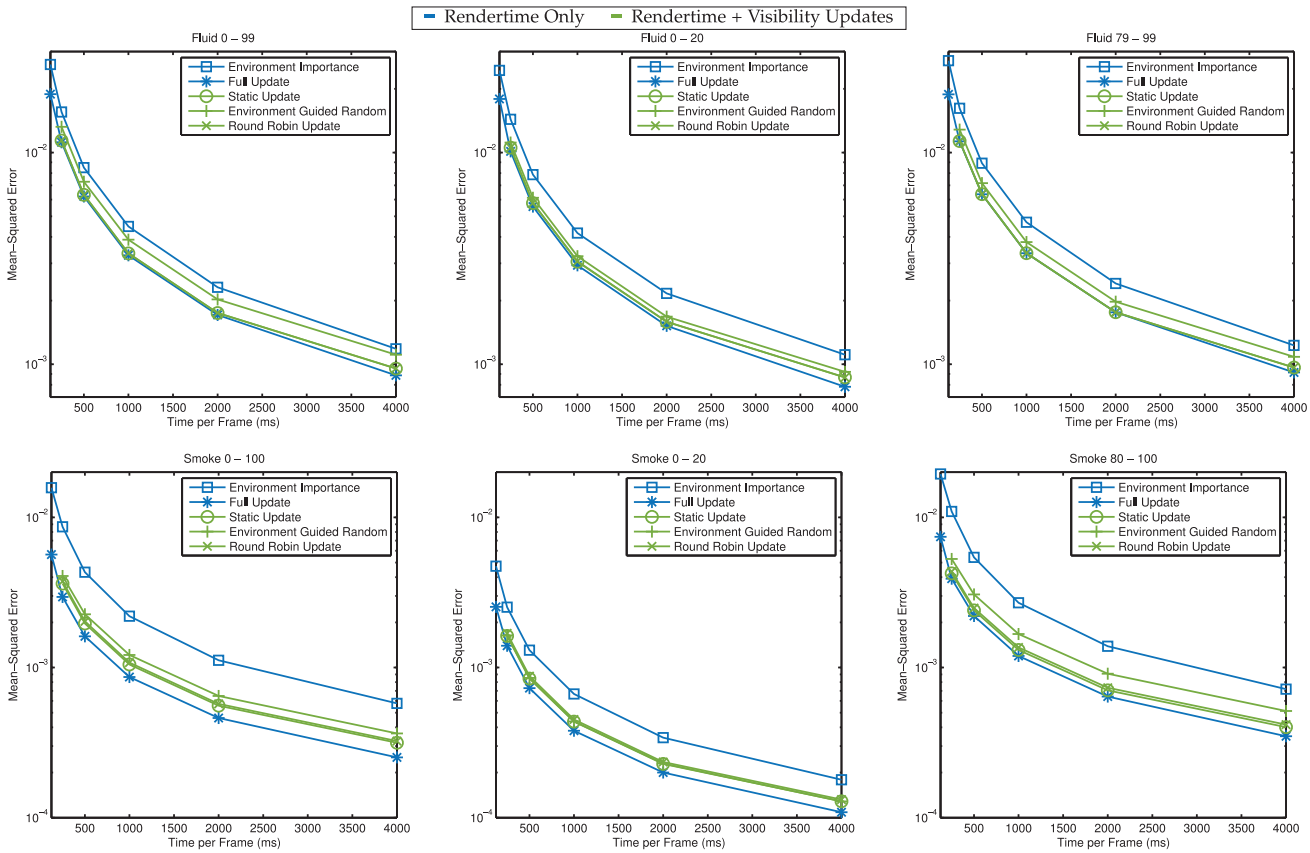

Fig. 15. *Adjusted equal time comparison for the animated Fluid and Smoke sequences (Fig. 14) for different frame intervals:* The additional update of the visibility grid is quickly amortized by the tested update methods (green). The first computation of visibility is not included, which accounts for about 1 percent of the total computation time for ≈ 2 seconds per frame in these 100 (101) frame sequence.

ACKNOWLEDGMENTS

This work was partially supported by the FP7 European Project Harvest4D and the IVCI at Saarland University. We thank Pablo Bauszat for sharing the optimized box filter code and Niels de Hoon for providing us with the Fluid sequence.

REFERENCES

[1] P. Debevec, "Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography," in *Proc. 25th Annu. Conf. Comput. Graph. Interactive Techn.*, 1998, pp. 189–198.

[2] P. Clarberg, W. Jarosz, T. Akenine-Möller, and H. W. Jensen, "Wavelet importance sampling: Efficiently evaluating products of complex functions," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 1166–1175, Jul. 2005.

[3] M. Pharr and G. Humphreys, *Physically Based Rendering, Second Edition: From Theory To Implementation*, 2nd ed. San Mateo, CA, USA: Morgan Kaufmann, 2010.

[4] T. Kroes, M. Eisemann, and E. Eisemann, "Visibility sweeps for joint-hierarchical importance sampling of direct lighting for stochastic volume rendering," in *Proc. 41st Graph. Interface Conf.*, 2015, pp. 97–104.

[5] D. Jnsson, E. Sundn, A. Ynnerman, and T. Ropinski, "A Survey of Volumetric Illumination Techniques for Interactive Volume Rendering," *Comput. Graph. Forum*, vol. 33, no. 1, pp. 27–51, 2014.

[6] S. Zhukov, A. Iones, and G. Kronin, "An ambient light illumination model," in *Proc. Eurographics Workshop Rendering Techn.*, 1998, pp. 45–56.

[7] P. Ljung, C. Lundstrm, and A. Ynnerman, "Multiresolution interblock interpolation in direct volume rendering," in *Proc. EuroVis*, 2006, pp. 259–266.

[8] T. Ropinski, J. Meyer-Spradow, S. Diepenbrock, J. Mensmann, and K. Hinrichs, "Interactive volume rendering with dynamic ambient occlusion and color bleeding," *Comput. Graph. Forum*, vol. 27, no. 2, pp. 567–576, 2008.

[9] T.-Y. Kim and U. Neumann, "Opacity shadow maps," in *In Proc. 12th Eurographics Workshop Rendering Techn.* 2001, pp. 177–182.

[10] L. Williams, "Casting curved shadows on curved surfaces," *SIGGRAPH Comput. Graph.*, vol. 12, no. 3, pp. 270–274, 1978.

[11] T. Lokovic and E. Veach, "Deep shadow maps," in *Proc. 27th Annu. Conf. Comput. Graph. Interactive Techn.*, 2000, pp. 385–392.

[12] M. Hadwiger, A. Kratz, C. Sigg, and K. Bühler, "GPU-accelerated deep shadow maps for direct volume rendering," in *Proc. 21st ACM SIGGRAPH/EUROGRAPHICS Symp. Graph. Hardware*, 2006, pp. 49–52.

[13] T. Ropinski, J. Kasten, and K. H. Hinrichs, "Efficient shadows for GPU-based volume raycasting," in *Proc. 16th Int. Conf. Central Europe Comput. Graph. Vis. Comput. Vis.*, 2008, pp. 17–24.

[14] T. Kroes, D. Schut, and E. Eisemann, *A Smooth Probability-Based Ambient Occlusion Approximation for Volume Rendering*. GPU Pro, 2015, ch. VI - 3, pp. 475–486. [Online]. Available: http://graphics.tudelft.nl/Publications-new/2015/KSE15

[15] P.-P. Sloan, J. Kautz, and J. Snyder, "Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 527–536, 2002.

[16] K. M. Beason, J. Grant, D. C. Banks, B. Futch, and M. Y. Hussaini, "Pre-computed illumination for isosurfaces," in *Proc. Conf. Vis. Data Anal.*, 2006, pp. 1–11.

[17] T. Ritschel, "Fast GPU-based visibility computation for natural illumination of volume data sets," in *Proc. Eurographics*, 2007, pp. 17–20.

[18] F. Lindemann and T. Ropinski, "Advanced light material interaction for direct volume rendering," in *Proc. IEEE/EG Int. Symp. Vol. Graph.*, 2010, pp. 101–108.

[19] J. Kronander, D. Jönsson, J. Löw, P. Ljung, A. Ynnerman, and J. Unger, "Efficient visibility encoding for dynamic illumination in direct volume rendering," *IEEE Trans. Vis. Comput. Graph.*, vol. 18, no. 3, pp. 447–462, Mar. 2012.

[20] E. Sundén, A. Ynnerman, and T. Ropinski, "Image Plane Sweep Volume Illumination," *IEEE Trans. Vis. Comput. Graph.*, vol. 17, no. 12, pp. 2125–2134, Dec. 2011.
[21] R. Fattal, "Participating media illumination using light propagation maps," *ACM Trans. Graph.*, vol. 28, no. 1, pp. 7:1–7:11, Feb. 2009.
[22] D. Patel, V. Šoltészová, J. M. Nordbotten, and S. Bruckner, "Instant convolution shadows for volumetric detail mapping," *ACM Trans. Graph.*, vol. 32, no. 5, 2013, Art. no. 154.
[23] C. Rezk Salama, "GPU-based monte-carlo volume raycasting," in *Proc. Pacific Graph.*, 2007, pp. 411–414.
[24] T. Kroes, F. H. Post, and C. P. Botha, "Exposure render: An interactive photo-realistic volume rendering framework," *PLoS One*, vol. 7, no. 7, 2012, Art. no. e38586.
[25] J. Novák, A. Selle, and W. Jarosz, "Residual ratio tracking for estimating attenuation in participating media," *ACM Trans. Graph.*, vol. 33, no. 6, 2014, Art. no. 179.
[26] S. Agarwal, R. Ramamoorthi, S. Belongie, and H. W. Jensen, "Structured importance sampling of environment maps," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 605–612, 2003.
[27] T. Kollig and A. Keller, "Efficient illumination by high dynamic range images," in *Proc. Eurographics Workshop Rendering Techniques*, 2003, pp. 45–51.
[28] V. Ostromoukhov, C. Donohue, and P.-M. Jodoin, "Fast hierarchical importance sampling with blue noise properties," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 488–495, 2004.
[29] E. Veach and L. J. Guibas, "Optimally combining sampling techniques for monte carlo rendering," in *Proc. 22nd Annu. Conf. Comput. Graph. Interactive Techn.*, 1995, pp. 419–428.
[30] D. Burke, A. Ghosh, and W. Heidrich, "Bidirectional importance sampling for direct illumination," in *Proc. 16th Eurographics Conf. Rendering Techn.*, 2005, pp. 147–156.
[31] D. Cline, P. K. Egbert, J. F. Talbot, and D. L. Cardon, "Two Stage Importance Sampling for Direct Lighting," in *Proc. Eurographics Symp. Rendering*, 2006, pp. 103–113.
[32] P. Clarberg and T. Akenine-Möller, "Practical product importance sampling for direct illumination," *Comput. Graph. Forum*, vol. 27, no. 2, pp. 681–690, 2008.
[33] F. Rousselle, P. Clarberg, L. Leblanc, V. Ostromoukhov, and P. Poulin, "Efficient product sampling using hierarchical thresholding," *Vis. Comput.*, vol. 24, no. 7–9, pp. 465–474, 2008.
[34] I. Georgiev, J. Křivánek, T. Hachisuka, D. Nowrouzezahrai, and W. Jarosz, "Joint importance sampling of low-order volumetric scattering," *ACM Trans. Graph.*, vol. 32, no. 6, 2013, Art. no. 164.
[35] E. Praun and H. Hoppe, "Spherical parametrization and remeshing," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 340–349, 2003.
[36] H. Niederreiter, *Random Number Generation and quasi-Monte Carlo Methods*. Philadelphia, PA, USA: Soc. Industrial Appl. Math., 1992.
[37] G. M. Morton, "A computer oriented geodetic data base and a new technique in file sequencing," IBM Ltd., Portsmouth, U. K., 1966.
[38] L. C. Henyey and J. L. Greenstein, "Diffuse radiation in the galaxy," *Astrophysical J.*, vol. 93, pp. 70–83, 1941.
[39] O. Elek, T. Ritschel, C. Dachsbacher, and H.-P. Seidel, "Principal-ordinates propagation for real-time rendering of participating media," *Comput. Graph.*, vol. 45, pp. 28–39, 2014.
[40] M. Levoy and P. Hanrahan, "Light field rendering," in *Proc. 23rd Annu. Conf. Comput. Graph. Interactive Techn.*, 1996, pp. 31–42.

Philipp von Radziewsky received the BSc degree from the University of Lübeck, Lübeck, Germany, and the MSc degree from Saarland University, Saarbrücken, Germany, in 2012 and 2015, respectively. He is currently working toward the PhD degree at Delft University of Technology. His main research interests include realistic image synthesis, real-time rendering, and geometry processing.

Thomas Kroes received the bachelor's degree in mechanical engineering and the master's degree in industrial design engineering. He received doctorate in computer science from Delft University of Technology (TU Delft). He is interested in medical visualization, and in particular real-time photorealistic volume-visualization. He is the author of Exposure Render, an interactive photorealistic volume rendering framework. He is a member of the IEEE.

Martin Eisemann received the diploma in computational visualistics from the University of Koblenz-Landau, Germany, in 2006, and the PhD degree in computer graphics from TU Braunschweig, Germany, 2011, He is a professor with TH Köln, since 2015. From 2011 until 2015, he was Akademischer Rat (postdoctoral researcher) in the Computer Graphics Lab, TU Braunschweig. Between 2007 and 2014 he was a visiting researcher with several institutions including TU Delft, Saarland University, EDM Hasselt, and the Max-Planck Institut für Informatik. His main research interests include image- and video-based rendering and editing, visual analytics, and realistic and interactive rendering.

Elmar Eisemann received the PhD degree from Grenoble Universities, INRIA Rhône-Alpes. He is a professor with Delft University of Technology and head of the Computer Graphics and Visualization group. Before, he was an associate professor with Télécom ParisTech and senior researcher in the Cluster of Excellence, MPII/Saarland University. He was Normalien witht the École Normale Supérieure. His interests include real-time and perceptual rendering, alternative representations, shadow algorithms, global illumination, and GPU acceleration techniques. In 2011, he was honored with the Eurographics Young Researcher Award. He is a member of the IEEE.