EUROGRAPHICS 2025 / A. Bousseau and A. Dai
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 44 (2025), Number 2

View-Dependent Visibility Optimization for Monte Carlo Volume Visualization

Nathan Lerzer and Carsten Dachsbacher

Institute for Visualization and Data Analysis, Karlsruhe Institute of Technology

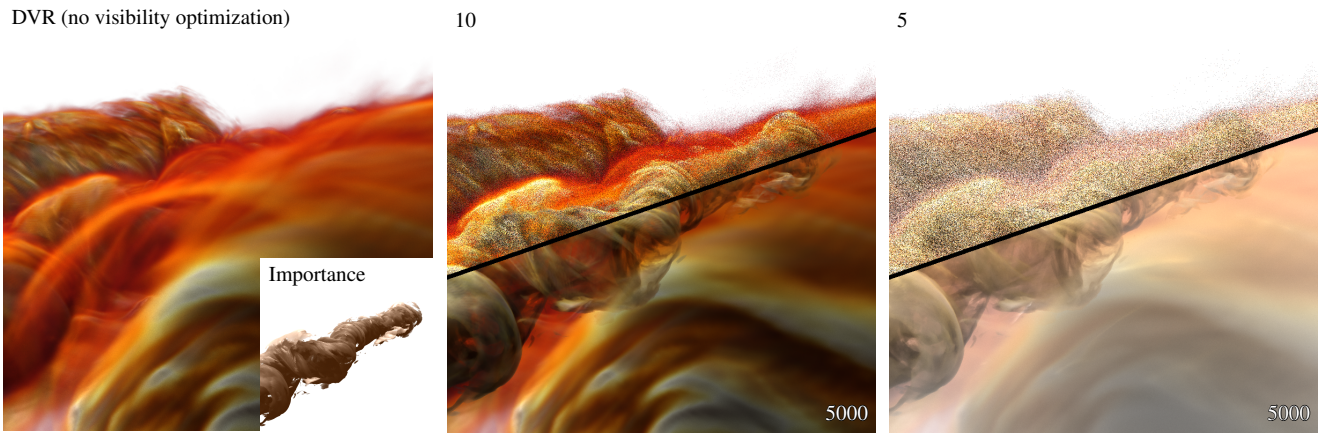

Figure 1: *Comparison between volume rendering without optimization (DVR), our method with single scattering and our method with multiple scattering (resolution* 800×800*). For single scattering we achieve a render time of* 150*ms (10 samples, RMSE* 0.102711*,* ꟻLIP 0.136049*) and* 71.40*s (5000 samples, RMSE* 0.0057634*,* ꟻLIP 0.0158829*). With extinction optimization for single plus multiple scattering the render time is* 726*ms (5 samples, RMSE* 0.205473*,* ꟻLIP 0.190859*) and* 711.55*s (5000 samples, RMSE* 0.0089001*,* ꟻLIP 0.0145564*).*

Abstract

Compared to classic ray marching-based approaches, Monte Carlo ray tracing for volume visualization can provide faster frame times through progressive rendering, improved image quality, and allows for advanced illumination models more easily. Techniques such as the view-dependent optimization of visibility and illumination of important regions, however, have been formulated for ray marching and rely on stepwise sampling along rays, and are thus incompatible with free-flight distance sampling of state-of-the-art Monte Carlo methods. In this paper we derive such a view-dependent optimization for Monte Carlo ray tracing where the visibility to the camera, the illumination and opacity of important regions is optimized for both single and multiple scattering rendering. For this we define a post-interpolative importance function, introduce an efficient data structure to sample, approximate and optimize the integrated extinction along rays, and devise an efficient Monte Carlo estimator for interactive visualization. Our method enables view-dependent visibility optimization with moderate memory overhead and unbiased, progressive Monte Carlo volume visualization. We demonstrate our method for various volume data sets as well as for data-dependent and spatially-dependent importance functions.

CCS Concepts

• ***Human-centered computing*** → ***Scientific visualization;***

1. Introduction

Volume visualization is an important tool in many fields such as medicine, life sciences, physics, and engineering. With direct volume rendering (DVR) the data is sampled and classified during rendering, e.g. to map data to absorption and emission of a volume [Max95]. The demands and developments in this field in the last decades led to a comprehensive set of techniques for mapping the data [LKG*16; XSL21] and to handle large data and heterogeneous representations [BGI*14; BHP15; SZD*23]. To support the utmost goal of visualization, many techniques have been conceived

for enhancing the perception of important features in the volume data. User studies, e.g. [LR11], have shown that advanced illumination effects are indispensable and exemplary techniques include advanced volumetric shading [JSYR14], better shading of surface-like structures [IZM18], anisotropic shading [AD16] and illustrative techniques [BG07], or optimizing shadowing [ASDW14], visibility [VKG05; GRT13; GRT14; GSE*14; ZRPD20] and/or illumination of important regions [AD16; AZD17].

In recent years, volume visualization based on Monte Carlo (MC) ray tracing [KPB12; Sal07; WMZ22; MSG*23; ZWS*24] gained more interest as these methods can achieve smaller latency, enable progressive rendering, and at the same time facilitate the integration of advanced illumination models and reaching high final image quality. These methods also benefit from significant research progress in physically-based volume rendering from outside of visualization [MGJ19; GMH*19] and from advances in denoising techniques facilitating interactive visualizations [IMM22; XCC*24].

Many advanced visualization techniques for enhancing the perception in DVR have been formulated for ray marching-based approaches which have dominated DVR for decades. They often rely on step-wise sampling along rays and are thus incompatible with free-flight distance sampling of state-of-the-art Monte Carlo methods [NGHJ18].

One class of techniques, which is also the focus of this paper, enhances the visibility and illumination of important regions in volumes: this requires information where important and non-important regions along rays are located, e.g. to locally modify the volume's extinction [AZD17]. The typical approach of marching along rays and collecting all the information is what makes these techniques as-is fundamentally incompatible with Monte Carlo methods.

In this paper we present a view-dependent optimization of the opacity, visibility, and direct illumination of important regions in DVR designed for Monte Carlo ray tracing with both single and multiple scattering rendering. Our contributions are:

- we formulate the extinction optimization for MC ray tracing and define a post-interpolative importance function that maps volume attributes and spatial locations inside the volume to importance which eventually steers the optimization,
- we introduce an efficient data structure to sample, approximate, and integrate the importance values along rays which is required to enhance the visualization by optimizing the extinction, and
- a Monte Carlo estimator for interactive visualization.

We demonstrate that our method achieves interactive, view-dependent optimizations with moderate memory overhead and unbiased, progressive Monte Carlo volume visualization. We evaluate our method for various volume data sets as well as for data-dependent and spatially-dependent importance functions.

2. Related Work

There is a tremendous body of work for (direct) volume visualization and a comprehensive overview is clearly beyond the scope of this work. We refer the reader to the excellent text books and surveys, beginning with Engel et al. [EHK*06] and Hadwiger et al. [HLSR09] covering earlier works, and more recent surveys focusing on transfer functions [LKG*16], handling large volume data [BHP15], compression [BGI*14], and heterogeneous representations [SZD*23]; we focus on the most closely related works.

Our work belongs to the categories of advanced volumetric shading [JSYR14], perceptually-motivated techniques [PBC*16] and feature enhancement [XSL21]. One crucial aspect in DVR is volume illumination and shading which significantly contributes to the depth and shape perception of features and, depending on the application, can be supported by appropriate illumination models [Max95], surface/feature shading [IZM18; AD16], non-local ambient techniques [AWD16], or global illumination [KPB12]. Surface shading and ambient techniques focus on illumination effects dictated by local properties (e.g. gradients) or small regions of the volume, while transfer function design and physically-based volume rendering have a global effect on the rendered volume. While slicing, selection and cut-away views can be used manually to explore individual features, the challenge of occlusion, and insufficient or inappropriate illumination caused by distant structures of the volume data can also be addressed in a possibly view-dependent, automatic manner. Examples include controllable soft shadows to remove disturbing patterns [ASDW14], importance-driven feature enhancement for cut-away and ghosted views [VKG05], transfer function generation based on visibility histograms [CM11], and avoiding occlusions by reducing the contributions of previous samples during ray marching [ZTL*15]. Inspired by work on opacity optimization for line- and surface-based visualizations [GRT14; GSE*14], Ament et al. [AZD17] presented a real-time view-dependent optimization of the visibility and illumination based on an importance function. This method is the starting point of our work and is briefly recapitulated in Section 4. Himmler and Günther [HG24] show how camera positions to find ideal viewpoints can be optimized concurrently with volume extinction.

3. Background

The volume rendering equation describes the light transport in heterogeneous participating media [Cha13]. More specifically we formulate our work using the null-scattering volume rendering equation [MGJ19]. We aim to optimize the visibility to the camera and direct illumination (i.e. visibility to light sources) of important volumetric features. Our work can be applied to the full transport including multiple scattering (see Sec. 6.4), but for notational lightweightedness we reduce the following explanation to single-scattering contributions only. Here each scattering vertex $\mathbf{y}$ along a camera ray is directly connected to a light source and we determine the incoming radiance $L(\mathbf{x},\omega)$ at point $\mathbf{x}$ from direction ω:

$$L(\mathbf{x},\omega) = \int_0^\infty T(\mathbf{x},\mathbf{y}) \Big(\mu_a(\mathbf{y}) L_e(\mathbf{y},\omega) + \mu_s(\mathbf{y}) L_s(\mathbf{y},\omega) + \mu_n(\mathbf{y}) L(\mathbf{y},\omega) \Big)\, dy \tag{1}$$

with the combined transmittance $T(\mathbf{x},\mathbf{y}) = e^{-\int_\mathbf{x}^\mathbf{y} \bar{\mu}(\mathbf{z})\, d\mathbf{z}}$ between points $\mathbf{x}$ and $\mathbf{y} = \mathbf{x} + t \cdot \omega$ with $\bar{\mu} = \mu_a + \mu_s + \mu_n$, the emitted radiance $L_e(\mathbf{y},\omega)$ of the volume, the real scattering part $L_s(\mathbf{y},\omega) = f(\mathbf{y},\omega,\omega_L) T(\mathbf{y},\mathbf{y}_L) L_e$ for the point $\mathbf{y}_L$ on a light source with direction $\omega_L = \frac{\mathbf{y}_L - \mathbf{y}}{||\mathbf{y}_L - \mathbf{y}||}$, and the null scattering part $L(\mathbf{y},\omega)$. A list of the

μ_a, μ_s	absorption and scattering coefficient
μ_n, μ_t	null-scattering and real extinction coefficient
$\bar{\mu}$	majorant extinction coefficient

Table 1: *List of interaction coefficients used throughout the paper.*

interaction coefficients used throughout the paper can be found in Table 1. Many scatter functions exist; in our implementation we use $f(\mathbf{y},\omega,\omega_L) = f_v(\mathbf{y},\omega,\omega_L) + ||\nabla\mu_t||(H\cdot N_{\mu_t})^n(N_{\mu_t}\cdot\omega_L)_+$ using the Henyey-Greenstein phase function [HG41] and the Blinn-Phong BRDF [Bli77] with normal $N_{\mu_t} = \frac{\nabla\mu_t}{||\nabla\mu_t||}$ ($\mu_t = \mu_a + \mu_s$) computed from the gradient of the extinction field and exponent n, scaled by $||\nabla\mu_t||$ to simulate surface like regions in the volume.

Woodcock Tracking To approximate a solution to Eq. (1), Monte Carlo methods sample free-flight distances. For this we use Woodcock (delta) tracking [WMHL65; NGHJ18] which results in the following estimator:

$$\langle L(\mathbf{x},\omega)\rangle = \frac{T(\mathbf{x},\mathbf{y})}{p(\mathbf{y})}\Big(\langle\mu_a(\mathbf{y})L_e(\mathbf{y},\omega)\rangle_{P_a} + \langle\mu_s(\mathbf{y})L_s(\mathbf{y},\omega)\rangle_{P_s} + \langle\mu_n(\mathbf{y})L(\mathbf{y},\omega)\rangle_{P_n}\Big) \tag{2}$$

with collision probabilities $P_* = \frac{\mu_*(\mathbf{y})}{\bar{\mu}}$ ($*\in\{a,s,n\}$). Note that this is a two sample estimator which first samples a tentative collision location $\mathbf{y}$ along a ray based on the combined transmittance with $p(\mathbf{y}) = \bar{\mu}\cdot T(\mathbf{x},\mathbf{y})$, and then classifies this collision based on the collision probabilities as either absorbing $L_e(\mathbf{y},\omega)$, real scattering $L_s(\mathbf{y},\omega)$, or null scattering $L(\mathbf{y},\omega)$.

4. Visibility Optimization and Monte Carlo Methods

Our goal is to optimize the visibility and illumination of important regions in Monte Carlo (MC) volume visualizations by modifying the extinction $\mu_*(.)$ along rays. In MC methods with Woodcock tracking this is tantamount to increasing the null-collision probability of the media between the camera/light source and the important regions. We first recapitulate the error function $E_i(\mu_i')$ introduced by Ament et al. [AZD17] to optimize the extinction in a ray marching setting, and subsequently derive a similar optimization for the continuous case. They equidistantly march along rays and minimize the error function $E_i(\mu_i')$ at each step i to determine an optimized extinction value μ_i' of a discrete ray sample (or segment), taking into account user-defined importance values $g_i = g(\mathbf{x}_i)$ and g_j sampled from a discretized importance field $g(\mathbf{x})\in[0,1]$ until the rays exit the medium at g_m:

$$E_i(\mu_i') = p(\mu_i'-\mu_i)^2 + \mu_i'^2(1-g_i)^{2\lambda}\left(r\sum_{j=1}^{i-1}g_j^2 + q\sum_{j=i+1}^{m}g_j^2\right), \tag{3}$$

using $\lambda\geq 0$ to control the emphasis of important regions, and weights $p,q,r\geq 0$ to penalize deviation of extinction coefficients from the sampled extinction (p), occlusion from unimportant regions closer to the camera (q), and visual background clutter (r). Note that the latter two terms in the error function require the sum of importance values in front of and behind the current ith sample.

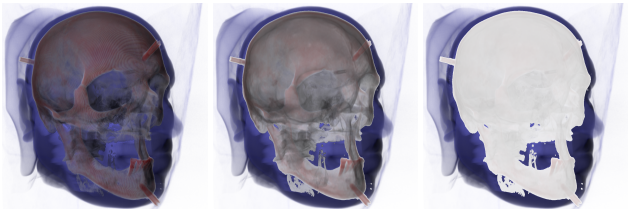

Figure 2: *The error function $E_i(\mu_i')$ [AZD17] leads to vanishing of the important regions when decreasing the step sizes (from left to right: 0.01, 0.001, and 0.0001) with $g(\mathbf{x})\in[0,0.9]$.*

The extinction μ_i' minimizing this error is computed as [AZD17]:

$$\mu_i' = \frac{p\mu_i}{p+(1-g_i)^{2\lambda}\left(r\sum_{j=1}^{i-1}g_j^2 + q\sum_{j=i+1}^{m}g_j^2\right)}. \tag{4}$$

4.1. Continuous Formulation

The error function cannot trivially be used in the continuous case by setting m to ∞ as all extinction values along a ray become 0 if all importance values are less than 1 and there exists a region with importance larger than 0 (see Fig. 2):

$$\lim_{m\to\infty} p+(1-g_i)^{2\lambda}\left(r\sum_{j=1}^{i-1}g_j^2 + q\sum_{j=i+1}^{m}g_j^2\right) = \infty, g_i<1, \exists g>0. \tag{5}$$

To derive the continuous formulation, we first replace the sums with their integrals between the volume entry $\mathbf{x}$ and exit $\mathbf{z}$ points and then split and reorder the integrals:

$$E(\mu_*'(\mathbf{y})) = p(\mu_*'(\mathbf{y})-\mu_*(\mathbf{y}))^2 + \mu_*'(\mathbf{y})^2(1-g(\mathbf{y}))^{2\lambda}h(\mathbf{y}) \tag{6}$$

$$\begin{aligned}\text{with } h(\mathbf{y}) &= r\int_{\mathbf{x}}^{\mathbf{y}} g(\mathbf{i})^2\,d\mathbf{i} + q\int_{\mathbf{y}}^{\mathbf{z}} g(\mathbf{i})^2\,d\mathbf{i}\\ &= r\int_{\mathbf{x}}^{\mathbf{y}} g(\mathbf{i})^2\,d\mathbf{i} + q\left(\int_{\mathbf{x}}^{\mathbf{z}} g(\mathbf{i})^2\,d\mathbf{i} - \int_{\mathbf{x}}^{\mathbf{y}} g(\mathbf{i})^2\,d\mathbf{i}\right)\\ &= (r-q)\int_{\mathbf{x}}^{\mathbf{y}} g(\mathbf{i})^2\,d\mathbf{i} + q\int_{\mathbf{x}}^{\mathbf{z}} g(\mathbf{i})^2\,d\mathbf{i}.\end{aligned} \tag{7}$$

Note how the formulation of Eq. (7) can simplify the calculation: we can calculate the second term once for an entire ray, and at each sampled location along the ray $\mathbf{y}$ we calculate the first term.

With this new error function we can find the optimal extinction coefficients $\mu_*'(\mathbf{y})$ by setting the first derivative to 0 to get the minimum of $E(\mu_*'(\mathbf{y}))$ (see the appendix for a formal derivation) as:

$$\mu_*'(\mathbf{y}) = \frac{p\mu_*(\mathbf{y})}{p+(1-g(\mathbf{y}))^{2\lambda}h(\mathbf{y})}. \tag{8}$$

Obviously, computing $\mu_*'(\mathbf{y})$ in practical MC rendering requires a good estimate for $h(\mathbf{y})$. The simplest and naive estimator would use random samples g_a and g_b along the ray:

$$\langle h(\mathbf{y})\rangle_{\text{Naive}} = (r-q)(\mathbf{y}-\mathbf{x})g_a^2 + q(\mathbf{z}-\mathbf{x})g_b^2. \tag{9}$$

Unfortunately, this naive approach is not practical if we consider the procedure of MC volume visualization: we would sample a

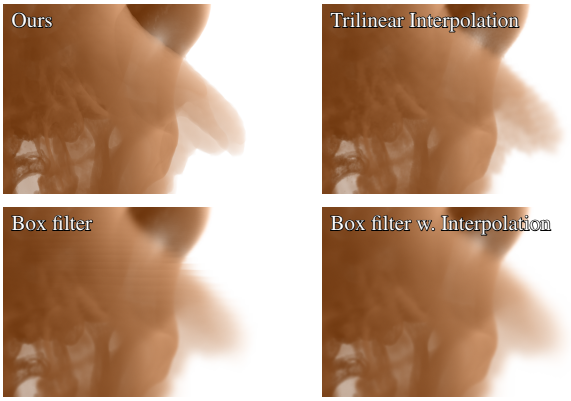

Figure 3: *Our new importance function enables post-interpolative classification and preserves fine details (top-left). Previous work relies on pre-interpolative classification with trilinear interpolation, filtering, or both.*

point $\mathbf{y}$ on the ray using distance sampling with the majorant extinction coefficient, and then we would have to classify whether it is a null collision or not. To do so, we need a good estimate of $h(\mathbf{y})$ which is costly to compute as it requires many additional samples. And as the estimator depends on $\mathbf{y}$ we would not be able to reuse these samples for the next sampled $\mathbf{y}$. In the next sections we describe how we define the importance function and how this leads to a significantly more efficient approximation of $h(\mathbf{y})$ and a practical rendering algorithm.

4.2. Post-Interpolative Importance Function

In previous work [AZD17] the importance function $g(\mathbf{x})$ was based on a pre-interpolative classification for every voxel with additional smoothing to reduce the introduced discontinuities from the discretized $g(\mathbf{x})$. In principle there are two intentions when defining the importance: (1) either regions whose attributes fulfil a certain criterion are considered important (e.g. mapping scalar values or gradients to importance), and/or (2) the importance depends on the spatial coordinates $\mathbf{x}$. In contrast to previous work, we propose a *post-interpolative importance* that covers these use cases and at the same time eliminates the need for smoothing. This preserves detail of important regions better (Fig. 3). Our importance maps points in the volume to n distinct values in $[0,1]$ (see below for a discussion) and consequently, we find segments of the same value along a ray through the volume. The importance function along a ray $k(t)$ thus is a step function mapping every point $\mathbf{y} = \mathbf{x} + t \cdot \omega \in \mathbb{R}^3$ to one of these importance values:

$$k(t) = \sum_{i=0}^{n} \alpha_i \chi_{A_i}(t), \text{ with} \tag{10}$$

$$\chi_A(t) = \begin{cases} 1, & \text{if } t \in A \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

where n is the number of segments along the ray with individual importance values, $\alpha_i \in [0,1]$ are these distinct importance values, and the A_i are the mutually exclusive segments along the ray.

That is, our new importance function defines distinct regions of the same importance (cf. Fig. 4). While this might seem a limitation at first compared to an arbitrary importance function $g(\mathbf{x})$, we found that the intended visibility optimization typically requires very small n, most often $n = 1$ is sufficient. Nevertheless using discrete segments leads to the limitation that smooth importance functions, like the region of interest used in Ament et al. [AZD17] can only be approximated. However, as we will see in the next section, this new importance function, together with a data structure to represent $k(t)$ efficiently, will allow us to formulate an efficient unbiased estimator for MC volume visualization.

5. Representing and Estimating Importance $k(t)$

The fact that our new importance function consists of distinct regions of the same importance has important advantages: we can compactly represent $\tilde{k}(t) \approx k(t)$ during MC volume rendering, progressively refine the approximation $\tilde{k}(t)$, and also precompute the integral $K_i = \int_0^{t_i} \tilde{k}(s)^2 ds$ easily (t_i is the lower bound (position) of the segment A_i). For each segment in the data structure we store a tuple consisting of its location along the ray t_i, the length of the segment l_i, its importance value k_i and the precomputed integral K_i. Recall that $\tilde{k}(s)$ is a step function and the integral can thus be computed as a simple sum. Fig. 4 shows an example of samples and the respective approximation.

In this section we will describe building and progressively updating the data structure to optimize the extinction along a camera ray (other rays, e.g. connecting to light sources, are discussed afterwards) and describe its use during MC rendering which consists of the following steps (carried out for every ray independently):

1. initialize the empty data structure,
2. draw N random samples k'_{t_i}, $i = 1..N$, of the importance function along the ray,
3. insert the samples into the data structure (Sect. 5.1),
4. perform Woodcock tracking using the optimized extinction from the data structure until we reach a scattering or absorption event,
5. in case of a scattering event, sample the light sources and compute the (optionally: visibility optimized) transmittance (see Section 5.3),
6. optionally: calculate the multiple-scattering contribution (see Section 5.4).

To perform progressive rendering, the steps 2 to 6 are repeated and the results are accumulated in image space. In Section 5.2 we discuss further sampling optimizations.

5.1. Construction and Update of the Importance Segments

To faithfully approximate $k(t)$ we have to sample the importance along an entire ray to account for both the occlusion of important regions from the camera and visual clutter from behind. In step 2 (see above) we draw N samples which are inserted into the data structure which possibly already contains samples from previous progressive rendering passes. The insertion, which we describe below, is more efficient if the samples are sorted according to t_i which we can easily ensure when using stratified sampling along the ray.

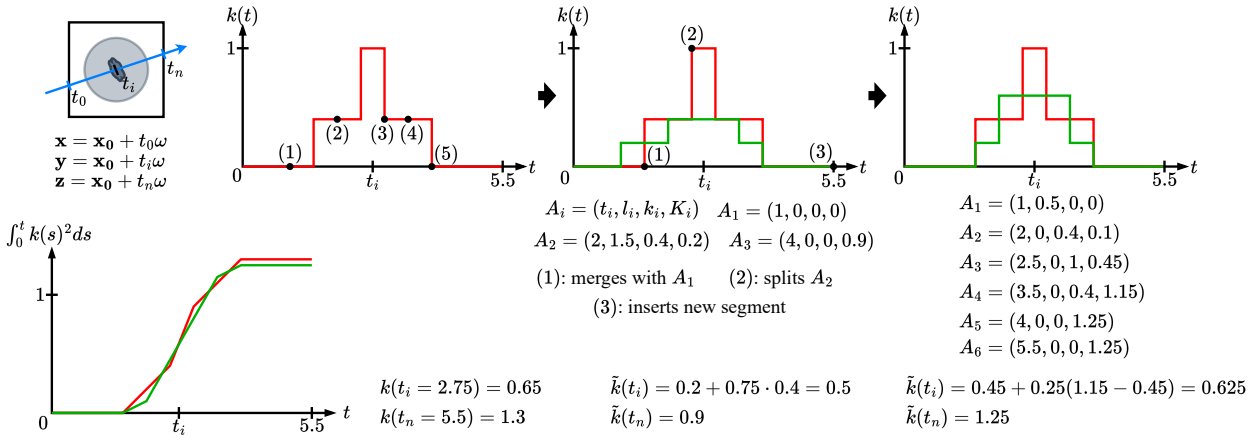

Figure 4: *Schematic overview of two insertion steps into our data structure illustrating the different insertion operations. The red curve represents the importance along the ray through the volume in the top left. Our reconstructed importance is shown as the green line. The extracted segments (consisting of location, length, importance and integral before the segment) created by the new samples (points along the red curve) used for the reconstruction are listed below the corresponding graphs. The true integral $k(t)$ as well as the computation of the approximated integral $\tilde{k}(t)$ are shown below each insertion step. In the bottom left a graph of the true integral in red along with the reconstructed integral after the second insertion step can be seen.*

Updating the data structure When updating the data structure we have two sorted lists of samples; the already stored ones (possibly an empty list) and the newly sampled ones. We then merge the two into a single new list which represents the refined $\tilde{k}(t)$. In order to keep our data structure compact, we make the following assumption: if the distance $||t_i - t_j||$ between two samples i and j is less than a user-defined threshold d, we consider them close enough to provide a sufficient approximation of $k(t)$ and omit new samples of the same importance value in between. Note that $\tilde{k}(t)$ would still be refined once other importance values in such regions are sampled.

Our merge algorithm (Alg. 1) takes the previous list of samples *segments*, the newly sampled list *samples*, and d as input and proceeds as follows:

- if two previously inserted samples at t_i and t_{i+1} form a segment, i.e. $||t_i - t_j|| < d$, and have the same importance value as a new sample j which lies on the segment $t_i < t_j < t_{i+1}$ then we discard sample j (line 11-13).
- if a new sample j has the same importance as a sample i, and $||(t_i + l_i) - t_j|| < d$, and no other samples are between t_i and t_j then we store sample j and consider the two forming a new segment (possibly extending an already existing segment) (line 7-10).
- if a new sample j lies on a segment $[t_i, t_{i+1}]$ but has a different importance value, then we store sample j and by this separate the segment $[t_i, t_{i+1}]$ into two new samples t_i and t_{i+1} thus forming two new segments at their location (line 15-20).
- otherwise we insert the segment (line 22).

Along with this merging we also precompute the integral K_i for each segment A_i. As there may be not yet sampled regions along a ray, we approximate the importance there as the mean importance value of the two consecutive, surrounding segments A_i and A_{i+1}; for empty regions at the begin and end of the ray we assume the border has an importance of 0. We then compute and accumulate the integrals for each segment and empty region, and store the result in the next segment A_{i+1}. (cf. Fig. 4)

Computation of $h(\mathbf{y})$ during rendering With the integrals precomputed, the computation of both integrals in $h(\mathbf{y})$ (Eq. (7)) simplifies to (1) a binary search of the segment $[t_i; t_{i+1})$ containing t (note $\mathbf{y} = \mathbf{x} + t \cdot \omega$), and (2) looking up the integral to the left of this segment (stored along with t_i) and adding the integral inside the segment. The latter is a simple multiplication of the length before t inside the segment and the importance: $(t - t_i) \cdot k_i$. In case that t lies between two segments $[t_{i-2}; t_{i-1})$ and $[t_i; t_{i+1})$ with $t_{i-1} \leq t$ and $t < t_i$, we linearly interpolate between the stored integral of the segment to the right (A_i) and the integral stored for the left segment (A_{i-2}) plus the integral of the left segment. (cf. Fig. 4)

5.2. Optimizing the Importance Sampling

As mentioned, we maintain one data structure per camera ray and use stratified sampling by dividing the whole ray into N equally-sized regions to obtain a sorted list of new samples which are inserted into the data structure during progressive rendering. Since our data structure essentially tracks locations where the importance value along a ray changes, we can alternate uniform strata sampling and an improved scheme that places the strata adaptively:

- in non-sampled regions between segments,
- within segments to detect regions where splits (changes in importance) occur, and
- in non-sampled regions close to the begin and end of the ray.

In our experiments the accuracy of $\tilde{k}(t)$ increased extremely fast when performing one uniform sampling, followed by three adaptive sampling steps in the subsequent progressive rendering passes.

Algorithm 1: Insertion into combined List

Input: size of list n, list of segments *list*, new candidate *cand*, merge distance d
Output: element which has to be processed later, n

```
if n == 0 then
    /* Insert segment in empty list */
    insert(list, cand);
    return null, 1;
curr = list[n − 1];
if curr.importance == cand.importance then
    if cand.left() − curr.right() < d then
        /* Merge segments */
        curr.right() = cand.right();
        return null, n;
    if cand in curr then
        /* Discard redundant segment */
        return null, n;
else
    if cand in curr then
        /* Split current segment */
        temp = curr.right();
        curr.right() = curr.left();
        insert(list, cand);
        return temp, n + 1;
/* Insert new segment */
insert(list, cand);
return null, n + 1;
```

5.3. Light Ray Extinction Optimization

In the previous sections we optimized the visibility along camera rays. In order to optimize the direct illumination, i.e. the extinction values between a scatter vertex $\mathbf{y}$ and a point on the light source, we use the same data structure. Note that reusing it across progressive rendering passes is not possible as the scatter vertices' locations and thus the rays to the light source change.

We proceed as follows: for every such ray we use stratified samples and insert them into the data structure where possibly merging operations take place. Note that splitting of segments or discarding of samples are only required for progressive updates of the data structure, and can be left out for an optimized implementation of Alg. 1. Using the data structure quickly amortizes, because during the transmittance estimation between the scattering vertex and the light source we use ratio tracking [NSJ14] which produces multiple vertices each of which requires an optimized extinction.

5.4. Monte Carlo Rendering with Extinction Optimization

We now briefly summarize the Monte Carlo rendering using our estimation of $h(\mathbf{y})$ starting with the classic single scattering setting: We use delta tracking to sample tentative collisions along camera rays. The collision location and the sampled extinction there are then used to evaluate the data structure (which has already been built/refined) and to compute the optimized extinction coefficient (Eq. (8)). With this $\mu'_*(\mathbf{y})$ we compute the decision probabilities which we use to determine the corresponding event of the delta tracking algorithm. If we classify the collision as scattering, we compute the transmittance to the light source, optionally using the optimization from Section 5.3.

Name	Size	Fig.
VISIBLE HUMAN *additional head images* *male lower body* [AHH*22]	$512 \times 512 \times 463$ $1360 \times 843 \times 1309$	3,6,9 10
HEPTANE *flame*	$302 \times 302 \times 302$	5,12
CHAMELEON	$1024 \times 1024 \times 1080$	8
SCRAMBLER *Phone Telta P-171D-ATS* [GR23]	$2177 \times 1935 \times 1306$	5
MAGNETIC RECONNECTION [GLDL14]	$512 \times 512 \times 512$	1
FOOT	$256 \times 256 \times 256$	11

Table 2: *List of datasets used in this paper; for visualization we scale the volumes to fit into the cube* $[-1,1]^3$.

Extension to Multiple Scattering We can extend the visualization to also include multiple scattering contributions. This can easily be achieved by sampling an additional new ray at scattering locations according to the phase function and accounting for their contributions. If we want to optimize the illumination of important regions we have two options in this setting: (1) optimize only at the first scattering event, i.e. direct illumination, or (2) optimize the extinction for every ray that directly connects a scatter vertex to a light source. We have found the latter options to improve the visualization results, e.g. when important regions are shadowed by other important regions. Optimizing the extinction between scattering events along longer transport paths is costly and we did not observe visual improvements. We provide a comparison of the different options in Fig. 9.

6. Evaluation

In this section we briefly provide implementation details, and present and discuss results of our method and compare it to ray marching-based extinction optimization [AZD17]. All timings were measured using a Ryzen 9 7900x with 32GB of RAM and an NVIDIA RTX 4080. We implemented our method in CUDA without any performance optimizations for volume rendering such as early termination, empty space skipping, or acceleration structures for tight extinction majorants [NGHJ18]; to reduce noise in the final image we can optionally enable the Optix denoiser. For the images we use a resolution of 800×800 and the extinction optimization parameters $p = 1$, $q = 70$, $r = 30$, and $\lambda = 1$ as well as 200 samples per direct illumination ray unless stated otherwise. To evaluate the visual quality of the images, we provide the RMSE and ꟻLIP [ANA*20] error compared to reference images with 10000 samples per pixel. Table 2 lists the datasets used for evaluation.

6.1. Implementation Details

We store the data structures for all pixels (Alg. 1) in one memory pool in global GPU memory. To access the data structure per pixel, we maintain two arrays storing the size and the offset of the respective data structure in the pool. To update this pool, we use

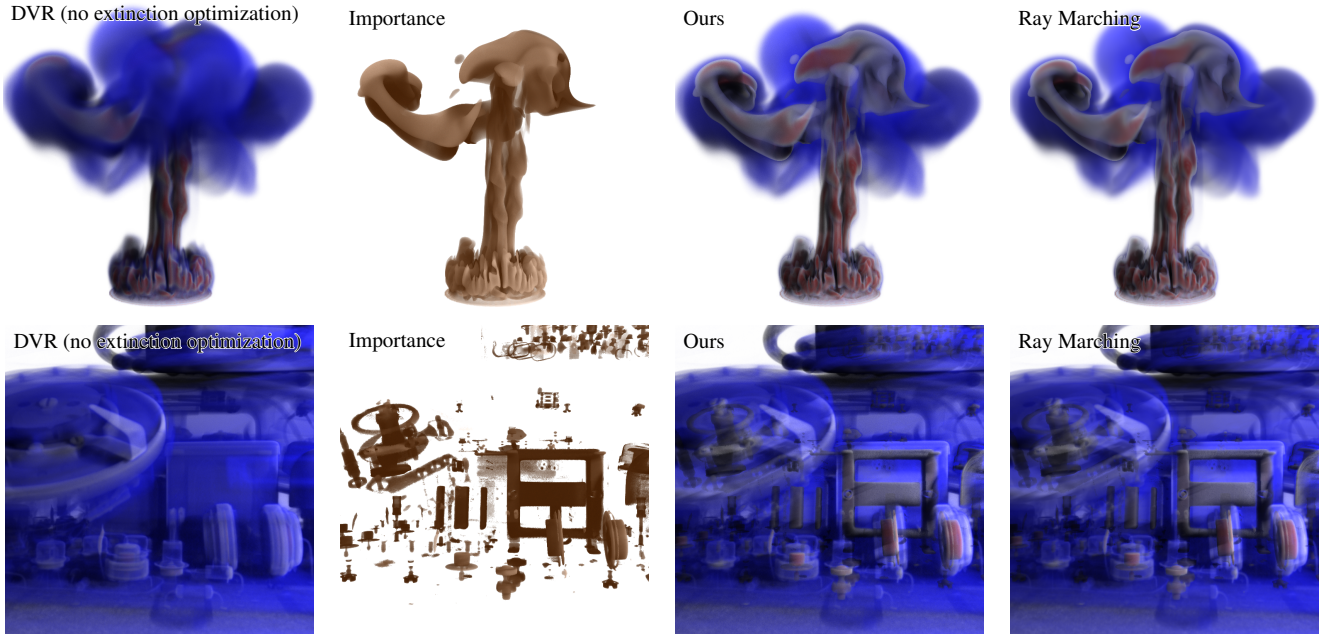

Figure 5: *Equal-time comparison of* 14.7*s (seconds) for the* HEPTANE *dataset with ray marching (step size* 0.002*, RMSE* 0.00400718*,* FLIP 0.0126611*) and our method (*2700 *samples, RMSE* 0.0061162*,* FLIP 0.0106164*) and of* 72*s for the* SCRAMBLER *dataset with ray marching (step size* 0.0009*, RMSE* 0.00480352*,* FLIP 0.0220397*) and our method (*1100 *samples, RMSE* 0.0102304*,* FLIP 0.0256413*). The render times of our approach without light optimization compared to standard DVR are for* HEPTANE *(DVR* 3.48*s, Ours* 10.08*s) and for* SCRAMBLER *(DVR* 21.80*s, Ours* 24.54*s)*

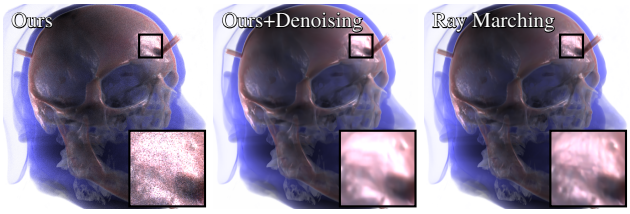

Figure 6: *Equal-time comparison at* 1*s with the* VISIBLE HUMAN *dataset with our method (*150 *samples, RMSE* 0.0358941*,* FLIP 0.052986*), our method with denoising (*150 *samples, RMSE* 0.0109182*,* FLIP 0.0417109*), and denoised jittered ray marching (step size* 0.008*,* 1 *sample, RMSE* 0.0268515*,* FLIP 0.0780313*). For shading we used the Blinn-Phong BRDF with* $n = 40$*.*

two buffers (in each iteration): a temporary buffer where we store the new samples, and a new buffer that has enough space for all new sample lists. In the best case, we would only need to store four floats per segment and two integer values per pixel, but due to the fact that we do not know beforehand, how many segments will be split in each iteration, we allocate double the size of the existing data structure (for the worst case that each segment is split). A detailed memory analysis can be found in Section 6.8.

6.2. Comparison with Ray Marching

A comparison with Ament et al.'s extinction optimization [AZD17] suggests itself. However, as the results of their optimization depend on the ray marching step size (Fig. 2), we make the comparison more meaningful by replacing their importance function with ours and also use our integrals approximated by a sum: $\sum_{j=0}^{n} k_j^2 \Delta x \approx \int_{t_0}^{t_n} k(s)^2 \, ds$, where Δx is the step size. Note that this has no negative impact on the performance of this method. Fig. 5 shows equal-time renderings for the HEPTANE and SCRAMBLER dataset with extinction optimization and a rendering without optimization (DVR) as reference. The render time is chosen such that the ray marching step size is just small enough so that banding artefacts are not noticeable. For the SCRAMBLER dataset we only use one light source (in contrast to two used in the HEPTANE dataset), because with ray marching the render time scales linearly with the number of light sources (unless light sources would also be sampled stochastically). We can observe that the residual noise in the MC visualizations is only perceivable when zooming in with these render times.

To show the effectiveness of denoising (Fig. 6) we reduce the number of samples per pixel. Given the same render time, ray marching without jittering would introduce visible banding artefacts that cannot be removed by denoisers. Still, ray marching with jittering significantly reduces the brightness of the specular highlights which leads to worse error metrics with and without denoising (RMSE 0.0268515, FLIP 0.0780313) compared to plain ray marching (RMSE 0.0167433, FLIP 0.0544853). As can be seen from the error metrics, the overall quality of our technique with denoising is significantly better than ray marching with and without jittering. The denoiser adds an additional 10ms of frame time and 400MB of memory consumption when enabled.

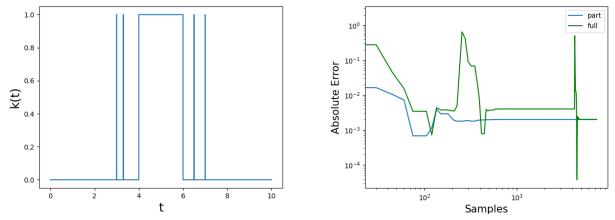

Figure 7: *We use an analytical function (a) to evaluate the convergence of the integration with our data structure. The absolute error (b) is shown for an increasing number of samples.*

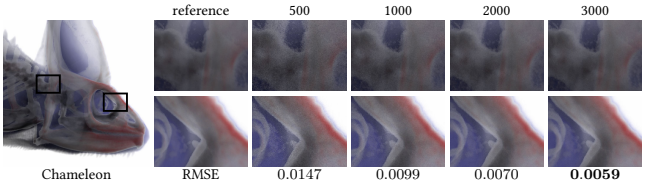

Figure 8: *Evaluation of the convergence depending on the number of samples per pixel using the* CHAMELEON *dataset. The render times are* 321.10*s for the reference, and* 10.80*s,* 22.76*s,* 45.36*s, and* 68*s for the other images; the RMSE compared to the reference is printed below the insets.*

6.3. Convergence of Data Structure and Rendering

To demonstrate the convergence behaviour of our data structure we define an analytical function $k(t)$ for which we can calculate the integral $\int k(s)^2 ds$. It has extremely narrow peaks (Fig. 7a) which are very difficult to sample and thus represents an extreme case for the data structure. To assess the convergence, we plot the absolute error of the approximated integral, i.e. $||\int k(s)^2 ds - \int \tilde{k}(s)^2 ds||$ over the entire domain in green and for one location $t = 5$ in blue in Fig. 7b. As we can see, the data structure converges to the true integral up to floating point precision. In the case of $t = 5$ it even matches the true integral exactly. As a next step, we evaluate the convergence of our whole technique in Fig. 8 using the CHAMELEON dataset with an increasing number of samples per pixel. As we can see, even for a low number of samples per pixel our algorithm produces results with low RMSE.

6.4. Comparison of Multiple Scattering

In Fig. 9 we show our extinction optimization for the single-scattering model, for multiple scattering where direct illumination is only optimized at the first scattering vertex (i.e. optimized single-scattering plus normal multiple scattering), and for multiple scattering where the direct illumination is optimized at every scattering event. As expected, the brightness significantly increases with multiple scattering (in practice one would obviously adjust the visualization parameters in the single-scattering case which is omitted for demonstration purposes here). One advantage of the full extinction optimization in this example is that the Atlas and Axis of the cervical vertebrae, which are occluded by the jaw (all bone structures are

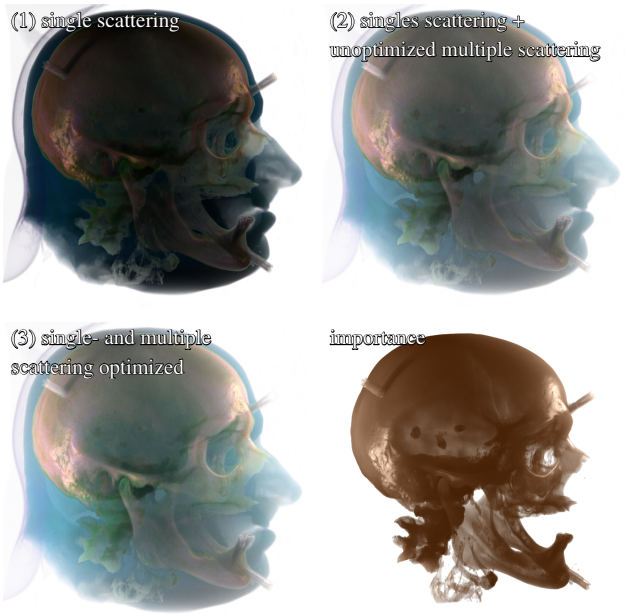

Figure 9: *Comparison between (1) single-scattering, (2) extinction-optimized single scattering plus multiple scattering, and (3) extinction optimization for both single and multiple scattering. We use* 2000 *samples per pixel and enable the Blinn-Phong BRDF with* $n = 40$*; the render times for converged images are (1)* 12.13*s, (2)* 72.16*s and (3)* 157.21*s.*

mapped to high importance), receive additional illumination and are better illuminated.

6.5. Evaluation of Parameters

Due to the fact that we changed the optimization function, we need to scale up the parameters q and r to produce a similar effect as Ament et al. [AZD17] (recall that in their work the parameters require scaling according to the step size, which is not the case in our formulation). We used $q = 70$ and $r = 30$ in our examples, and in Fig. 10 we showcase different configurations of r ranging from 0 to 70 to demonstrate its effect. As we can see with increasing r the volume behind the thin structures of the hip occludes the background less and adds illumination to the important region.

6.6. Importance Functions

To demonstrate the effectiveness of our post-interpolative importance function we show an extreme case where we set non-zero importance for the periosteum and endosteum of the foot bones, i.e. a very small region. As can be seen in Fig. 11 using a pre-interpolative importance function makes it impossible to discern the specified region accurately. As mentioned in Section 4.2, we can flexibly define our importance function, e.g. based on the volume data (extinction) and/or spatial attributes. All aforementioned results were obtained mapping extinction to importance. In Fig. 10 we show an importance function based on segmentation data: the right half of the pelvis of the human is mapped to an importance

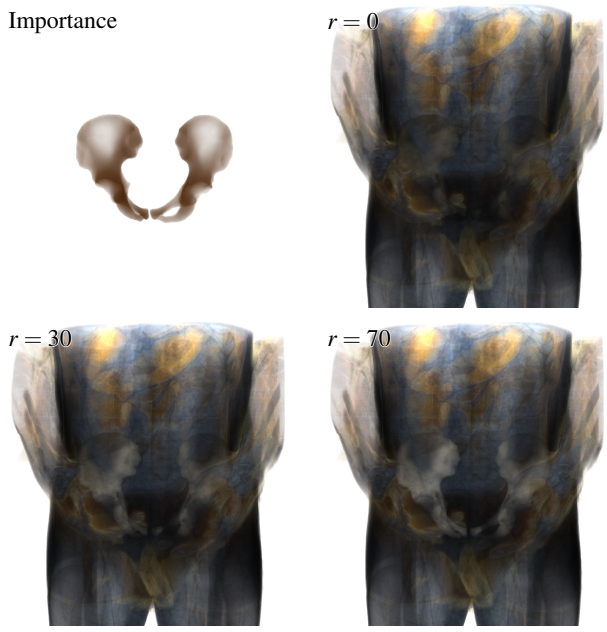

Figure 10: *We define the importance function to optimize visibility and illumination for the pelvis of a human (right half mapped to importance* 0.75*, left half to* 1*). We also show the impact of different values for the optimization parameters r. The images are rendered with* 2000 *samples per pixel and have a render time of* 47.5*s.*

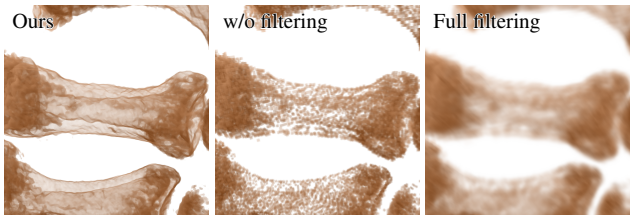

Figure 11: *Comparison between the post-interpolative and pre-interpolative importance function, with trilinear interpolation and a box filter, for the periosteum and endosteum of the foot bones.*

value of 0.75, the left half is mapped to 1. To this end, we generate a signed distance field from the pelvis in the segmentation data and when evaluating the importance function, we use the distance field and a threshold of 0.5 (similar to Green [Gre07]) as a boundary criterion for the important regions. We can also observe that having different importance values and especially importance values < 1 leads to inferior visibility in the visualization. This is often not desired and underpins our discussion in Section 4.2. This is also why we set the importance to only two discrete values, 1 and 0, in most of our examples.

6.7. Evaluation of Light Ray Optimization

In the previous examples we have demonstrated the extinction optimization for direct illumination. In contrast to camera rays, we cannot reuse the importance samples across progressive rendering steps, as the scattering locations along the camera ray, and thus

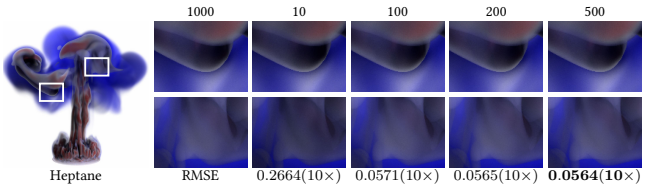

Figure 12: *Evaluation of the light ray extinction optimization using an increasing number of samples for the* HEPTANE *dataset. Every image is rendered with* 5000 *samples per pixel and the render times are:* 51.7*s for the reference, and* 21.47*s,* 27.17*s,* 28.2*s, and* 37 *for the varying number of samples. Below each inset we print the RMSE scaled by* 10*.*

the rays cast for computing the direct illumination, are determined stochastically. For this reason, we draw a fixed number of samples for each of these rays and build the data structure for efficient optimized transmittance computation from scratch. In Fig. 12 we vary the number of samples and can see that not many samples are required to approximate the integral faithfully. As a good trade off between accuracy and speed we chose 200 samples in all other visualizations.

6.8. Memory and Performance Analysis

We analyse the memory consumption and performance of our method using the HEPTANE and CHAMELEON datasets rendered at three different resolutions (800×800 like in the other experiments, 1600×1600 and 1920×1080). The memory consumption comprises of:

1. a static allocation for the light ray optimization which is independent of the dataset and resolution with a size of 156MB,
2. the per-pixel allocation for the sizes and offsets in the data structure, which only depends on the resolution,
3. the memory pool holding the data structure, depending on the number of segments for all pixels combined.

Table 3 shows the measured memory consumption for the per-pixel data and the memory pool for the data structures. As we can see, the memory consumption is dominated by the memory pool which depends only on the number of extracted segments. The frame time scales almost linearly with the resolution for the HEPTANE dataset. It is noticeably smaller at 1920×1080 because of the empty space added left and right of the volume (different aspect ratio). The CHAMELEON dataset scales better due to the increase in texture fetching latency which gets hidden better with the increased compute workload.

7. Conclusions

In this paper we have presented a method for importance-driven, view-dependent optimization of visibility and illumination for Monte Carlo volume visualization. Previously existing techniques relied on stepwise sampling along rays and thus DVR using ray marching, and thus were incompatible with free-flight distance sampling of state-of-the-art Monte Carlo methods. We defined a post-interpolative importance function, introduced an efficient data

Memory Usage (MB)	800^2	1600^2	1920×1080
HEPTANE	544	2752	2144
CHAMELEON	576	2912	1664
Per-Pixel data	14	50	40
Frame Time (ms)			
HEPTANE	47	175	101
CHAMELEON	177	467	396

Table 3: *Memory usage for storing the segments of the data structure, the per-pixel data, as well as the frame times with 10 samples per pixel for the* HEPTANE *and* CHAMELEON *datasets at three different resolutions.*

structure to sample, approximate and optimize the integrated extinction along rays, and devised a Monte Carlo estimator for interactive visualization. Our technique has only a moderate memory overhead and preserves all the beneficial properties of Monte Carlo rendering, such as progressive rendering or its flexibility regarding scattering models. We hope that our work spurs future research making the transfer of other perceptually-motivated visualization techniques from the ray marching domain into modern Monte Carlo-based systems possible.

Acknowledgement

This work was funded by KiKIT, the Pilot Program for Core-Informatics at the Karlsruhe Institute of Technology by the Helmholtz Association. The VISIBLE HUMAN dataset is courtesy of the U.S. National Library of Medicine. We thank the University of Utah Center for Simulation of Accidental Fires and Explosions for making the HEPTANE dataset available, Digital Morphology for the CHAMELEON, and Philips Research for the FOOT dataset. Open Access funding enabled and organized by Projekt DEAL.

References

[AD16] AMENT, MARCO and DACHSBACHER, CARSTEN. "Anisotropic Ambient Volume Shading". *IEEE Transactions on Visualization and Computer Graphics* 22.1 (2016), 1015–1024. DOI: 10.1109/TVCG.2015.2467963 2.

[AHH*22] ANDREASSEN, THOR E., HUME, DONALD R., HAMILTON, LANDON D., et al. *Visible Human Male*. 2022. DOI: 10.56902/COB.vh.2022.2 6.

[ANA*20] ANDERSSON, PONTUS, NILSSON, JIM, AKENINE-MÖLLER, TOMAS, et al. "ꟻLIP: A Difference Evaluator for Alternating Images". *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 3.2 (2020), 15:1–15:23. DOI: 10.1145/3406183 6.

[ASDW14] AMENT, MARCO, SADLO, FILIP, DACHSBACHER, CARSTEN, and WEISKOPF, DANIEL. "Low-Pass Filtered Volumetric Shadows". *IEEE Transactions on Visualization and Computer Graphics* 20.12 (2014), 2437–2446. DOI: 10.1109/TVCG.2014.2346333 2.

[AWD16] AMENT, MARCO, WEISKOPF, DANIEL, and DACHSBACHER, CARSTEN. "Ambient Volume Illumination". *Computing in Science & Engineering* 18.2 (2016), 90–97. DOI: 10.1109/MCSE.2016.23 2.

[AZD17] AMENT, MARCO, ZIRR, TOBIAS, and DACHSBACHER, CARSTEN. "Extinction-Optimized Volume Illumination". *IEEE Transactions on Visualization and Computer Graphics* 23.7 (2017), 1767–1781. DOI: 10.1109/TVCG.2016.2569080 2–4, 6–8.

[BG07] BRUCKNER, STEFAN and GRÖLLER, EDUARD. "Style Transfer Functions for Illustrative Volume Rendering". *Computer Graphics Forum* 26.3 (2007), 715–724. DOI: 10.1111/j.1467-8659.2007.01095.x 2.

[BGI*14] BALSA RODRÍGUEZ, MARCOS, GOBBETTI, ENRICO, IGLESIAS GUITIÁN, JOSÉ A., et al. "State-of-the-Art in Compressed GPU-Based Direct Volume Rendering". *Computer Graphics Forum* 33.6 (2014), 77–100. DOI: 10.1111/cgf.12280 1, 2.

[BHP15] BEYER, JOHANNA, HADWIGER, MARKUS, and PFISTER, HANSPETER. "State-of-the-Art in GPU-Based Large-Scale Volume Visualization". *Computer Graphics Forum* 34.8 (2015), 13–37. DOI: 10.1111/cgf.12605 1, 2.

[Bli77] BLINN, JAMES F. "Models of light reflection for computer synthesized pictures". *SIGGRAPH Comput. Graph.* 11.2 (1977), 192–198. ISSN: 0097-8930. DOI: 10.1145/965141.563893 3.

[Cha13] CHANDRASEKHAR, SUBRAHMANYAN. *Radiative Transfer*. Dover Books on Physics. Dover Publications, 2013. URL: https://books.google.de/books?id=1YHCAgAAQBAJ 2.

[CM11] CORREA, CARLOS D. and MA, KWAN-LIU. "Visibility Histograms and Visibility-Driven Transfer Functions". *IEEE Transactions on Visualization and Computer Graphics* 17.2 (2011), 192–204. DOI: 10.1109/TVCG.2010.35 2.

[EHK*06] ENGEL, KLAUS, HADWIGER, MARKUS, KNISS, JOE M., et al. *Real-time Volume Graphics*. USA: A. K. Peters, Ltd., 2006. ISBN: 1568812663 2.

[GLDL14] GUO, FAN, LI, HUI, DAUGHTON, WILLIAM, and LIU, YI-HSIN. "Formation of Hard Power Laws in the Energetic Particle Spectra Resulting from Relativistic Magnetic Reconnection". *Phys. Rev. Lett.* 113 (15 2014), 155005. DOI: 10.1103/PhysRevLett.113.155005 6.

[GMH*19] GEORGIEV, ILIYAN, MISSO, ZACKARY, HACHISUKA, TOSHIYA, et al. "Integral formulations of volumetric transmittance". *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 38.6 (2019). DOI: 10/dffn 2.

[GR23] GÖGGERLE, MATTHIAS and REIMS, NILS. *Scrambler Phone Telta P-171D-ATS (Inv. 2017-407)*. Zenodo, 2023. DOI: 10.5281/zenodo.8143282 6.

[Gre07] GREEN, CHRIS. "Improved alpha-tested magnification for vector textures and special effects". *ACM SIGGRAPH Courses*. 2007, 9–18. DOI: 10.1145/1281500.1281665 9.

[GRT13] GÜNTHER, TOBIAS, RÖSSL, CHRISTIAN, and THEISEL, HOLGER. "Opacity optimization for 3D line fields". *ACM Transaction on Graphics* 32.4 (2013). DOI: 10.1145/2461912.2461930 2.

[GRT14] GÜNTHER, TOBIAS, RÖSSL, CHRISTIAN, and THEISEL, HOLGER. "Hierarchical opacity optimization for sets of 3D line fields". *Computer Graphics Forum* 33.2 (2014), 507–516. DOI: 10.1111/cgf.12336 2.

[GSE*14] GÜNTHER, TOBIAS, SCHULZE, MAIK, ESTURO, JANICK MARTINEZ, et al. "Opacity Optimization for Surfaces". *Computer Graphics Forum* 33.3 (2014), 11–20. DOI: 10.1111/cgf.12357 2.

[HG24] HIMMLER, PAUL and GÜNTHER, TOBIAS. "Transmittance-based Extinction and Viewpoint Optimization". *Computer Graphics Forum (Proc. of Eurographics Conference on Visualization)* 43.3 (2024). DOI: 10.1111/cgf.15096 2.

[HG41] HENYEY, LOUIS G. and GREENSTEIN, JESSE L. "Diffuse radiation in the galaxy". *Astrophysical Journal* 93 (1941), 70–83 3.

[HLSR09] HADWIGER, MARKUS, LJUNG, PATRIC, SALAMA, CHRISTOF REZK, and ROPINSKI, TIMO. "Advanced illumination techniques for GPU-based volume raycasting". *ACM SIGGRAPH Courses*. 2009. DOI: 10.1145/1667239.1667241 2.

[IMM22] IGLESIAS-GUITIAN, JOSE A., MANE, PRAJITA, and MOON, BOCHANG. "Real-Time Denoising of Volumetric Path Tracing for Direct Volume Rendering". *IEEE Transactions on Visualization and Computer Graphics* 28.7 (2022), 2734–2747. DOI: 10.1109/TVCG.2020.3037680 2.

[IZM18] IGOUCHKINE, OLEG, ZHANG, YUBO, and MA, KWAN-LIU. "Multi-Material Volume Rendering with a Physically-Based Surface Reflection Model". *IEEE Transactions on Visualization and Computer Graphics* 24.12 (2018), 3147–3159. DOI: 10.1109/TVCG.2017.2784830 2.

[JSYR14] JÖNSSON, DANIEL, SUNDÉN, ERIK, YNNERMAN, ANDERS, and ROPINSKI, TIMO. "A Survey of Volumetric Illumination Techniques for Interactive Volume Rendering". *Computer Graphics Forum* 33 (1 2014), 27–51. DOI: 10.1111/cgf.12252 2.

[KPB12] KROES, THOMAS, POST, FRITS H., and BOTHA, CHARL P. "Exposure Render: An Interactive Photo-Realistic Volume Rendering Framework". *PLOS ONE* 7.7 (July 2012), 1–10. DOI: 10.1371/journal.pone.0038586 2.

[LKG*16] LJUNG, PATRIC, KRÜGER, JENS, GROLLER, EDUARD, et al. "State of the Art in Transfer Functions for Direct Volume Rendering". *Computer Graphics Forum* 35.3 (2016), 669–691. DOI: 10.1111/cgf.12934 1, 2.

[LR11] LINDEMANN, FLORIAN and ROPINSKI, TIMO. "About the Influence of Illumination Models on Image Comprehension in Direct Volume Rendering". *IEEE Transactions on Visualization and Computer Graphics* 17.12 (2011), 1922–1931. DOI: 10.1109/TVCG.2011.161 2.

[Max95] MAX, NELSON. "Optical models for direct volume rendering". *IEEE Transactions on Visualization and Computer Graphics* 1.2 (1995), 99–108. DOI: 10.1109/2945.468400 1, 2.

[MGJ19] MILLER, BAILEY, GEORGIEV, ILIYAN, and JAROSZ, WOJCIECH. "A null-scattering path integral formulation of light transport". *ACM Transactions on Graphics (Proc. of SIGGRAPH)* 38.4 (2019). DOI: 10/gf6rzb 2.

[MSG*23] MORRICAL, NATE, SAHISTAN, ALPER, GÜDÜKBAY, UĞUR, et al. "Quick Clusters: A GPU-Parallel Partitioning for Efficient Path Tracing of Unstructured Volumetric Grids". *IEEE Transactions on Visualization and Computer Graphics* 29.1 (2023), 537–547. DOI: 10.1109/TVCG.2022.3209418 2.

[NGHJ18] NOVÁK, JAN, GEORGIEV, ILIYAN, HANIKA, JOHANNES, and JAROSZ, WOJCIECH. "Monte Carlo methods for volumetric light transport simulation". *Computer Graphics Forum (Proc. of Eurographics – State of the Art Reports)* 37.2 (2018). DOI: 10/gd2jqq 2, 3, 6.

[NSJ14] NOVÁK, JAN, SELLE, ANDREW, and JAROSZ, WOJCIECH. "Residual Ratio Tracking for Estimating Attenuation in Participating Media". *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)* 33.6 (2014). DOI: 10/f6r2nq 6.

[PBC*16] PREIM, BERNHARD, BAER, ALEXANDRA, CUNNINGHAM, DOUGLAS, et al. "A Survey of Perceptually Motivated 3D Visualization of Medical Image Data". *Computer Graphics Forum* (2016). DOI: 10.1111/cgf.12927 2.

[Sal07] SALAMA, CHRISTOF REZK. "GPU-Based Monte-Carlo Volume Raycasting". *Pacific Conference on Computer Graphics and Applications*. 2007, 411–414. DOI: 10.1109/PG.2007.27 2.

[SZD*23] SARTON, JONATHAN, ZELLMANN, STEFAN, DEMIRCI, SERKAN, et al. "State-of-the-art in Large-Scale Volume Visualization Beyond Structured Data". *Computer Graphics Forum (EuroVis STAR)* 42.3 (2023), 491–515. DOI: 10.1111/cgf.14857 1, 2.

[VKG05] VIOLA, IVAN, KANITSAR, ARMIN, and GRÖLLER, EDUARD. "Importance-driven feature enhancement in volume visualization". *IEEE Transactions on Visualization and Computer Graphics* 11.4 (2005), 408–418. DOI: 10.1109/TVCG.2005.62 2.

[WMHL65] WOODCOCK, E. R., MURPHY, T., HEMMINGS, P. J., and LONGWORTH, T. C. "Techniques Used in the GEM Code for Monte Carlo Neutronics Calculations in Reactors and Other Systems of Complex Geometry". *Applications of Computing Methods to Reactor Problems*. Argonne National Laboratory. 1965 3.

[WMZ22] WALD, INGO, MORRICAL, NATE, and ZELLMANN, STEFAN. "A Memory Efficient Encoding for Ray Tracing Large Unstructured Data". *IEEE Transactions on Visualization and Computer Graphics* 28.1 (2022), 583–592. DOI: 10.1109/TVCG.2021.3114869 2.

[XCC*24] XU, CHUNXIAO, CHENG, HAOJIE, CHEN, ZHENXIN, et al. "Real-time Realistic Volume Rendering of Consistently High Quality with Dynamic Illumination". *IEEE Transactions on Visualization and Computer Graphics* (2024), 1–15. DOI: 10.1109/TVCG.2024.3445339 2.

[XSL21] XU, CHAOQING, SUN, GUODAO, and LIANG, RONGHUA. "A survey of volume visualization techniques for feature enhancement". *Visual Informatics* 5.3 (2021), 70–81. DOI: 10.1016/j.visinf.2021.08.001 1, 2.

[ZRPD20] ZEIDAN, MAHMOUD, RAPP, TOBIAS, PETERS, CHRISTOPH, and DACHSBACHER, CARSTEN. "Moment-Based Opacity Optimization". *Eurographics Symposium on Parallel Graphics and Visualization*. 2020. DOI: 10.2312/pgv.20201072 2.

[ZTL*15] ZHOU, ZHIGUANG, TAO, YUBO, LIN, HAI, et al. "Occlusion-free feature exploration for volume visualization". *Multimedia Tools and Applications* 74 (2015), 10243–10258. DOI: 10.1007/s11042-014-2162-4 2.

[ZWS*24] ZELLMANN, STEFAN, WU, QI, SAHISTAN, ALPER, et al. "Beyond ExaBricks: GPU Volume Path Tracing of AMR Data". *Computer Graphics Forum* (2024). DOI: 10.1111/cgf.15095 2.

Appendix

$$E(\mu'_*(\mathbf{y})) = p(\mu'_*(\mathbf{y}) - \mu_*(\mathbf{y}))^2 + \mu'_*(\mathbf{y})^2(1-g(\mathbf{y}))^{2\lambda} \cdot \left(r\int_{\mathbf{x}}^{\mathbf{y}} g(\mathbf{i})^2\,d\mathbf{i} + q\int_{\mathbf{y}}^{\mathbf{z}} g(\mathbf{i})^2\,d\mathbf{i}\right). \tag{12}$$

Expand the first term and substitute integrals with Eq. (14):

$$E(\mu'_*(\mathbf{y})) = p\mu'_*(\mathbf{y})^2 - 2p\mu'_*(\mathbf{y})\mu_*(\mathbf{y}) + p\mu_*(\mathbf{y})^2 + \mu'_*(\mathbf{y})^2(1-g(\mathbf{y}))^{2\lambda}h(\mathbf{y}), \tag{13}$$

$$\begin{aligned} \text{with } h(\mathbf{y}) &= r\int_{\mathbf{x}}^{\mathbf{y}} g(\mathbf{i})^2\,d\mathbf{i} + q\int_{\mathbf{y}}^{\mathbf{z}} g(\mathbf{i})^2\,d\mathbf{i} \\ &= r\int_{\mathbf{x}}^{\mathbf{y}} g(\mathbf{i})^2\,d\mathbf{i} + q\left(\int_{\mathbf{x}}^{\mathbf{z}} g(\mathbf{i})^2\,d\mathbf{i} - \int_{\mathbf{x}}^{\mathbf{y}} g(\mathbf{i})^2\,d\mathbf{i}\right) \\ &= r\int_{\mathbf{x}}^{\mathbf{y}} g(\mathbf{i})^2\,d\mathbf{i} + q\int_{\mathbf{x}}^{\mathbf{z}} g(\mathbf{i})^2\,d\mathbf{i} - q\int_{\mathbf{x}}^{\mathbf{y}} g(\mathbf{i})^2\,d\mathbf{i} \\ &= (r-q)\int_{\mathbf{x}}^{\mathbf{y}} g(\mathbf{i})^2\,d\mathbf{i} + q\int_{\mathbf{x}}^{\mathbf{z}} g(\mathbf{i})^2\,d\mathbf{i}. \end{aligned} \tag{14}$$

Factor out $\mu'_*(\mathbf{y})^2$:

$$E(\mu'_*(\mathbf{y})) = \mu'_*(\mathbf{y})^2\left(p + (1-g(\mathbf{y}))^{2\lambda}h(\mathbf{y})\right) - 2p\mu'_*(\mathbf{y})\mu_*(\mathbf{y}) + p\mu_*(\mathbf{y})^2. \tag{15}$$

Compute critical point by setting derivative to 0:

$$\frac{dE}{d\mu'_*(\mathbf{y})} = 2\mu'_*(\mathbf{y})\left(p + (1-g(\mathbf{y}))^{2\lambda}h(\mathbf{y})\right) - 2p\mu_*(\mathbf{y}) = 0. \tag{16}$$

Solving for $\mu'_*(\mathbf{y})$:

$$\mu'_*(\mathbf{y}) = \frac{p\mu_*(\mathbf{y})}{p + (1-g(\mathbf{y}))^{2\lambda}h(\mathbf{y})}. \tag{17}$$

We can classify the critical point as the minimum due to the fact that the second derivative is always > 0 for the value range of our parameters (it cannot be 0 as this would lead to $\mu'_*(\mathbf{y}) = \frac{0}{0}$ which is undefined):

$$\frac{d^2E}{d\mu'_*(\mathbf{y})^2} = 2\left(p + (1-g(\mathbf{y}))^{2\lambda}h(\mathbf{y})\right) > 0. \tag{18}$$