Don’t Splat your Gaussians: Volumetric Ray-Traced Primitives for Modeling and Rendering Scattering and Emissive Media

JORGE CONDOR, Meta Reality Labs, Switzerland and USI Lugano, Switzerland
SÉBASTIEN SPEIERER, Meta Reality Labs, Switzerland
LUKAS BODE, Meta Reality Labs, Switzerland
ALJAŽ BOŽIČ, Meta Reality Labs, Switzerland
SIMON GREEN, Meta Reality Labs, United Kingdom
PIOTR DIDYK, USI Lugano, Switzerland
ADRIÁN JARABO, Meta Reality Labs, Spain

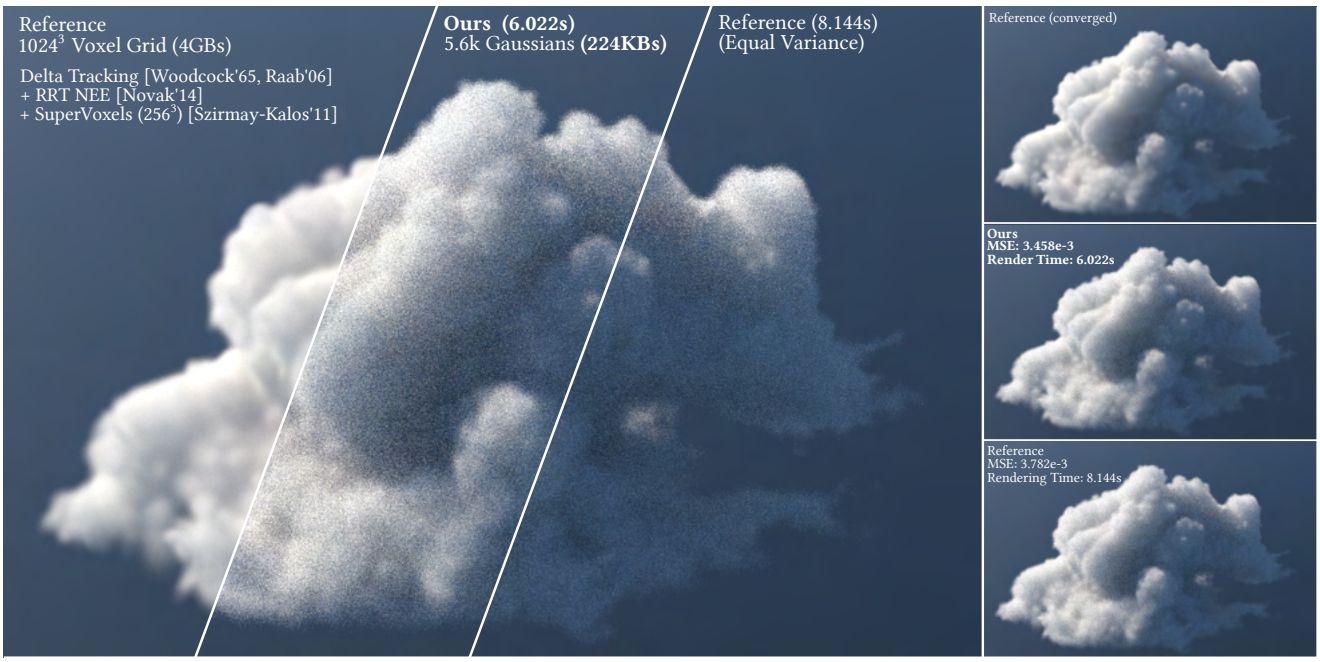


Fig. 1. We represent a complex volumetric cloud using traditional grid-based methods (left and right, 1024^3 voxel grid resolution, 4GBs) and our primitives-based representation using Gaussian kernels (middle, 5.6k primitives, 224KBs), and render it with volumetric path tracing. Our method achieves substantial speedups thanks to the analytical transmission estimation and sampling, our efficient rendering approach and its extremely compact representation. When compared to the original asset, at a potential cost of detail (Figure 7), we provide large performance and memory compression benefits. Asset is part of the Walt Disney Animation Studios *cloud* dataset (CC-BY-SA 3.0). Rendering times reported on a NVIDIA A6000.

Authors’ addresses: Jorge Condor, jorge.condor@usi.ch, Meta Reality Labs, Giesshübelstrasse 30, Zurich, Zurich, Switzerland, 8045 and USI Lugano, Via Giuseppe Buffi 13, Lugano, Ticino, Switzerland, 6900; Sébastien Speierer, speierers@meta.com, Meta Reality Labs, Giesshübelstrasse 30, Zurich, Switzerland, 8045; Lukas Bode, lbode@meta.com, Meta Reality Labs, Giesshübelstrasse 30, Zurich, Switzerland, 8045; Aljaž Božič, aljaz@meta.com, Meta Reality Labs, Giesshübelstrasse 30, Zurich, Switzerland, 8045; Simon Green, simongreen@meta.com, Meta Reality Labs, London, United Kingdom, 8045; Piotr Didyk, piotr.didyk@usi.ch, USI Lugano, Via Giuseppe Buffi 13, Lugano, Switzerland, 6900; Adrián Jarabo, ajarabo@meta.com, Meta Reality Labs, , Zaragoza, Spain.

ACM 0730-0301/2025/1-ART1
https://doi.org/10.1145/3711853

Efficient scene representations are essential for many computer graphics applications. A general unified representation that can handle both surfaces and volumes simultaneously, remains a research challenge. In this work we propose a compact and efficient alternative to existing volumetric representations for rendering such as voxel grids. Inspired by recent methods for scene reconstruction that leverage mixtures of 3D Gaussians to model radiance fields, we formalize and generalize the modeling of scattering and emissive media using mixtures of simple kernel-based volumetric primitives. We introduce closed-form solutions for transmittance and free-flight distance sampling for different kernels, and propose several optimizations to use our method efficiently within any off-the-shelf volumetric path tracer. We demonstrate our method in both forward and inverse rendering of complex scattering media. Furthermore, we adapt and showcase our method in radiance field optimization and rendering, providing additional flexibility

compared to current state of the art given its ray-tracing formulation. We also introduce the Epanechnikov kernel and demonstrate its potential as an efficient alternative to the traditionally-used Gaussian kernel in scene reconstruction tasks. The versatility and physically-based nature of our approach allows us to go beyond radiance fields and bring to kernel-based modeling and rendering any path-tracing enabled functionality such as scattering, relighting and complex camera models.

CCS Concepts: • **Computing methodologies** → **Shape representations**; **Ray tracing**; **Volumetric models**; **Point-based models**; **Reconstruction**.

Additional Key Words and Phrases: Volume Rendering, Scattering, Radiance Fields, 3D Reconstruction, Volumetric Primitives, Volumetric Representations, Ray Tracing, Inverse Rendering

ACM Reference Format:
Jorge Condor, Sébastien Speierer, Lukas Bode, Aljaž Božič, Simon Green, Piotr Didyk, and Adrián Jarabo. 2025. Don't Splat your Gaussians: Volumetric Ray-Traced Primitives for Modeling and Rendering Scattering and Emissive Media. *ACM Trans. Graph.* 1, 1, Article 1 (January 2025), 18 pages. https://doi.org/10.1145/3711853

1 INTRODUCTION

Volumetric representations of appearance have found good success at representing complex appearances such as cloth and hair [Aliaga et al. 2017; Khungurn et al. 2015; Schröder et al. 2011; Zhao et al. 2011], trees [Loubet and Neyret 2017; Neyret 1998], clouds and smoke [Kallweit et al. 2017], or particle aggregates [Meng et al. 2015; Moon et al. 2007; Müller et al. 2016], to name a few. In this work we propose a compact volumetric representation of appearance, that allows efficient physics-based rendering of both scattering and emissive media.

Most previous approaches for modeling volumetric appearances leverage grid-like representations of appearance such as voxels-grids, potentially with a multilevel underlying hierarchy, for which highly efficiently libraries exist [Museth 2013]. Voxel grids are standard in rendering, and used in most research [Jakob 2010; Jakob et al. 2022b; Pharr et al. 2023] and production renderers [Burley et al. 2018; Christensen et al. 2018; Fascione et al. 2018; Georgiev et al. 2018]. Voxel-grids are flexible, provide $O(1)$ query time, and spatial connectivity for e.g., downsampling and level-of-detail representations [Heitz et al. 2015; Loubet and Neyret 2017]. Unfortunately, they scale poorly in terms of memory, specially when representing very fine detail and sparse structures. While hierarchical grid-based methods [Fong et al. 2017] have proven very effective at subdividing complex heterogeneous models into simpler local components, their building blocks are still grid-based, essentially inheriting the same problems. For an efficient integration of e.g., transmittance, voxel grids require the use of stochastic tracking techniques [Novák et al. 2018], which introduces additional variance when rendering media.

In parallel, volume rendering techniques and continuous volumetric representations have recently seen unprecedented interest in the fields of computer vision and image-based graphics, spearheaded by efforts such as neural radiance fields [Mildenhall et al. 2020]. These methods allow to capture and render photorealistic three-dimensional scenes, by optimizing an underlying volumetric representation of matter. Different volumetric representations have been proposed, including point-based methods [Kerbl et al. 2023; Sainz and Pajarola 2004], implicit neural models [Barron et al. 2021, 2022; Mildenhall et al. 2020], explicit voxel-based representations [Fridovich-Keil et al. 2022; Yu et al. 2021], and hybrid approaches [Lombardi et al. 2021; Müller et al. 2022; Xu et al. 2022]. While implicit neural models generally achieve the highest compression rates, point-based methods can be considered as the state of the art in terms of image quality and rendering speed for radiance fields rendering, while still being substantially more compact than voxel-grid structures. However, while their derivation starts from physics (i.e., the volume rendering equation), these methods in general are not suitable for their use within a physics-based renderer, given the fairly large assumptions and simplifications imposed in their image formation model to favour speed and ease of optimization.

Inspired by the success of recent 3D Gaussian-based representations of radiance fields [Kerbl et al. 2023], we propose a new representation for general scattering and emissive media based on mixtures of three-dimensional kernel-based volumetric primitives. Each primitive statistically represents a spatial distribution of matter with the same optical properties. We introduce this representation inside the radiative transfer theory (RTT) [Chandrasekhar 1960] and derive closed-form solutions for transmittance and emission without the need of stochastic sampling, as well as sampling routines for computing inscattering.

While the scene representation is similar to the Gaussian splatting technique [Kerbl et al. 2023], the image formation model is fundamentally different: Kerbl et al. assume that each primitive becomes a billboard oriented towards the camera and therefore the density along the ray becomes a sum of delta functions; this allows them to implement their model in an extremely efficient rasterization pipeline. In contrast, our radiative primitive-based formulation preserves the primitive's three-dimensional density and integrates following the physical transport process. This is crucial for physics-based rendering since it allows preserving reciprocity, while solving some view-dependent primitive ordering problems in Gaussian splatting. Our theoretical framework is general and supports a wide range of three-dimensional kernels. We demonstrate our work using the Gaussian kernel employed by Kerbl et al. [2023], but also on 3D Epanechnikov kernels, implementing closed-form solutions for volumetric transmittance, emission, and free-flight sampling. We implement our work by leveraging hardware-accelerated ray tracing for finding the primitives' bounds and integrating their contribution along the ray. This fits well inside modern ray-tracing-based renderers, allowing us to compute transmittance and emission for both primary and secondary rays, needed for multiple scattering. We also derive the adjoint of our image formation model, which we demonstrate in inverse rendering applications. We showcase our primitive-based volumetric model through several applications both in forward and inverse rendering: 1) traditional forward rendering of scattering media, where we demonstrate improved performance over voxel-grids using local aggregated statistics and state of the art transmittance estimators; 2) inverse rendering of scattering and purely absorptive media; 3) inverse tomographic reconstructions from focus stacks using telecentric cameras and 4) radiance field optimization and rendering of complex scenes (both real and synthetic), with increased generality and controllability. Furthermore, our integration into the general RTT allows for future extensions in relighting and scattering. In short, our **contributions** are:

- a novel kernel-based primitive representation for volumetric media that fits into the radiative transfer framework and can be integrated in any physics-based rendering engine;
- closed-form expressions for transmittance and emission, as well as distance sampling routines proportional to transmittance, for 3D Gaussian and Epanechnikov kernels;
- an efficient ray tracing-based implementation for solving light transport using our novel volumetric representation for both scattering and emissive (radiance field) media;
- and adjoint derivatives of our forward methods for solving inverse reconstruction problems efficiently.

2 RELATED WORK

Volumetric light transport. Simulating light transport in media has been thoroughly studied in computer graphics [Novák et al. 2018]. It involves solving the radiative transfer equation (RTE) [Chandrasekhar 1960], which has been extended to account for anisotropic [Jakob et al. 2010], refractive [Ament et al. 2014], or spatially correlated [Bitterli et al. 2018; Jarabo et al. 2018] media. These generalizations make it possible to render a wider range of light transport phenomena.

Numerical estimation of transmittance is at the core of modern volumetric light transport simulation [Novák et al. 2018]; via biased quadrature rules (ray-marching) [Muñoz 2014; Tuy and Tuy 1984] and modern unbiased variants [Kettunen et al. 2021], to stochastic tracking algorithms (null-scattering estimators) [Woodcock 1965], including extensions with variance reduction via control variates [Crespo et al. 2021; Novák et al. 2014; Szirmay-Kalos et al. 2017], alternative formulations based on power-series [Georgiev et al. 2019], or differentiable formulations targeting inverse rendering [Nimier-David et al. 2022]. All these approaches require tight estimates of the maximum density (*majorant*) for performance, either obtained in precomputation [Szirmay-Kalos et al. 2011; Yue et al. 2010] or estimated on-the-fly [Carter et al. 1972; Galtier et al. 2013; Kutz et al. 2017; Misso et al. 2023]. Our approach, on the other hand, allows to compute transmittance in closed-form, as the product of transmittance of all primitives along the ray.

Volumetric representations of matter. For heterogeneous media, discrete voxel-grid approaches are ubiquitously found in most applications due to their simplicity and flexibility. They can feature multi-level hierarchical structures [Museth 2013, 2021], which are helpful for faster traversal and filtering for level-of-detail applications, but suffer from poor scalability in terms of memory consumption when the modeled medium possesses finer details. Sparser representations like collections of isotropic and simple volumes have been used in other fields, e.g., particle physics [Brown and Martin 2003], where they can model neutron transport in graphite pebble-bed reactors. The advantage of having collections of these simple volumes is that, individually, they offer closed-form solutions for transmittance estimation and sampling, reducing variance when compared with most tracking methods normally used in voxel grids. However, searching for the boundaries of these volumes can be inefficient and slow in many situations [Bitterli et al. 2018]. In graphics, we can find hierarchical grid-based approaches [Fong et al. 2017] for subdividing complex, potentially sparse, volumes into smaller, more compact representations, though eventually they rely on smaller voxel-based primitives. Alternatively, implicit representations of participating media have been proposed, most notably in the context of radiance fields but also in compression of large volumes, including large multilayer perceptrons (MLP) [Kim et al. 2024; Mildenhall et al. 2020] or sparse hash grids of features combined with tiny MLPs [Müller et al. 2022]. These approaches result in large compression rates, at the cost of expensive queries. Closer to our work, Knoll et al. [2021] proposed to model emissive and absorbing media using isotropic Gaussian mixture models, in the context of particle-based volumes [Max 1979]; however, they did not leverage closed-forms expressions for transmittance, requiring stochastic integration via tracking, and cannot represent scattering media. In contrast, our work proposes closed-form solutions for transmittance and sampling, and further extends to support scattering media, anisotropic kernels, and efficient inverse rendering.

Inverse volumetric rendering. Early works on inverse rendering of heterogeneous media used inverse volumetric rendering for matching micron-scale cloth patches to measures [Khungurn et al. 2015], approximating multiscale volumetric representations [Zhao et al. 2016], scattering compensation for 3D printing [Condor et al. 2023; Elek et al. 2017; Nindel et al. 2021; Sumin et al. 2019], or inverse scattering [Gkioulekas et al. 2016] by using hand-made derivatives of light transport. Zhang et al. [2019] proposed a differential radiative transfer framework suitable for inverse volumetric rendering. These works use voxel grids for representing media density, though our proposed representation would fit in these inverse pipelines.

Radiance fields. Neural Radiance Fields (NeRF) [Mildenhall et al. 2020] introduced a continuous, implicit volumetric representation based on multi-layer perceptrons (MLPs). Subsequent works have leveraged multi-scale representations [Barron et al. 2021] and extended to unbounded scenes [Barron et al. 2022; Reiser et al. 2023; Zhang et al. 2020]; however, training and evaluating the MLPs still requires significant computation. In contrast, Instant-NGP [Müller et al. 2022] relies on a sparse hash grid of features and a small MLP, enabling faster scene reconstruction. Sparsification and hierarchization has been a popular avenue for optimizing radiance field fitting and rendering [Božič et al. 2022; Chen et al. 2023; Hedman et al. 2021; Lombardi et al. 2021; Yu et al. 2021], achieving real-time rendering even on mobile devices, or helping on accelerating off-line rendering with complex assets [Condor and Jarabo 2022; Zhu et al. 2021]. Most recently, 3D Gaussian Splatting [Kerbl et al. 2023] removes the need for MLPs altogether, relying instead on an explicit Gaussian-based representation. Combined with an efficient splatting technique for rasterization [Zwicker et al. 2001], their approach is able to achieve fast rendering performance and high-quality reconstruction, allocating more primitives in areas where finer detail is needed. Our work takes Kerbl's approach, and extends it to general scattering media in the context of the radiative transfer framework.

Gaussians in physically-based rendering. Gaussian mixture models defined over the sphere have been extensively used in graphics for fitting BRDFs or environment maps [Xu et al. 2013], precomputed radiance transfer [Green et al. 2006], or on-line path guiding [Vorba et al. 2014]. Yan et al. [2016] fitted a 4D Gaussian mixture model for

accelerating computation when rendering glinty surfaces. Jakob et al. [2011] proposed to fit a 3D anisotropic Gaussian mixture model to the distribution of radiance in media, showing accelerated rendering using photon-based techniques. Despite their work sharing some similarities to ours (closed-form integration along Gaussians), it focuses on fitting radiance in the context of radiance estimation, while our approach focuses on modeling matter (density) and can be used in arbitrary volumetric renderers. Combining both representations (kernel-based representations for both media and radiance) would be an interesting line of future work.

3 BACKGROUND

Under the geometric optics assumption, light transport in participating media is governed by the radiative transfer equation (RTE) [Chandrasekhar 1960]. We compute the incident radiance $L(\mathbf{x}, \omega)$ at point $\mathbf{x}$ and direction ω as

$$(\omega \cdot \nabla)L(\mathbf{x}, \omega) = -\mu_a(\mathbf{x})L(\mathbf{x}, \omega) - \mu_s(\mathbf{x})L(\mathbf{x}, \omega) + \mu_a(\mathbf{x})Q(\mathbf{x}, \omega) + \mu_s(\mathbf{x})S(\mathbf{x}, \omega), \tag{1}$$

where S is the in-scattering term, defined as

$$S(\mathbf{x}, \omega) = \int_{\mathcal{S}^2} f_\mathrm{p}(\mathbf{x}, \omega' \to \omega)L(\mathbf{x}, \omega')\,\mathrm{d}\omega', \tag{2}$$

that models the light scattered in direction ω as a function of the integral of the light incoming from all directions on the unit sphere $\mathcal{S}^2$. The interaction of light and matter is characterized by the optical properties of the medium, described in terms of the differential probabilities of absorption and scattering (or coefficients) $\mu_a(\mathbf{x})$ and $\mu_s(\mathbf{x})$ respectively, the extinction coefficient $\mu_t(\mathbf{x}) = \mu_a(\mathbf{x}) + \mu_s(\mathbf{x})$, the phase function $f_\mathrm{p}(\mathbf{x}, \omega' \to \omega)$, and the emission $Q(\mathbf{x}, \omega)$. Assuming uncorrelated statistics of matter, the differential probability of extinction is defined as the product between the density $\rho(\mathbf{x})$ of the particles forming the medium at point $\mathbf{x}$ [m^{-3}] and the particles' cross section $\sigma(\mathbf{x})$ [m^2] as $\mu_t(\mathbf{x}) = \rho(\mathbf{x})\,\sigma(\mathbf{x})$. For simplicity, we ignore the wavelength dependency in Equation (1), assume scalar elastic light-matter interactions, and omit the potential anisotropy of the medium [Jakob et al. 2010]; generalizing the following derivations to include these additional dimensions is trivial. By integrating both sides of the differential RTE (1) along the direction ω we obtain the volumetric rendering equation, that models radiance $L(\mathbf{x}, \omega)$ as

$$L(\mathbf{x}_0, \omega) = \int_0^s \mathrm{T}(\mathbf{x}_0, \mathbf{x}_t)\,\mu_t(\mathbf{x}_t)\,L_o(\mathbf{x}_t, \omega)\,\mathrm{d}t, + \mathrm{T}(\mathbf{x}_0, \mathbf{x}_s)L(\mathbf{x}_s, \omega) \quad \text{with} \tag{3}$$

$$L_o(\mathbf{x}, \omega) = (1 - \alpha(\mathbf{x}))\,Q(\mathbf{x}, \omega) + \alpha(\mathbf{x})S(\mathbf{x}, \omega), \tag{4}$$

with $\mathbf{x}_t = \mathbf{x}_0 - t\omega$ and distance t, $\alpha(\mathbf{x}) = \mu_s(\mathbf{x})/\mu_t(\mathbf{x})$ as the single scattering albedo, and $\mathrm{T}(\mathbf{x}_0, \mathbf{x}_t)$ is the transmittance defined as the fractional visibility due to absorption and out-scattering in the medium, and modeled following the Beer-Lambert law as

$$\mathrm{T}(\mathbf{x}_0, \mathbf{x}_t) = \exp\left(-\int_0^t \mu_t(\mathbf{x}_{t'})\mathrm{d}t'\right). \tag{5}$$

Finally, $L(\mathbf{x}_s, \omega)$ is the outgoing radiance at the medium boundary $\mathbf{x}_s$ at distance s modeled by the rendering equation [Kajiya 1986].

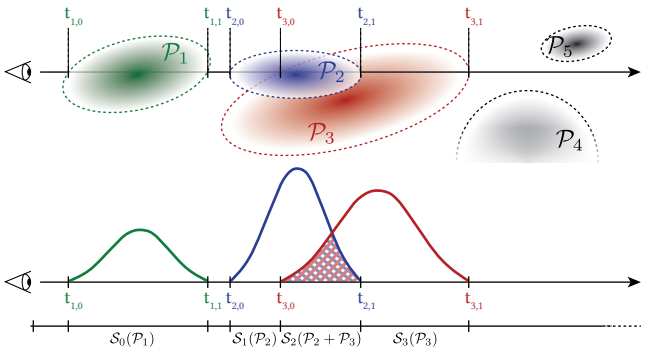


Fig. 2. **Top:** Flatland representation of a medium modeled using $N_\mathrm{total} = 5$ primitives $\mathcal{P}_i$ with $i \in [1, 5]$. For the depicted ray from the camera only three primitives contribute directly to the ray. **Bottom:** The primitives are projected in 1D along the ray, and their density, emission and in-scattering are integrated over that line, which can be separated in disjoint segments defined by the boundaries of the primitives. The contribution of each segment $\mathcal{S}_k$ is the integral over the primitives overlapping on that segment.

4 VOLUMETRIC PRIMITIVES

Inspired by recent works using Gaussian primitives for reconstructing radiance fields [Kerbl et al. 2023], and following a similar medium definition as Knoll et al. [2021], we model media using sets of primitives. Each primitive $\mathcal{P}_i$ represents a statistical aggregate of matter with identical emission, cross section and phase function, and with density $\rho_i(\mathbf{x})$ defined by a three-dimensional un-normalized kernel $K_i(\mathbf{x})$. Given these primitives, we can model the distribution of matter, and therefore the extinction probability as a mixture of these volumetric primitives following

$$\mu_t(\mathbf{x}) = \sum_{i=1}^{N} \sigma_i\,K_i(\mathbf{x}), \tag{6}$$

with N the number of primitives affecting $\mathbf{x}$, and σ_i their cross-section. For kernels with infinite support, N is the total number of primitives $N_\mathrm{total} = N$. In our case, we assume that all kernels have limited support, so typically $N < N_\mathrm{total}$. We compute the single-scattering albedo and phase function at $\mathbf{x}$ analogously, with the key difference of requiring normalization (see Appendix A).

4.1 The RTE with volumetric primitives

Defining matter as a mixture of volumetric primitives with finite support allow us to reinterpret Equation (3) as the summation of the contribution of all the primitives. For that, as shown in Figure 2 (bottom), we split the ray in segments based on the primitives' entry and exit points; then, we integrate radiance along segments in a front-to-back pass, keeping track of the multiplicative transmittance which is computed analytically. This approach is similar to how production renderers deal with complex overlapping media [Fong et al. 2017, Ch.6]; the key difference is that our approach approximates the media with primitives, which allows having closed-form transmittance and sampling expressions.

More formally, the boundaries of each primitive $t_{i,0}$ and $t_{i,1}$ subdivide the ray into ordered segments $\mathcal{S}_k = [\widehat{t}_{k,0}, \widehat{t}_{k,1}]$, with $\widehat{t}_{k,0} =$

$\widehat{t}_{k-1,1}$ for $k > 0$, with $\widehat{t}_{0,0} = \mathbf{x}$. Each of these segments $\mathcal{S}_k$ might overlap with zero, one, or multiple primitives denoted with the set $\{\mathcal{P}_i | i \in S_k\}$, where S_k is the per-segment set of indices, with $|S_k| \leq N$ the number of primitives overlapping in segment $\mathcal{S}_k$. With that definition of segments along the ray, we rewrite Equation (3) (omitting the boundary condition at $\mathbf{x}_s$ for simplicity, as

$$L(\mathbf{x}_0, \omega) = \sum_{k=1}^{M} \mathrm{T}_{k-1}(\mathbf{x}_0, \mathbf{x}_t) L_k(\mathbf{x}_{\widehat{t}_{k,0}}, \omega), \tag{7}$$

where M is the number of segments along the ray, $L_k(\mathbf{x}_{\widehat{t}_{k,0}}, \omega)$ the outgoing radiance at the segment $\mathcal{S}_k$ defined as

$$L_k(\mathbf{x}_{\widehat{t}_{k,0}}, \omega) = \int_{\widehat{t}_{k,0}}^{\widehat{t}_{k,1}} \mathrm{T}(\mathbf{x}_{\widehat{t}_{k,0}}, \mathbf{x}_t) \sum_{i \in S_k} \left[\sigma_i K_i(\mathbf{x}_t) L_{o,i}(\mathbf{x}_t, \omega) \right] \mathrm{d}t, \tag{8}$$

with $L_{o,i}(\mathbf{x}, \omega)$ the outgoing radiance from primitive $\mathcal{P}_i$, and $\mathrm{T}_{k-1}(\mathbf{x}_0, \mathbf{x}_t)$ is the transmittance defined as a recursive operator

$$\mathrm{T}_k(\mathbf{x}_0, \mathbf{x}_t) = \mathrm{T}_{k-1}(\mathbf{x}_0, \mathbf{x}_{\widehat{t}_{k-1,1}}) \, \mathrm{T}(\mathbf{x}_{\widehat{t}_{k,0}}, \mathbf{x}_t), \tag{9}$$

with $\mathrm{T}_0(\mathbf{x}_0, \mathbf{x}_t) = 1$ and $\mathrm{T}_M(\mathbf{x}_0, \mathbf{x}_t) = \mathrm{T}(\mathbf{x}_0, \mathbf{x}_t)$. Trivially, transmittance in segment k is computed as the product of transmittance of all primitives overlapping the segment

$$\mathrm{T}(\mathbf{x}_{\widehat{t}_{k,0}}, \mathbf{x}_t) = \exp\left(- \sum_{i \in S_k} \tau_i \left(\mathbf{x}_{\widehat{t}_{k,0}}, \mathbf{x}_{\min(t, \widehat{t}_{k,1})} \right) \right), \tag{10}$$

with $\tau_i(\mathbf{x}_a, \mathbf{x}_b)$ the optical depth from primitive $\mathcal{P}_i$ in the range $t \in [a, b]$ defined as

$$\tau_i(\mathbf{x}_a, \mathbf{x}_b) = \sigma_i \int_{\max(a, t_{i,0})}^{\min(b, t_{i,1})} K_i(\mathbf{x}_{t'}) \, \mathrm{d}t'. \tag{11}$$

By choosing the appropriate primitive kernels, we show in Section 5.1 that this integral can be computed in closed form. A more detailed derivation can be found in the Supplemental material.

4.2 Solving Equation 7 for distance sampling

While Equation (7) is just a summation over segments that can be computed analytically (we show examples later in Section 7), in the most general case $L_{o,i}(\mathbf{x}_t, \omega)$ hides the inscattering integral, which requires numerical evaluation using Monte Carlo sampling. Computing a Monte Carlo estimate would quickly become impractical; thus we need to sample a single segment per ray, which requires essentially sampling the distance t along the ray, ideally with probability distribution function (PDF) $p(t) = \mu_t(\mathbf{x}_t)\mathrm{T}(\mathbf{x}_0, \mathbf{x}_t)$.

While our primitive-based media allows the use of delta tracking (or any other Monte Carlo-based) estimators for distance sampling, we found that we can leverage the recursive formulation of transmittance (9) and pose the problem as an iterative search problem, where we uniformly sample the transmittance with a random variable $\xi \in (0, 1)$, and search for the distance t so that $\xi(t) = \mathrm{T}(\mathbf{x}_0, \mathbf{x}_t)$. This approach is very similar to regular tracking [Sutton et al. 1999], which we extend to handle overlapping kernels efficiently. In particular, we first search for a segment $\mathcal{S}_k$ so that $(1 - \xi) \in [\mathrm{T}_{k-1}, \mathrm{T}_k]$. Then, we invert the transmittance inside the segment, solving for

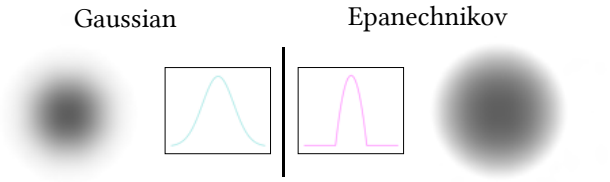


Fig. 3. Visual comparison between the 1D (insets) and 2D Gaussian and Epanechnikov kernels. We plot $\pm 3\sigma$ support to visualize the main difference between them: while the Gaussian kernel has infinite support, the Epanechnikov is finite, and decays sharply to 0. We further analyze differences between these kernels in Section 8.

$t \in [\widehat{t}_{k,0}, \widehat{t}_{k,1}]$ the following equation:

$$\log(1 - \xi) = - \sum_{i \in S_k} \tau_i \left(\mathbf{x}_{\widehat{t}_{k,0}}, \mathbf{x}_t \right). \tag{12}$$

Depending on the type of kernel $K_i(\mathbf{x})$ being used, and the amount of overlaps in segment $\mathcal{S}_k$ this equation may have a closed-form analytical solution. However, for cases where Equation (12) does not have a simple analytical solution (e.g. many overlapping kernels at the same segment), we rely on numerical root-finding. In particular, we use two different methods depending on the kernel: the Newton-Raphson and the bisection solvers. The former is our primary choice, since it is efficient and converges with good precision with a few iterations (1-3), given that our problem is well-conditioned. However, for certain long-tailed kernels (e.g., Gaussian) it occasionally suffers from numerical instability. In these cases, we rely in the slightly slower but more stable bisection method. Details for both methods can be found in the Supplemental.

Biased uniform distance sampling inside segments. To maximize speed, in complex assets fitted with a high number of small kernels we can avoid inversion (12) and simply sample inside each segment uniformly. While this means we sample with pdf $p(t) \neq \mu_t(\mathbf{x}_t)\mathrm{T}(\mathbf{x}_0, \mathbf{x}_t)$, in our implementation we assume that the probability distribution is close enough to it, allowing us to cancel out transmittance when computing the path throughput. This introduces a small amount of bias in exchange for a significant performance boost, which we analyze in Section 7.1.

5 KERNELS

In the previous section, we have derived a general framework for radiative transport in primitive-based media, defined using arbitrary kernel functions. Any kernel could be used within our framework as long as 1) they have limited support or their decay is such that they can be bounded or clipped efficiently and 2) closed-form solutions to their line integrals exist or can be numerically computed efficiently.

Here we implement our framework using two different kernels: the Gaussian and the Epanechnikov kernel. These have been successfully used in the signal processing, density estimation, rendering, and inverse graphics literature [Jakob et al. 2011; Kerbl et al. 2023; Liu et al. 2021]. Figure 3 shows their shape; we compare their performance in different applications in Section 8.

5.1 Gaussian Kernel

The normalized Gaussian kernel $g_i(\mathbf{x})$ for primitive $\mathcal{P}_i$ is defined as

$$g_i(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{3}{2}}(|\Sigma_i|)^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{c}_i)^T\Sigma_i^{-1}(\mathbf{x}-\boldsymbol{c}_i)}, \tag{13}$$

where Σ_i is the 3×3 covariance matrix and $\boldsymbol{c}_i$ its mean. As opposed to other convex kernels, Gaussians have unlimited support, which would potentially require evaluating all primitives in the scene for every ray. In practice, similar to Kerlb et al. [2023], we bound the kernel's support to three times the standard deviation, covering 99.73% of the Gaussian's density.

Transmittance Evaluation. We compute the closed-form transmittance of a Gaussian primitive $\mathcal{P}_i$ from point $\mathbf{x}_{t_0}$ to point $\mathbf{x}_{t_1}$ along the ray $\mathrm{r}(t) = \mathbf{x}_0 + \omega\, t$, by integrating the optical depth $\tau_i(\mathbf{x}_{t_0}, \mathbf{x}_{t_1})$. These parameters are them defined in the Gaussian's local coordinate system, also known as whitened space (which we obtain by linearly transforming the ray into the canonical space of the Gaussian ellipsoidal shell, where the ellipsoid is a sphere of radius 1). Due to the symmetry of the Gaussian kernel around its mean, we can simplify the definite integral by shifting the origin of the ray to ensure a symmetric integration domain, integrating optical depth $\tau_i(\mathbf{x}_{-t_s}, \mathbf{x}_{t_s})$ with $t_s = \frac{t_0+t_1}{2}$. This reduces the number of error functions required from 2 to 1. Plugging in Equation (13) into Equation (11) we get

$$\tau_i(\mathbf{x}_{-t_s}, \mathbf{x}_{t_s}) = \sigma_i \int_{-t_s}^{t_s} g_i(\mathbf{x}_t)\,\mathrm{d}t = Ge^{-\frac{1}{2}(a-b^2)}\left[\mathrm{erf}\left(t_s\sqrt{\frac{1}{2}}\right)\right] \tag{14}$$

with

$$a = \mathbf{x}_0 \times \mathbf{x}_0, \quad b = \mathbf{x}_0 \times \omega, \quad G = \frac{\sigma_i}{2\pi\sqrt{v^T\Sigma_i^{-1}v|\Sigma_i|}} \tag{15}$$

with $\mathrm{erf}(\cdot)$ being the error function. Assuming exponential media, we can compute the transmittance as $T(\mathbf{x}_{t_0}, \mathbf{x}_{t_1}) = \exp(-\tau_i(\mathbf{x}_{t_0}, \mathbf{x}_{t_1}))$.

Distance Sampling. For sampling the free-flight distance t in a single Gaussian primitive $\mathcal{P}_i$ with PDF $p(t) = \alpha_i G_i(\mathbf{x}_t)\mathrm{T}(\mathbf{x}_0, \mathbf{x}_t)$, we need to invert its optical depth (14), which by setting $t_0 = 0$ and using a random value $\xi \in (0, 1)$, and assuming an accumulated transmittance up to t_0 of β has closed form following

$$t(\xi) = \sqrt{2}\,\mathrm{erf}^{-1}\left(\mathrm{erf}\left(\frac{b}{\sqrt{2}}\right) - \frac{2\log(\xi/\beta)}{Ge^{-\frac{1}{2}(a-b^2)}}\right) - b \tag{16}$$

$$\tag{17}$$

For segments where more than one 3D Gaussian is contributing, we revert to using the bisection solver presented in Section 4. For single kernels, we normally use Equation 16. In practice, the piece-wise approximation of the erf^{-1} function commonly implemented in GPU math libraries [Kirk 2007] produces large errors for certain input values, and for these we resort to the bisection solver as well, at the cost of some performance.

5.2 Epanechnikov Kernel

In our primitive-based rendering framework, Epanechnikov kernels have a significant advantage compared to Gaussian kernels due to their limited support. This allows more compact primitive shells, which accelerate rendering by reducing the number of overlaps. The 3D Epanechnikov kernel $\mathcal{E}_i(x)$ for primitive $\mathcal{P}_i$ is defined as

$$\mathcal{E}_i(\mathbf{x}) = \begin{cases} \frac{15}{8\pi(7^3|\Sigma_i|)^{\frac{1}{2}}}\left[1 - \frac{1}{7}d(\mathbf{x})\right] & \text{if } d(\mathbf{x}) \le 1 \\ 0 & \text{otherwise}, \end{cases} \tag{18}$$

where, analogously to the Gaussian kernel, Σ_i is the 3×3 covariance matrix, $\boldsymbol{c}_i$ the mean, and $d(\mathbf{x}) = (\mathbf{x}-\boldsymbol{c}_i)^T\Sigma_i^{-1}(\mathbf{x}-\boldsymbol{c}_i)$.

Transmittance Evaluation. We derive the closed-form transmittance of an Epanechnikov-based primitive between points $\mathbf{x}_{t_0}$ and $\mathbf{x}_{t_1}$ by integrating the optical depth between them as

$$\begin{aligned}\tau_i(\mathbf{x}_{t_0}, \mathbf{x}_{t_1}) &= \sigma_i \int_{t_0}^{t_1} \mathcal{E}_i(\mathbf{x}_t)\mathrm{d}t \\ &= \sigma_i\mathcal{K}_{\mathrm{norm}}(\mathcal{K}_3(t_0^3 - t_1^3) + \mathcal{K}_2(t_0^2 - t_1^2) + \mathcal{K}_1(t_0 - t_1)),\end{aligned} \tag{19}$$

where $\mathcal{K}_1$, $\mathcal{K}_2$, $\mathcal{K}_3$ and $\mathcal{K}_{\mathrm{norm}}$ are constants that depend on the ray position and direction, and the eigenvalues of the kernel's covariance matrix Σ_i. Explicit expressions for these constants can be found in Appendix B. Analogously to any other kernel, and assuming exponential media, we can finally compute the transmittance as $T(\mathbf{x}_{t_0}, \mathbf{x}_{t_1}) = \exp(-\tau_i(\mathbf{x}_{t_0}, \mathbf{x}_{t_1}))$.

Distance Sampling. While there is an analytic solution for inverting Equation (12) for one Epanechnikov kernel, it is not practical, given its complexity. Instead, we directly use the Newton-Raphson solver for segments with both one or multiple overlapping kernels.

6 IMPLEMENTATION DETAILS

We implement our method in Mitsuba 3 [Jakob et al. 2022b], which we extended to support volumes of kernel primitives with different shell geometries, where we use ray tracing for querying primitives along the ray. We implement two different integrators, depending on the target application: 1) a *volumetric-primitives path tracer* (*VPPT*) supporting scattering media, and 2) a simplified *volumetric-primitives radiance field* (*VPRF*) integrator that computes the radiance field along a primary ray.

In both cases, we compute transmittance and emission by following the segment-based formulation described in Section 4.2, by iterating over the segments in an ordered fashion: This fits very well with our ray-tracing-based querying of the primitives, since we collect new segments by simply casting new rays from the previous intersection. Listing 1 shows a skeleton pseudocode for our two integrators.

6.1 Integrators

Volumetric-primitives path tracer (VPPT). This integrator is mostly an off-the-shelf volumetric path tracer supporting next-event estimation, with the key modifications of the transmittance evaluation and sampling, and the medium interaction routines. It solves Equation (7) using Monte Carlo estimation over multiple paths where, assuming non-emissive media, the throughput of each sample path is recursively computed as

$$L(\mathbf{x}_0) = \beta(\mathbf{x}_0, \mathbf{x}_i)\mu_t(\mathbf{x}_i)\alpha(\mathbf{x}_i)S(\mathbf{x}_i, \omega_i), \tag{20}$$

with $\omega_i = \frac{\mathbf{x}_i - \mathbf{x}_{i-1}}{|\mathbf{x}_i - \mathbf{x}_{i-1}|}$, and $\beta(\mathbf{x}_0 \rightarrow \mathbf{x}_i)$ the throughput of the eye subpath until $\mathbf{x}_i$ defined as

$$\beta(\mathbf{x}_0, \mathbf{x}_i) = \frac{\beta(\mathbf{x}_0, \mathbf{x}_{i-1}) \mathrm{T}(\mathbf{x}_{i-1}, \mathbf{x}_i) \mu_t(\mathbf{x}_{i-1}) \alpha(\mathbf{x}_i) f_\mathrm{p}(\mathbf{x}_{i-1}, \omega_{i-1} \rightarrow \omega_i)}{p(\mathbf{x}_i)}, \tag{21}$$

where $p(\mathbf{x}_i)$ is the probability of generating $\mathbf{x}_i$ from the previous scattering vertex $\mathbf{x}_{i-1}$. In our implementation, we obtain $\mathbf{x}_i$ using the standard practice of first sampling the phase function at $\mathbf{x}_{i-1}$, and then sampling distance with probability proportional to $\mathrm{T}(\mathbf{x}_{i-1}, \mathbf{x}_i)$.

Volumetric-primitives radiance field (VPRF). This implements in our volumetric primitives framework the radiance field image formation model, similar to e.g. [Kerbl et al. 2023; Mildenhall et al. 2020]. It significantly simplifies the image formation model from VPPT, by only considering the primary rays from the camera, accumulating radiance along the ray by querying a spherical harmonics-based emission in the primitives. In addition, we disable overlapping logic in order to favour speed. In practice, this means that we approximate Equation (7) using an ad-hoc approximate of the emission at each segment following

$$L_k(\mathbf{x}_{\widehat{t}_{k,0}}, \omega) \approx \left(1 - \mathrm{T}_k(\mathbf{x}_{\widehat{t}_{k,0}}, \mathbf{x}_{\widehat{t}_{k,1}})\right) \sum_{i \in S_k} \frac{\tau_i\left(\mathbf{x}_{\widehat{t}_{k,0}}, \mathbf{x}_{\widehat{t}_{k,1}}\right)}{\sum_{j \in S_k} \tau_j\left(\mathbf{x}_{\widehat{t}_{k,0}}, \mathbf{x}_{\widehat{t}_{k,1}}\right)} \overline{Q}_i(\omega). \tag{22}$$

6.2 Ray tracing volumetric primitives

As described above, we limit the support of our kernels by bounding their bandwidth with an ellipsoid shell. We query the primitives along a ray leveraging ray-tracing, which we accelerate through the construction of a BVH using Optix [Parker et al. 2010] for efficient ray traversal. We implemented a ray-ellipsoid intersection as a custom intersection routine in Optix. However, we found that this produces suboptimal BVHs given the potentially large support of the primitives. Moreover, a custom intersection kernel is significantly less performant than a hardware-accelerated ray-triangle intersection test. Thus, instead of using the ray-ellipsoid intersection, we can triangulate the ellipsoids, and ray-trace on triangles. We analyzed a variety of triangulation approaches (see Figure 12), and found a significant boost on performance despite the higher number of primitives in the scene, finding highly-tesselated icospheres to perform best in terms of execution times. One could consider using mesh instancing to avoid duplicating the shell's geometry. However, this approach reintroduces the problem of using axis-aligned bounding boxes for the individual instances in the instances BVH.

Stack array allocation. A limitation of our *primitive tracing* algorithms is their need to stack information while iterating over the primitives along the ray. This is particularly necessary for calculating the list of primitive exit points during segment iteration, as well as maintaining a record of the active primitives for the current segment. Unfortunately, most JIT frameworks, such as Dr.Jit or PyTorch, do not offer an API with a low enough level to perform pointer arithmetic. For efficiency, it is crucial to execute this performance-critical algorithm in registers, which necessitates the ability to allocate, access, and write into arrays of variables on the stack rather than on the heap. Fortunately, the NVidia PTX assembly language offers local-state space private memory for each thread to store its own data, which can be utilized for this purpose. We modified the Dr.Jit [Jakob et al. 2022a] core to include routines in the JIT compiler that generate such intrinsics.

```python
def primitive_tracing(ray, max_depth, P = []):
  depth = 0
  t0 = 0.0 # Current segment start time
  while depth < max_depth:
    V = [(p, p.t1) for p in P] # List primitive exit points
    p = ray_intersect(ray)     # Find next primitive
    V.append((p, p.t0))        # Add primitive entry point

    # Process all exit points up to this intersection
    while not V.is_empty():
      # Find next vertex to process
      v = min(V, key=lambda x: x.t)

      # Process segment according to application
      process_segment(ray, G, t0, v.t)

      t0 = v.t # Move on to next segment
      if p == v.p: break # Check if we have reached p
      P.remove(v.p) # Exiting v.p
      V.remove(v)

    P.append(p) # Ray is now entering primitive p
    ray = ray.move_to(p.t0) # Update ray position
    depth += 1
```

Listing 1. Pseudo-code implementation of our primitive tracing algorithm. In *VPPT*, `process_segment()` samples a medium interaction in that segment and recursively calls `primitive_tracing()` on sampled position. In *VPRF*, `process_segment()` computes the radiance field contribution of that segment following Equation (22).

6.3 Differentiating volumetric primitive integrators

We develop a backward derivatives propagation routine for our integrators, so that they can be used to solve inverse problems. Drawing parallels to the work by Vicini et al. [2021b], these routines utilize the same sequence of random numbers to generate identical light paths in both the primal and backward rendering phases, and propagate the gradients to and from the scene parameters along these paths. Additionally, they accept the total radiance along these paths as input, which is then used to retrieve the various quantities required for reverse-mode differentiation.

A unique aspect of our adjoint implementation is the inclusion of loops to iterate over primitives along a ray and influence the different segments. Unfortunately, automatic differentiation systems like Dr.Jit [Jakob et al. 2022a] do not support automatic derivative propagation through loops. Although it is possible to modify Dr.Jit to automatically generate the appropriate transformation, it would not yield an efficient outcome, as each loop iteration would require storing copies of all loop variables to facilitate a reversal under general conditions. Consequently, it is imperative to provide an adjoint formulation of the derivative propagation within those loops.

Appendix C includes the adjoint form for our two integrators. The full derivations and implementation details can be found in

the Supplemental document, as well as comparisons against finite differences. It is important to highlight that the backward propagation process in our implementation has the capability to propagate gradients to all primitive parameters simultaneously. This is a significant advantage over finite differences, which typically handles parameters individually.

7 APPLICATIONS

We can use our volumetric primitives and rendering algorithms in a wide range of volume rendering applications, from traditional forward rendering of scattering media, to physically-based (PB) inverse rendering, and radiance field optimization and rendering.

7.1 Forward Rendering

In our forward rendering experiments we focus on scenes where only density varies, and we fix the single-scattering albedo and phase function. We set the primitive kernel to a Gaussian kernel, which is more suitable for representing smooth volumetric data such as clouds or smoke. We use three different volumetric assets, obtained from voxel grid representations. We transform the grid-based representation to a Gaussian mixture model (GMMs) by optimizing the GMM using the inverse tomography pipeline we describe later in Section 7.2, where the GMM is optionally initialized using expectation-maximization [Moon 1996]. The cost of fitting depends on the number of primitives, ranging from 15 minutes of the *Smoke* asset (Figure 6), to several hours in the most complex *Cloud* asset (Figure 1). Note however that in our implementation this process is not optimized and can be significantly accelerated. Further details can be found in the Supplemental document.

7.1.1 Experiment setup. As baseline, we compare our method against a grid-based representation of the volume rendered using a weighted delta tracking free-flight sampler [Raab et al. 2006; Woodcock 1965] with transmittance computed using residual ratio tracking [Novák et al. 2014]. For both free-flight sampling and transmittance estimation, we use a local majorant precomputed in a coarser grid where each supervoxel stores local statistics (local majorant, minorant, and average density) over a 4^3 voxels neighborhood. This supervoxel-grid is traversed during sampling using DDA [Szirmay-Kalos et al. 2011]. Having tighter (local) majorants substantially improves free-flight sampling in complex heterogeneous media, reducing the probability of null-scattering events. We choose *residual ratio tracking* as the transmittance estimator for next-event estimation in all the references, using the average local density as control coefficient; cost per sample for our DDA traversal-based *residual ratio tracking* is substantially smaller than e.g. *power-series* estimators [Georgiev et al. 2019], which is typically favored for high albedo volumes, where long paths and higher sample counts are expected. Unless otherwise stated, the reference voxel grid is computed by voxelizing our mixture model into a high-resolution grid, so that the distributions of density between the reference and our method are as close as possible, allowing per-pixel error comparisons in terms of computing gain.

As for our method, we use our *VPPT* integrator with next-event estimation (NEE). We trade off quality for performance in our analytic transmittance estimates (Figure 6) with Russian Roulette termination during NEE, and simplifying sampling by uniformly sampling a position along the critical segment instead of inverting the CDF over all intersecting Gaussians.

Fig. 4. *Dust explosion* asset, fitted to 40k Gaussians and discretized into a 512^3 resolution grid, both rendered until convergence. We achieve high quality fits and renders on tiny memory budgets.

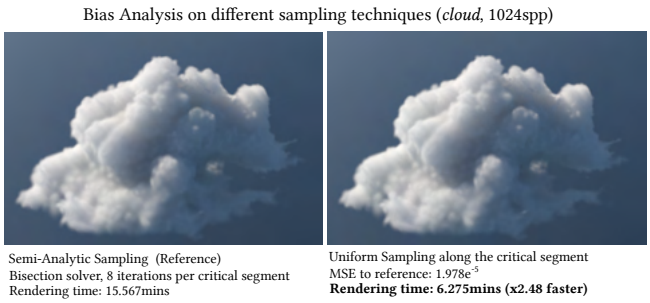


Fig. 5. *Cloud* asset, rendered with 1024 samples per pixel with both our unbiased semi-analytic (left) and our biased uniform sampling (right) distance sampling techniques inside each segment. Given the accuracy of our transmittance estimates and the small size of segments in complex assets such as *cloud*, the bias introduced is minimal, while obtaining significant speed-ups. Rendering times reported on a NVIDIA RTX 3080.

7.1.2 Results. Figures 1 and 4 demonstrate our method rendering complex assets with high constant single-scattering albedo ($\alpha = 0.99$). The main source of variance in these scenarios is the free-flight sampling. Our method compares positively against state-of-the-art techniques for grid-based volume rendering, both in terms of performance and memory footprint. Given that the main limitation in production volume renderers has traditionally been memory bandwidth, our approach could deliver substantial speedups; while also being appealing in more memory-constrained environments.

Figure 5 analyzes the bias introduced by using the biased uniform distance sampling inside the segment, instead of numerically inverting the CDF. At the cost of a small amount of bias, uniformly sampling the distance inside segments gives a very substantial speed-up compared to the exact semi-analytical CDF inversion.

Comparison against transmittance estimators. In Figure 6 we compare our analytical transmittance estimation against state-of-the-art

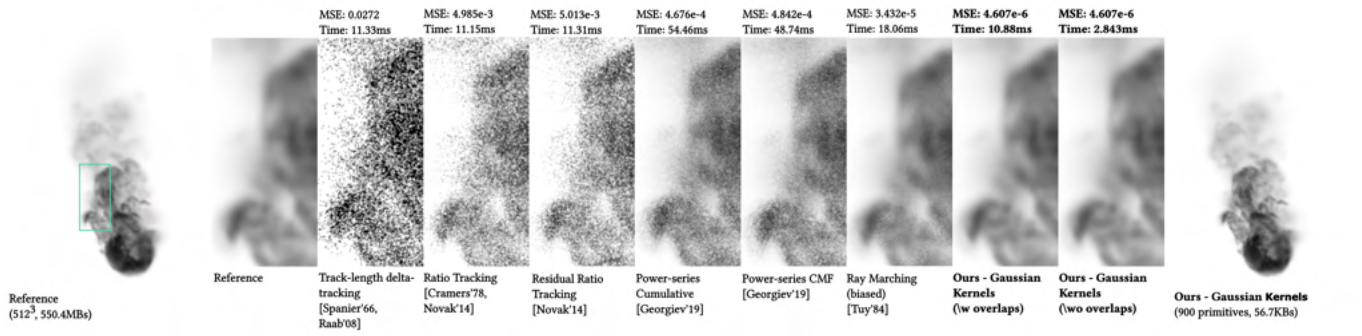


Fig. 6. *Smoke* asset, fitted to 900 Gaussians (37KB) and discretized into a 512^3 resolution grid (550.4MBs) to be used by previous work. The supervoxels grid has a resolution of 64^3 voxels. In this integrator, only primary rays are traced. We show a single sample per pixel (spp) in all the renders above. Our analytic transmittance estimation produces no variance from a single spp (reported error is primarily the discretization difference between the GMM and voxel grid distributions), while our rendering solution still delivers lower cost per sample than competing Monte Carlo-based estimators, at a fraction of the memory footprint. We can further reduce cost per sample to 2.84ms by disabling segment-by-segment integration, and simply integrating every intersected Gaussian along the primary ray paths as a whole, producing the same image. We use the former during distance sampling, and the latter during next-event estimation, maximizing efficiency while staying unbiased. Rendering times reported on a NVIDIA RTX 3080.

tracking techniques on an absorbing medium lighted by a constant white environment, where the sole contributor to variance is transmittance. We use the *Smoke* asset, approximated with 900 Gaussians. We compare against a wide range of transmittance estimators: *track-length delta tracking* [Raab et al. 2006; Spanier 1966; Spanier and Gelbard 2008], *ratio tracking*[Cramer 1978; Novák et al. 2014], *residual ratio tracking* [Novák et al. 2014], *ray marching* [Perlin and Hoffert 1989; Tuy and Tuy 1984] and both *power series-cumulative* and *power series-CMF* [Georgiev et al. 2019]. Note that all of these estimators are unbiased except *ray marching*. For *residual ratio tracking*, we use the average local density as the control coefficient, and both this estimator, *ratio tracking* and *track-length delta tracking* leverage local statistics to reduce cost per sample and improve sampling efficiency. The rest of the estimators use global statistics instead. Our analytic transmittance integrator produces estimates with zero variance even on a single sample per pixel. Our cost per sample is also the smallest of all tested approaches, which further showcases the efficiency of our approach.

Analysis on quality-performance trade-off. The performance of our method is dependent on the number of primitives used to model the density distribution, and most importantly, on the amount of overlap between them. However, aggregation is straightforward when using Gaussian distributions; in Figure 7, we explore the scalability of our approach: at the cost of some visual fidelity, we can drastically increase compactness and rendering speed. This also makes our representation very suitable for level-of-detail (LoD) applications.

Note that our approach provides a significant compression rate even when compared with state-of-the-art adaptive grid approaches (OpenVDB), with a small penalty on quality (sharpness). In terms of performance, adaptive grid approaches require tracking algorithms as discussed earlier and thus would suffer from similar increased variance compared with our method. It is expected though that NanoVDB would provide some speed-up compared with dense grids due to reduced GPU memory bandwidth, with additional penalties on lookups due to tree traversal.

7.2 Physically-Based Inverse Rendering

Our primitive-based formulation of media is very suitable for inverse problems where the optical properties of heterogeneous media are reconstructed from observations, given its compactness in both representation and gradients. For that, we use a differentiable version of our *VPPT* integrator, with the adjoints presented in Section 6. In this section we demonstrate a proof-of-concept inverse pipeline, under different problems setups, including both the inversion of absorbing and scattering media. Note that it is outside the scope of this paper to provide a robust, fully-fledged optimization pipeline, which would require further investigation.

Parameterization. We parameterize our 3D Gaussians similarly to 3D Gaussian splatting [Kerbl et al. 2023], distilling them into mean, scale and rotation (using Euler angles instead of quaternions), which we optimize and then later use to compute the covariance matrix Σ_i for the primitive $\mathcal{P}_i$ as

$$\Sigma_i = \mathcal{R}_i \mathcal{S}_i \mathcal{S}_i^T \mathcal{R}_i^T \tag{23}$$

with $\mathcal{R}_i$ and $\mathcal{S}_i$ the primitive rotation and scale matrices, respectively.

Optimization Procedure. We begin our set of primitives with a single entity and gradually incorporate more primitives by randomly sampling points on the bounding sphere of all current primitives. The parameters of the primitive are optimized using an adapted version of the Adam optimizer [Kingma and Ba 2017] that we call *Bounded Adam*, which establishes hard limits on the boundary values of some parameters to avoid 1) intersection artifacts on extremely small primitives; 2) generating primitives outside of the determined scene bounding box, and 3) avoiding large primitives, which due to our reliance on axis-aligned bounding boxes for the BVH would increase our rendering times.

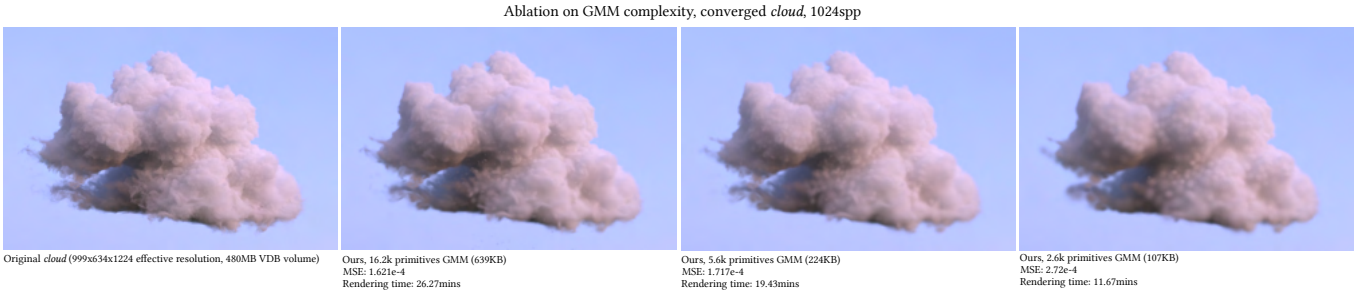


Fig. 7. *Cloud* asset, rendered at 1024spp with varying amounts of Gaussian primitives, vs the original VDB volume. We achieve high compression rates (a ×5251 compression rate with respect to the dense voxel grid, compared to the ×6.823 rate obtained by OpenVDB) and can easily trade off quality for memory and performance. Reported timings on a NVIDIA RTX 3080, equal settings for our method. Further performance gains can be achieved by tuning down individual parameters depending on the asset complexity (e.g., reducing the maximum number of overlapping kernels to be considered per segment).

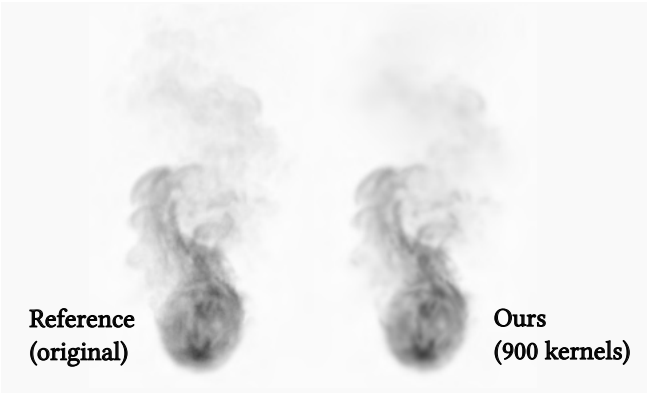


Fig. 8. Original *Smoke* voxel grid (left), and our fit with a 900 Gaussians computed using our inverse tomography pipeline (right), starting from a random set of primitives. We achieve a PSNR of 54.028dB and a compression rate over the original volume of ×291. We use this optimized asset in Figure 6.

Bounded Adam. The key idea of Bounded Adam is to move the optimized value by half of the distance towards the bound if a gradient step reaches one of the bounds. This ensures that the optimization process can approach the bound as closely as possible without ever stepping over it. In our experience, this works better than simply clamping the values, as this process will also reset the optimizer state for this parameter.

Loss. We employ a λ-weighted combination of mean-absolute error (L1) and D-SSIM [Wang et al. 2004], to which we add two regularization terms to control anisotropy and density of small primitives. This prevents individual primitives from overfitting specific details in the training images. It should be noted that these regularization terms are only active above a certain threshold. In summary, the loss we use for reconstruction reads

$$\mathcal{L} = \lambda \mathcal{L}_1 + (1-\lambda)\mathcal{L}_{\text{SSIM}} + \mathcal{L}_{\text{ani}} + \mathcal{L}_{\text{d}}\,, \tag{24}$$

with

$$\mathcal{L}_{\text{ani}} = w_{\text{ani}} \sum_{i=1}^{N} \frac{\min\left(\text{Eig}(\mathcal{S}_i)\right)}{\max\left(\text{Eig}(\mathcal{S}_i)\right)} \qquad \text{and} \qquad \mathcal{L}_{\text{d}} = w_{\text{d}} \sum_{i=1}^{N} \frac{\sigma_i}{|\mathcal{S}_i|},$$

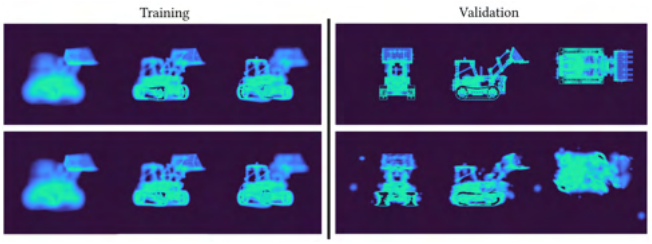


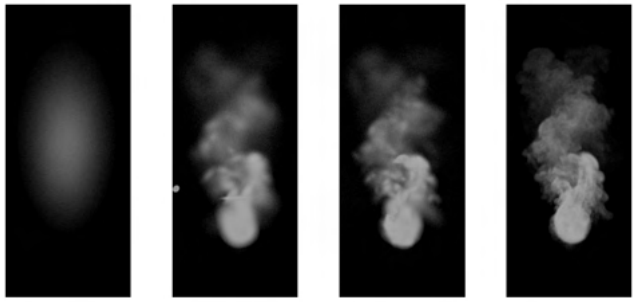
Fig. 9. Inverse tomography from a focal stack. Left: Selected frames from a focal stack composed of 32 images, rendered with the same camera pose and linearly interpolating focus distance between far and near planes (top); fitting results after 1500 iterations (bottom, 907 primitives, PSNR: 39.3186). Right: Reference evaluation views (top) and novel rendered views (bottom) from our inverse telecentric tomographic reconstruction. We can see that even when trained on a single camera pose, it can reconstruct 3D structure.

where w_{ani} and w_{d} are the weight parameters controlling the strength of the two regularizers, and $\text{Eig}(\cdot)$ being the eigenvalue operator.

7.2.1 *Results.*

Inverse tomography. We first demonstrate our pipeline on reconstructing heterogeneous density on purely absorbing media, illuminated by a constant white environment with a Gaussian kernel. Figure 8 shows results on the *Smoke* dataset, where we obtain a PSNR of 54.028dB with only 900 Gaussians, starting from a random set of Gaussians, from a set of 16 images around the smoke plume. Note that this pipeline has been used to generate all the assets in Section 7.1.

Figure 9 further demonstrates our pipeline with a more challenging camera model, mimicking tomographic microscopy. We create a focal stack using a telecentric (orthographic thin-lens) camera from a single point of view on an absorbing *Bulldozer* dataset. We reconstruct it with 907 Gaussians from that focal stack of 32 images using our pipeline. Even when employing a single point of view and using limited numbers of primitives, we obtain good 3D reconstructions from the focus stack (PSNR: 39.3186), even from unseen points of view (right).



(a) initial **(b)** intermediate **(c)** converged **(d)** reference

Fig. 10. Inverse scattering results on the *Smoke* asset. Starting from a single primitive (a), we iteratively optimize its parameters and spawn child primitives surrounding it, progressively adding more detail (b). Our final mixture model (c) achieves a PSNR of 36.64614 dB on average using only 551 Gaussians. The scattering medium is illuminated by a white enviroment, removed for visualization.

Inverse scattering. Figure 10 shows the result of a more complex experiment, where we reconstruct a heterogeneous *scattering medium*. This is significantly more challenging than tomography, given the much higher dimensionality of the problem (i.e. long paths, high sampling variance due to multiple scattering). We reconstruct a highly-scattering version of the *Smoke* plume with an isotropic phase function, from a set of 32 images (resolution of 256^2 pixels) around the asset. As opposed to the inverse tomography in Figure 8, we start from a single Gaussian and progressively spawn child Gaussians surrounding it in order to add detail in a progressive and controlled manner, and run our optimization for 1200 iterations. We converged to a 551-Gaussians fit, with a PSNR 36.64 dB on average.

7.3 Radiance Fields

As a third application, we present a complete inverse radiance field optimization and rendering pipeline, that reconstructs view-dependent appearance of both synthetic and real-world scenes from captured images. The purpose of this experiment is to further showcase the flexibility of our approach and demonstrate how it can enhance the capabilities of competing kernel-based radiance field solutions such as 3DGS.

For this application, we use our simpler *volumetric primitive radiance field (VPRF)* integrator (Section 6). Note that while this integrator is a coarse approximation of the underlying physical model described in Section 4, it is less important in this application, since the parameters for the kernel primitives are optimized to match the input data, thus compensating for the approximation. This simplification improves performance by dropping support of segment by segment integration. Since we reconstruct scenes with surface-like features, we use both Gaussian and Epanechnikov kernels in this application. Similar to previous works [Kerbl et al. 2023] we use a spherical harmonics expansion to model the anisotropic emission per primitive $\overline{Q}_i(\omega)$. For a detailed view on our optimization pipeline, including initialization, pruning and cloning strategies, and hyperparameters used, we refer to the supplementary material.

Table 1. Quantitative quality comparison against competing methods on the real MipNeRF-360 [Barron et al. 2022] and Tanks [Knapitsch et al. 2017] datasets, and on the synthetic Blender dataset [Mildenhall et al. 2020]. Average values for each metric and dataset are reported. Individual results and average execution times can be found in Supplemental.

	MipNeRF-360		Tanks		NeRF-Blender	
Method	**PSNR↑**	**SSIM↑**	**PSNR↑**	**SSIM↑**	**PSNR↑**	**SSIM↑**
NeRF	21.76	0.455	21.76	0.455	32.54	0.961
Plenoxels	21.91	0.496	23.22	0.774	34.10	0.975
iNGP-Base	22.19	0.491	23.26	0.779	35.64	0.981
Mip-NeRF 360	24.31	0.685	24.91	0.857	36.10	0.980
3D Gaussian Splatting	25.25	0.771	25.19	0.879	36.073	0.982
Ours - Gaussian	22.07	0.536	23.93	0.850	32.11	0.957
Ours - Epanechnikov	21.56	0.515	23.98	0.845	33.60	0.972

As opposed to 3DGS, we use the same parameters in all our experiments, for both synthetic and real datasets. Fine-tuning per dataset could substantially improve the presented results.

7.3.1 Results. Figure 11 shows qualitative comparisons against previous works [Barron et al. 2022; Fridovich-Keil et al. 2022; Kerbl et al. 2023; Mildenhall et al. 2020; Müller et al. 2022]. Quantitative average quality metrics can be found on Table 1, while Table 2 gives an overview of where our method stands with respect to previous work in terms of compression, rendering times, optimization times and other metrics. A complete quantitative evaluation can be found in Supplemental.

Our results suggest comparable performance in terms of image quality to previous ray-traced radiance field solutions (e.g., instant NGP [Müller et al. 2022]), while falling behind the state-of-the-art, rasterized 3DGS [Kerbl et al. 2023]. However, our method is faster than previous radiance field ray-traced works, achieving framerates of ~40-50 FPS on complex scenes on average using quality presets. In many cases, the Epanechnikov kernel not only outperforms Gaussian kernels in rendering speed, but also in visual quality.

Ablation on the choice of shell. We use analytic ellipsoids as shells during the optimization due to their memory compactness. However, as discussed in Section 4, we propose several alternatives when rendering (forward or inverse) with bigger memory budgets. In Figure 12 we include an ablation over the synthetic datasets we optimized, where we show up to 4.96 times faster rendering speed when using alternative triangle-mesh based shells.

Ablation on Maximum Primitive Depth. We can trade off quality for performance by limiting maximum primitive depth. Since 3DGS sorts the primitives before splatting, the potential performance gain by early integration termination is limited. In contrast, using ray-tracing can provide significant speedups in rendering times by terminating recursion early, as shown in Figure 13.

7.3.2 Comparison with 3DGS. While slower than the rasterization-based baseline 3DGS, our ray-traced volumetric model, derived from the RTE introduces several advantages, including:

Accurate Ordering. Due to the inherent order obtained through recursive ray-tracing, we do not need to sort our primitives, and our order is exact, taking the anisotropic shape of our primitives into account. This avoids potential ordering artifacts observed in 3DGS, where primitives are sorted solely on their means and transformed

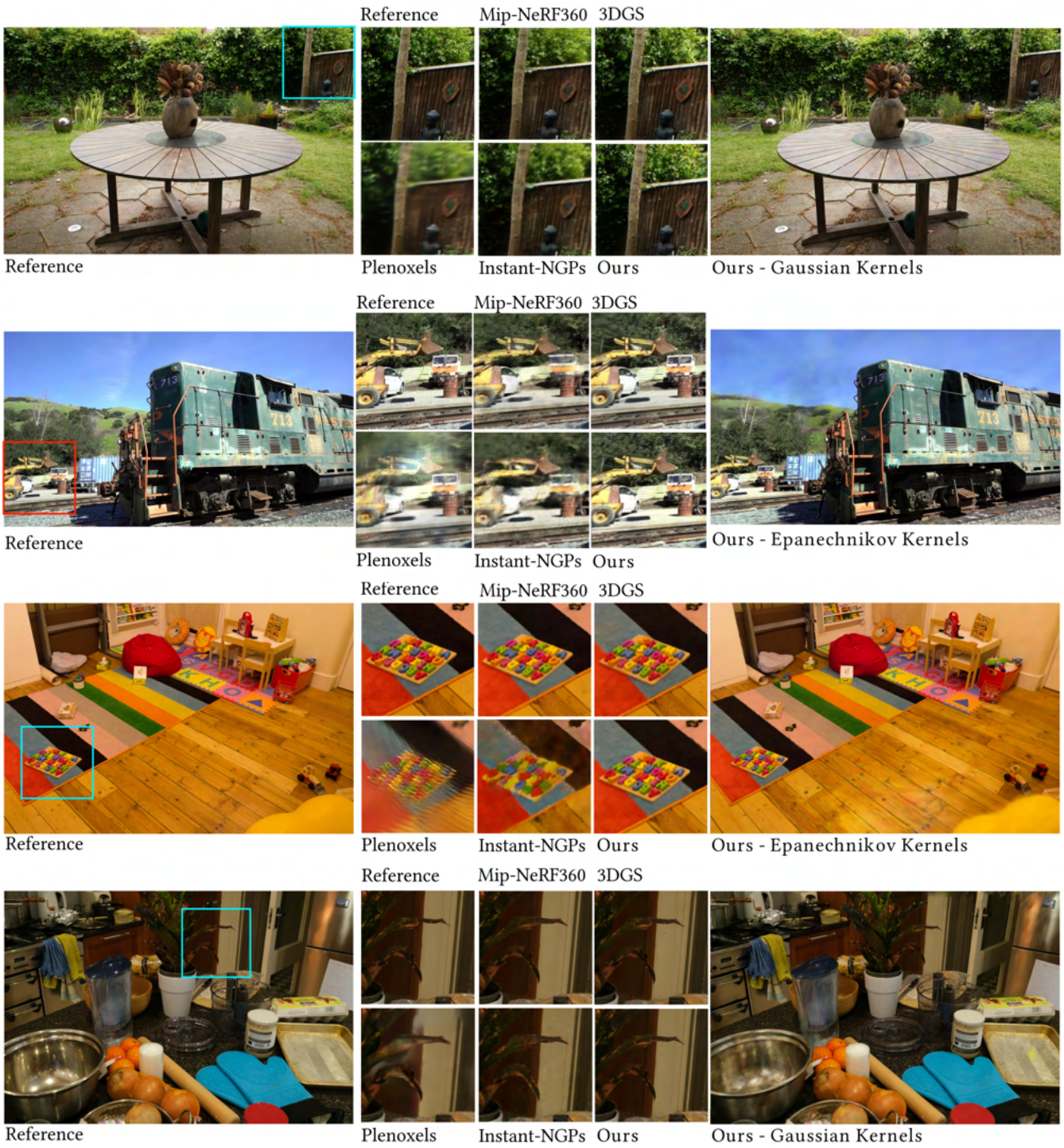


Fig. 11. Qualitative comparison against previous radiance field rendering methods, including Plenoxels [Fridovich-Keil et al. 2022], Instant NGPs [Müller et al. 2022] and Mip-NeRF360 [Barron et al. 2022], and the rasterized 3DGS [Kerbl et al. 2023], in the real datasets *garden*, *train*, *playroom* and *counter* (Tanks and Temples [Knapitsch et al. 2017] and MipNeRF360 datasets [Barron et al. 2022]). Our method offers competitive quality with respect to the state of the art, and it stands as the fastest ray-traced solution (Table 2). Both our tested kernels perform similarly in terms of quality in real scenes, while Epanechnikov kernels are substantially faster. Complete quantitative evaluations can be found in Appendix, as well as expanded images of the insets presented in this figure

.

Table 2. **Qualitative comparison of our method with other popular radiance-field (RF) approaches**. Reported performance metrics are averages over the tested real datasets, using quality (lower bound) and performance (upper bound) presets in our case. The complete set of results can be found in Supplemental. All timings are reported on similar setups using an RTX A6000, with timings of previous works taken from [Kerbl et al. 2023].

Method	NeRF [Mildenhall et al. 2020]	Plenoxels [Fridovich-Keil et al. 2022]	iNGP [Müller et al. 2022]	Mip-NeRF 360 [Barron et al. 2022]	3DGS [Kerbl et al. 2023]	**Ours - Gaussian**	**Ours - Epanechnikov**
Optimization time (RF)	~12-24h	~0.1-0.5h	~2-8m	~48h	~0.3-0.5h	~2h-16h	1h-10h
Rendering speed (RF, real datasets)	~0.05-0.1 FPS	~6-13 FPS	~3-17 FPS	~0.06-0.14 FPS	~134-197 FPS	~28-39 FPS	~42-53 FPS
Memory footprint (RF)	~5-10MB	~1-2.7GB	~13-48MB	~5-10MB	~100-730MB	~100-900MB	~100-900MB
Rendering framework	Ray-tracing	Ray-tracing	Ray-tracing	Ray-tracing	Rasterization	Ray-tracing	Ray-tracing
Complex camera models	Yes	Yes	Yes	Yes	No	Yes	Yes

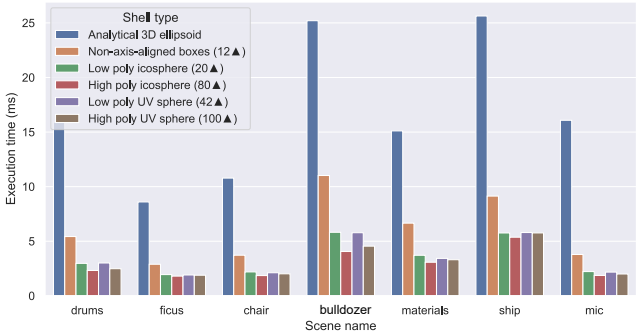


Fig. 12. Rendering times using different shells across synthetic dataset scenes optimized with our method. Triangle-mesh-based shells excel at performance due to hardware ray-triangle intersection support and a more efficient traversal, at the cost of some extra memory per primitive. On average, we got a speedup of X4.96 by using triangle mesh icospheres vs analytical 3D ellipsoids.

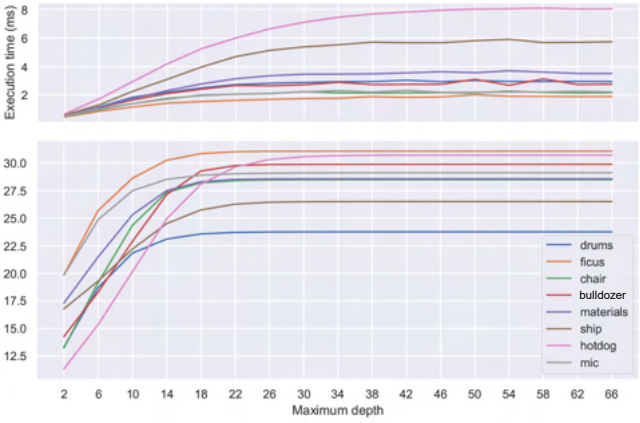


Fig. 13. Effect of maximum depth (number of Gaussians integrated in the ray), as a function of PSNR quality and rendering time, on synthetic datasets. At a maximum depth of 14 Gaussians/ray, we averaged 2.36ms per frame (434.78 FPS) across all datasets, while achieving a good balance between quality and performance. With a more aggressive setting, at 10 Gaussians/ray, we measured an average execution time of 1.82ms (549.45 FPS).

Fig. 14. Complex camera models in radiance fields. We render the *playroom* dataset (MipNeRF360 datasets [Barron et al. 2022]) using a 360-degree camera. Our ray-tracing-based framework allows us to use arbitrarily complex camera models, as opposed to 3DGS. Artifacts on the ceiling and under the table are due to undersampling in the training data.

to billboards when splatted, which may cause popping artifacts (Figure 15) and inconsistent ordering among different views.

Ray tracing. Among the many advantages of working in a ray-tracing framework compared to rasterization are the possibility of employing complex camera models without any perspective corrections for both reconstruction and rendering. We showcase the latter by rendering one of our RF optimized scenes with a 360-degree camera in Figure 14. It is also straightforward to support better foveated and wide-angle rendering in VR applications and further accelerate path-tracing pipelines that rely on precomputed radiance emission to model complex light sources [Condor and Jarabo 2022; Zhu et al. 2021]. By leveraging prior work on early ray-termination, we also obtain a simple way of controlling rendering speed at the cost of some quality. A physically-based ray-traced formulation further enables future extensions towards relighting, global illumination, soft shadows, etc.

8 COMPARISON BETWEEN KERNELS

Finally, we assess the differences between the two proposed kernels. Gaussian kernels have properties that could be desirable in some situations (e.g. the convolution of Gaussians is Gaussian; convolving Gaussian kernels over the GMMs could be used to simulate animation, LoD, and motion blur [Leimkühler et al. 2018] efficiently). In contrast, Epanechnikov kernels have limited support, which creates more compact primitives, resulting in more efficient acceleration

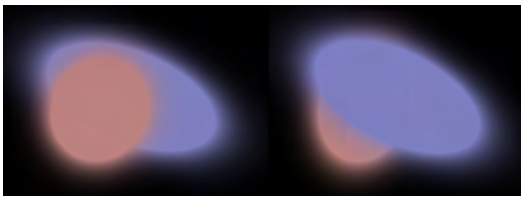


Fig. 15. Splatting a mean-sorted stack of Gaussians creates noticeable artifacts. Rotating 3DGS Gaussians can "jump" order sharply depending on viewpoint, which is normally compensated through the optimization process by adding more Gaussians.

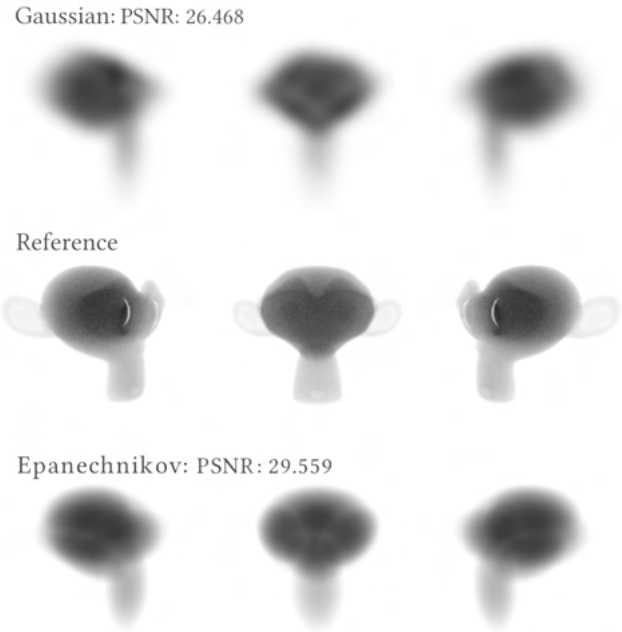


Fig. 16. Example toy tomography of a mesh-bounded absorbing medium. We render 8 different views around the volume. For both kernels, we initialize on the same 8 randomly sampled primitives and optimize this fixed set until convergence (1-2 minutes). The Epanechnikov kernel performs substantially better in this controlled example; its faster decay seems to better model sharp edges. We observed a similar behaviour in our (uncontrolled) experiments with radiance field optimizations in Section 7.3.

structures and a sparser representation. We showcase these differences on a couple of experiments.

Regression quality comparison. We set up a simple inverse tomography experiment, where we optimize an heterogeneous absorbing medium and with an 8-primitives mixture model. The absorbing media is surrounded by a constant environmental emitter. The results of this experiment can be seen in Figure 16. The sharp decay and smaller footprint of Epanechnikov kernel results in much sharper fits. This behavior makes them a potentially better candidate for hard surfaces and high-frequency detail.

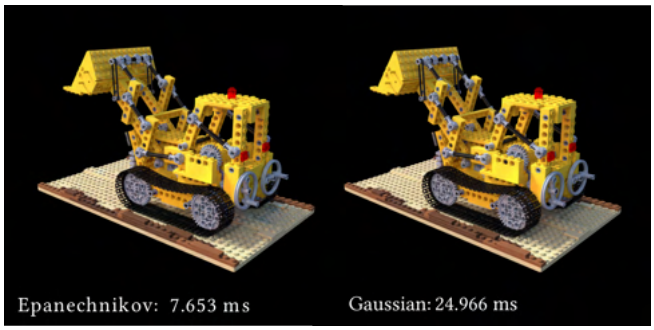


Fig. 17. Speed comparison between Gaussian and Epanechnikov kernels. For virtually the same radiance field (PSNR between the different mixture renders of 38.05 dB), the Epanechnikov kernel is 3.26X faster than the Gaussian one (1spp, average over 6 views, analytic shells).

Rendering speed comparison. In Figure 17 we compare the rendering performance from two mixtures (Gaussian vs Epanechnikov) on a more complex radiance field example. We use our radiance field pipeline (Section 7.3), and train a Gaussian mixture model of the *Bulldozer* asset. Then, we swap the Gaussian kernel with the Epanechnikov, and optimize solely the scale for a few more iterations (using the Gaussian renders as a reference, not the original images). Producing virtually the same images, the Epanechnikov-based mixtures are much faster mainly due to the compactness of the primitive shells. The main performance bottleneck of our approach is ray traversal in a potentially degenerate BVH due to large overlapping kernels, which substantially improves with compaction. The difference can be even greater on large-scale scenes, where a single large primitive (e.g., in the sky) can substantially decrease overall performance.

9 DISCUSSION

We have presented a general framework for model scattering and emissive media using volumetric primitives based on 3D kernels. Together with an efficient implementation in different integrators (and their adjoints), we have shown a variety of applications including physically-based forward and inverse rendering of media, as well as radiance field rendering. Our volumetric primitives are flexible, compact and very efficient to render. We have shown the versatility of our approach in using different kernels, and the potential benefits of extending beyond the classic Gaussian kernel in rendering tasks. We have cleared the way for the introduction of other kinds of kernels in the future, and expect the method to be of great interest in a media production environment progressively more interested in real-time solutions for path tracing and physically-based rendering.

Limitations & Future Work. Our work presents several limitations and opportunities for future work. First of all, exploring how to generalize our radiative framework to anisotropic or correlated media will increase the range of scenes our model can support, including volume-based representations of scenes with solid surfaces. Exploring how a primitive-based general media can be used for level-of-detail of complex appearance is also a promising avenue of

future work, compared to voxel grid-based approaches [Loubet and Neyret 2017; Vicini et al. 2021a].

Our inverse optimization pipeline demonstrates the potential of our method for inverse rendering tasks, but we acknowledge that more work is needed to optimize for more complex volumetric media. This would require a more thorough analysis on the initialization, losses and optimization strategies, specially as we increase the complexity of the underlying light transport theory. This would significantly improve how our assets are modeled, which for now are limited to density-only heterogeneities.

As for the kernels proposed, there is a myriad of potentially good candidates, and different applications might benefit from different kernels. One particularly interesting avenue of future work would be to use parametric kernels, which could potentially bring higher flexibility and expressive power per primitive, reducing the number of primitives required for modelling an asset.

Another limitation of our inverse pipelines is the comparatively longer training times when compared to other alternatives. A large contributor to this is the JIT-compilation of the scene, which is re-run for every new iteration. This can be substantially improved by re-using compiled code across iterations, which is merely a technical limitation of DrJIT and could be lifted in the near future.

In terms of absolute rendering quality, our radiance fields are slightly lacking when compared with current state of the art [Kerbl et al. 2023]. Still, we perform similarly to previous ray-tracing based solutions while offering much faster rendering and gracefully-degrading quality-performance tools. Furthermore, a physically-based formulation for modelling and rendering can bring many advantages over a rasterized framework (i.e. shadows, scattering, complex camera models, integration into physically-based renderers, etc). We believe more development on the optimization pipeline can easily bring it up to par with 3DGS in quality as well.

ACKNOWLEDGEMENTS

We would like to thank Christophe Hery and Olivier Maury for their continued support throughout this project and proof-reading. Jorge Condor and Piotr Didyk have been supported by the Swiss National Science Foundation (SNSF, Grant 200502) and an academic gift from Meta.

REFERENCES

Carlos Aliaga, Carlos Castillo, Diego Gutiérrez, Miguel A. Otaduy, Jorge Lopez-Moreno, and Adrián Jarabo. 2017. An Appearance Model for Textile Fibers. *Computer Graphics Forum* 36, 4 (2017).

Marco Ament, Christoph Bergmann, and Daniel Weiskopf. 2014. Refractive radiative transfer equation. *ACM Trans. Graph.* 33, 2 (2014).

Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. 2021. Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields. In *Proceedings of ICCV*.

Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. 2022. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. In *Proceedings of CVPR*.

Benedikt Bitterli, Srinath Ravichandran, Thomas Müller, Magnus Wrenninge, Jan Novák, Steve Marschner, and Wojciech Jarosz. 2018. A radiative transfer framework for non-exponential media. *ACM Trans. Graph.* 37, 6 (2018).

Aljaž Božič, Denis Gladkov, Luke Doukakis, and Christoph Lassner. 2022. Neural Assets: Volumetric Object Capture and Rendering for Interactive Environments. *arXiv preprint arXiv:2212.06125* (2022).

Forrest B Brown and William R Martin. 2003. Direct sampling of Monte Carlo flight paths in media with continuously varying cross-sections. In *Proc. ANS Mathematics & Computation Topical Meeting*, Vol. 2.

Brent Burley, David Adler, Matt Jen-Yuan Chiang, Hank Driskill, Ralf Habel, Patrick Kelly, Peter Kutz, Yining Karl Li, and Daniel Teece. 2018. The Design and Evolution of Disney's Hyperion Renderer. *ACM Trans. Graph.* 37, 3 (2018).

LL Carter, ED Cashwell, and WM Taylor. 1972. Monte Carlo sampling with continuously varying cross sections along flight paths. *Nuclear science and engineering* 48, 4 (1972).

Subrahmanyan Chandrasekhar. 1960. *Radiative Transfer*. Courier Corporation.

Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. 2023. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In *Proceedings of CVPR*.

Per Christensen, Julian Fong, Jonathan Shade, Wayne Wooten, Brenden Schubert, Andrew Kensler, Stephen Friedman, Charlie Kilpatrick, Cliff Ramshaw, Marc Bannister, Brenton Rayner, Jonathan Brouillat, and Max Liani. 2018. RenderMan: An Advanced Path-Tracing Architecture for Movie Rendering. *ACM Trans. Graph.* 37, 3 (2018).

Jorge Condor and Adrián Jarabo. 2022. A Learned Radiance-Field Representation for Complex Luminaires. In *Proceedings of EGSR*.

Jorge Condor, Michal Piovarči, Bernd Bickel, and Piotr Didyk. 2023. Gloss-aware Color Correction for 3D Printing. In *ACM SIGGRAPH Conference Papers*.

SN Cramer. 1978. Application of the fictitious scattering radiation transport model for deep-penetration Monte Carlo calculations. *Nuclear Science and Engineering* 65, 2 (1978).

Miguel Crespo, Adrian Jarabo, and Adolfo Muñoz. 2021. Primary-space adaptive control variates using piecewise-polynomial approximations. *ACM Trans. Graph.* 40, 3 (2021).

Oskar Elek, Denis Sumin, Ran Zhang, Tim Weyrich, Karol Myszkowski, Bernd Bickel, Alexander Wilkie, and Jaroslav Křivánek. 2017. Scattering-Aware Texture Reproduction for 3D Printing. *ACM Trans. Graph.* 36, 6 (2017).

Luca Fascione, Johannes Hanika, Mark Leone, Marc Droske, Jorge Schwarzhaupt, Tomáš Davidovič, Andrea Weidlich, and Johannes Meng. 2018. Manuka: A Batch-Shading Architecture for Spectral Path Tracing in Movie Production. *ACM Trans. Graph.* 37, 3 (2018).

Julian Fong, Magnus Wrenninge, Christopher Kulla, and Ralf Habel. 2017. Production volume rendering. In *ACM SIGGRAPH 2017 Courses*.

Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. 2022. Plenoxels: Radiance fields without neural networks. In *Proceedings of CVPR*.

Mathieu Galtier, Stephane Blanco, Cyril Caliot, Christophe Coustet, Jérémi Dauchet, Mouna El Hafi, Vincent Eymet, Richard Fournier, Jacques Gautrais, Anais Khuong, et al. 2013. Integral formulation of null-collision Monte Carlo algorithms. *Journal of Quantitative Spectroscopy and Radiative Transfer* 125 (2013).

Iliyan Georgiev, Thiago Ize, Mike Farnsworth, Ramón Montoya-Vozmediano, Alan King, Brecht Van Lommel, Angel Jimenez, Oscar Anson, Shinji Ogaki, Eric Johnston, Adrien Herubel, Declan Russell, Frédéric Servant, and Marcos Fajardo. 2018. Arnold: A Brute-Force Production Path Tracer. *ACM Trans. Graph.* 37, 3 (2018).

Iliyan Georgiev, Zackary Misso, Toshiya Hachisuka, Derek Nowrouzezahrai, Jaroslav Křivánek, and Wojciech Jarosz. 2019. Integral formulations of volumetric transmittance. *ACM Trans. Graph.* 38, 6 (2019).

Ioannis Gkioulekas, Anat Levin, and Todd Zickler. 2016. An evaluation of computational imaging techniques for heterogeneous inverse scattering. In *Proceedings of ECCV*.

Paul Green, Jan Kautz, Wojciech Matusik, and Frédo Durand. 2006. View-dependent precomputed light transport using nonlinear gaussian function approximations. In *Proceedings of I3D*.

Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. 2021. Baking neural radiance fields for real-time view synthesis. In *Proceedings of ICCV*.

Eric Heitz, Jonathan Dupuy, Cyril Crassin, and Carsten Dachsbacher. 2015. The SGGX microflake distribution. *ACM Trans. Graph.* 34, 4 (2015).

Wenzel Jakob. 2010. Mitsuba renderer.

Wenzel Jakob, Adam Arbree, Jonathan T Moon, Kavita Bala, and Steve Marschner. 2010. A radiative transfer framework for rendering materials with anisotropic structure. *ACM Trans. Graph.* 29, 4 (2010).

Wenzel Jakob, Christian Regg, and Wojciech Jarosz. 2011. Progressive Expectation–Maximization for Hierarchical Volumetric Photon Mapping. *Computer Graphics Forum* 30, 4 (2011).

Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, Merlin Nimier-David, Delio Vicini, Tizian Zeltner, Baptiste Nicolet, Miguel Crespo, Vincent Leroy, and Ziyi Zhang. 2022b. *Mitsuba 3 renderer*. https://mitsuba-renderer.org.

Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, and Delio Vicini. 2022a. Dr.Jit: A Just-In-Time Compiler for Differentiable Rendering. *ACM Trans. Graph.* 41, 4 (2022).

Adrian Jarabo, Carlos Aliaga, and Diego Gutierrez. 2018. A radiative transfer framework for spatially-correlated materials. *ACM Trans. Graph.* 37, 4 (2018).

James T. Kajiya. 1986. The Rendering Equation. *SIGGRAPH Comput. Graph.* 20, 4 (1986).

Simon Kallweit, Thomas Müller, Brian Mcwilliams, Markus Gross, and Jan Novák. 2017. Deep Scattering: Rendering Atmospheric Clouds with Radiance-Predicting Neural Networks. *ACM Trans. Graph.* 36, 6 (2017).

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Trans. Graph.*

42, 4 (2023).
Markus Kettunen, Eugene D'Eon, Jacopo Pantaleoni, and Jan Novák. 2021. An unbiased ray-marching transmittance estimator. *ACM Trans. Graph.* 40, 4 (2021).
Pramook Khungurn, Daniel Schroeder, Shuang Zhao, Kavita Bala, and Steve Marschner. 2015. Matching Real Fabrics with Micro-Appearance Models. *ACM Trans. Graph.* 35, 1 (2015).
Doyub Kim, Minjae Lee, and Ken Museth. 2024. NeuralVDB: High-resolution Sparse Volume Representation using Hierarchical Neural Networks. *ACM Trans. Graph.* 43, 2 (2024).
Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980
David Kirk. 2007. NVIDIA CUDA software and GPU parallel computing architecture. In *Proceedings of ISMM*. 2 pages.
Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. 2017. Tanks and Temples: Benchmarking Large-Scale Scene Reconstruction. *ACM Trans. Graph.* 36, 4 (2017).
Aaron Knoll, Gregory P. Johnson, and Johannes Meng. 2021. Path Tracing RBF Particle Volumes. In *Ray Tracing Gems II*, Adam Marrs, Peter Shirley, and Ingo Wald (Eds.). Apress.
Peter Kutz, Ralf Habel, Yining Karl Li, and Jan Novák. 2017. Spectral and decomposition tracking for rendering heterogeneous volumes. *ACM Trans. Graph.* 36, 4 (2017), 1–16.
Thomas Leimkühler, Hans-Peter Seidel, and Tobias Ritschel. 2018. Laplacian Kernel Splatting for Efficient Depth-of-field and Motion Blur Synthesis or Reconstruction. *ACM Trans. Graph.* 37, 4 (2018).
Boning Liu, Yan Zhao, Xiaomeng Jiang, and Shigang Wang. 2021. Three-dimensional Epanechnikov mixture regression in image coding. *Signal Processing* 185 (2021).
Stephen Lombardi, Tomas Simon, Gabriel Schwartz, Michael Zollhoefer, Yaser Sheikh, and Jason Saragih. 2021. Mixture of Volumetric Primitives for Efficient Neural Rendering. *ACM Trans. Graph.* 40, 4 (2021).
Guillaume Loubet and Fabrice Neyret. 2017. Hybrid mesh-volume LoDs for all-scale pre-filtering of complex 3D assets. *Computer Graphics Forum* 36, 2 (2017).
Nelson L Max. 1979. ATOMLLL: ATOMS with shading and highlights. *ACM SIGGRAPH Computer Graphics* 13, 2 (1979).
Johannes Meng, Marios Papas, Ralf Habel, Carsten Dachsbacher, Steve Marschner, Markus Gross, and Wojciech Jarosz. 2015. Multi-Scale Modeling and Rendering of Granular Materials. *ACM Trans. Graph.* 34, 4 (2015).
Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *Proceedings of ECCV*.
Zackary Misso, Yining Karl Li, Brent Burley, Daniel Teece, and Wojciech Jarosz. 2023. Progressive null-tracking for volumetric rendering. In *ACM SIGGRAPH Conference Papers*.
Jonathan T. Moon, Bruce Walter, and Stephen R. Marschner. 2007. Rendering Discrete Random Media Using Precomputed Scattering Solutions. In *Proceedings of EGSR*.
T.K. Moon. 1996. The expectation-maximization algorithm. *IEEE Signal Processing Magazine* 13, 6 (1996).
Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM Trans. Graph.* 41, 4 (2022).
Thomas Müller, Marios Papas, Markus Gross, Wojciech Jarosz, and Jan Novák. 2016. Efficient Rendering of Heterogeneous Polydisperse Granular Media. *ACM Trans. Graph.* 35, 6 (2016).
Adolfo Muñoz. 2014. Higher order ray marching. *Computer Graphics Forum* 33, 8 (2014).
Ken Museth. 2013. VDB: High-resolution sparse volumes with dynamic topology. *ACM Trans. Graph.* 32, 3 (2013).
Ken Museth. 2021. NanoVDB: A GPU-friendly and portable VDB data structure for real-time rendering and simulation. In *ACM SIGGRAPH 2021 Talks*.
Fabrice Neyret. 1998. Modeling, Animating, and Rendering Complex Scenes Using Volumetric Textures. *IEEE Trans. Vis. Comput. Graph.* 4, 1 (1998).
Merlin Nimier-David, Thomas Müller, Alexander Keller, and Wenzel Jakob. 2022. Unbiased Inverse Volume Rendering with Differential Trackers. *ACM Trans. Graph.* 41, 4 (2022).
Thomas Klaus Nindel, Tomáš Iser, Tobias Rittig, Alexander Wilkie, and Jaroslav Křivánek. 2021. A gradient-based framework for 3D print appearance optimization. *ACM Trans. Graph.* 40, 4 (2021).
Jan Novák, Iliyan Georgiev, Johannes Hanika, and Wojciech Jarosz. 2018. Monte Carlo methods for volumetric light transport simulation. *Computer Graphics Forum* 37, 2 (2018).
Jan Novák, Andrew Selle, and Wojciech Jarosz. 2014. Residual ratio tracking for estimating attenuation in participating media. *ACM Trans. Graph.* 33, 6 (2014).
Steven G. Parker, James Bigler, Andreas Dietrich, Heiko Friedrich, Jared Hoberock, David Luebke, David McAllister, Morgan McGuire, Keith Morley, Austin Robison, and Martin Stich. 2010. OptiX: A General Purpose Ray Tracing Engine. *ACM Trans. Graph.* 29, 4 (2010).
K. Perlin and E. M. Hoffert. 1989. Hypertexture. In *Proceedings of SIGGRAPH*. 10 pages.
Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2023. *Physically based rendering: From theory to implementation.* MIT Press.
Matthias Raab, Daniel Seibert, and Alexander Keller. 2006. Unbiased global illumination with participating media. In *Monte Carlo and Quasi-Monte Carlo Methods 2006*.
Christian Reiser, Rick Szeliski, Dor Verbin, Pratul Srinivasan, Ben Mildenhall, Andreas Geiger, Jon Barron, and Peter Hedman. 2023. Merf: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. *ACM Trans. Graph.* 42, 4 (2023).
Miguel Sainz and Renato Pajarola. 2004. Point-based rendering techniques. *Comput. Graph.* 28, 6 (2004).
Kai Schröder, Reinhard Klein, and Arno Zinke. 2011. A Volumetric Approach to Predictive Rendering of Fabrics. *Computer Graphics Forum* 30, 4 (2011).
Jerome Spanier. 1966. Two pairs of families of estimators for transport problems. *SIAM J. Appl. Math.* 14, 4 (1966).
Jerome Spanier and Ely M Gelbard. 2008. *Monte Carlo principles and neutron transport problems.* Courier Corporation.
Denis Sumin, Tobias Rittig, Vahid Babaei, Tim Weyrich, Thomas Nindel, Piotr Didyk, Bernd Bickel, Jaroslav Křivánek, Alexander Wilkie, and Karol Myszkowski. 2019. Geometry-Aware Scattering Compensation for 3D Printing. *ACM Trans. Graph.* 38, 4 (2019).
T M Sutton, F B Brown, F G Bischoff, D B MacMillan, C L Ellis, J T Ward, C T Ballinger, D J Kelly, and L Schindler. 1999. The Physical Models and Statistical Procedures Used in the RACER Monte Carlo Code. (1999). https://doi.org/10.2172/767449
László Szirmay-Kalos, Iliyan Georgiev, Milán Magdics, Balázs Molnár, and Dávid Légrády. 2017. Unbiased light transport estimators for inhomogeneous participating media. *Computer Graphics Forum* 36, 2 (2017).
László Szirmay-Kalos, Balázs Tóth, and Milán Magdics. 2011. Free path sampling in high resolution inhomogeneous participating media. *Computer Graphics Forum* 30, 1 (2011).
Heang K Tuy and Lee Tan Tuy. 1984. Direct 2-D display of 3-D objects. *IEEE Computer Graphics and Applications* 4, 10 (1984).
Delio Vicini, Wenzel Jakob, and Anton Kaplanyan. 2021a. A non-exponential transmittance model for volumetric scene representations. *ACM Trans. Graph.* 40, 4 (2021).
Delio Vicini, Sébastien Speierer, and Wenzel Jakob. 2021b. Path Replay Backpropagation: Differentiating Light Paths using Constant Memory and Linear Time. *ACM Trans. Graph.* 40, 4 (2021).
Jiří Vorba, Ondřej Karlík, Martin Šik, Tobias Ritschel, and Jaroslav Křivánek. 2014. On-line learning of parametric mixture models for light transport simulation. *ACM Trans. Graph.* 33, 4 (2014).
Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing* 13, 4 (2004).
E Woodcock. 1965. Techniques used in the GEM code for Monte Carlo neutronics calculations in reactors and other systems of complex geometry. In *Proceedings of the Conference on Applications of Computing Methods to Reactor Problems*, Vol. 557.
Kun Xu, Wei-Lun Sun, Zhao Dong, Dan-Yong Zhao, Run-Dong Wu, and Shi-Min Hu. 2013. Anisotropic Spherical Gaussians. *ACM Trans. Graph.* 32, 6 (2013).
Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. 2022. Point-nerf: Point-based neural radiance fields. In *Proceedings of CVPR*.
Ling-Qi Yan, Miloš Hašan, Steve Marschner, and Ravi Ramamoorthi. 2016. Position-normal distributions for efficient rendering of specular microstructure. *ACM Trans. Graph.* 35, 4 (2016).
Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. 2021. PlenOctrees for Real-time Rendering of Neural Radiance Fields. In *Proceedings of ICCV*.
Yonghao Yue, Kei Iwasaki, Bing-Yu Chen, Yoshinori Dobashi, and Tomoyuki Nishita. 2010. Unbiased, adaptive stochastic sampling for rendering inhomogeneous participating media. *ACM Trans. Graph.* 29, 6 (2010).
Cheng Zhang, Lifan Wu, Changxi Zheng, Ioannis Gkioulekas, Ravi Ramamoorthi, and Shuang Zhao. 2019. A Differential Theory of Radiative Transfer. *ACM Trans. Graph.* 38, 6 (2019).
Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. 2020. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492* (2020).
Shuang Zhao, Wenzel Jakob, Steve Marschner, and Kavita Bala. 2011. Building Volumetric Appearance Models of Fabric Using Micro CT Imaging. *ACM Trans. Graph.* 30, 4 (2011).
Shaung Zhao, Lifan Wu, Frédo Durand, and Ravi Ramamoorthi. 2016. Downsampling Scattering Parameters for Rendering Anisotropic Media. *ACM Trans. Graph.* 35, 6 (2016).
Junqiu Zhu, Yaoyi Bai, Zilin Xu, Steve Bako, Edgar Velázquez-Armendáriz, Lu Wang, Pradeep Sen, Miloš Hašan, and Ling-Qi Yan. 2021. Neural complex luminaires: representation and rendering. *ACM Trans. Graph.* 40, 4 (2021).
M. Zwicker, H. Pfister, J. van Baar, and M. Gross. 2001. EWA volume splatting. In *Proceedings of Visualization*.

A SINGLE-SCATTERING ALBEDO AND PHASE FUNCTION

For completeness, here we extend the mixture-based optical properties' definition introduced in Section 4 to kernel-mixture-based single-scattering albedo and phase function. They are computed similarly to the extinction coefficient in Equation (6), with the key difference of requiring normalization. Thus, for a set of N primitives, the single-scattering albedo $\alpha(\mathbf{x})$ and phase function $f_\mathrm{p}(\mathbf{x}, \omega' \to \omega)$ are defined as

$$\alpha(\mathbf{x}) = \sum_{i=1}^{N} \frac{\sigma_i K_i(\mathbf{x}) \alpha_i}{\sum_{i=1}^{N} \sigma_i K_i(\mathbf{x})}, \tag{25}$$

$$f_\mathrm{p}(\mathbf{x}, \omega' \to \omega) = \sum_{i=1}^{N} \frac{\sigma_i K_i(\mathbf{x})\, \alpha_i\, f_{\mathrm{p},i}(\omega' \to \omega)}{\alpha(\mathbf{x})}, \tag{26}$$

respectively, with α_i and $f_{\mathrm{p},i}(\omega' \to \omega)$ the single-scattering albedo and phase function of primitive $\mathcal{P}_i$.

B KERNEL TRANSMITTANCE EXPRESSIONS

B.1 Gaussian kernel

Assuming our ray direction normalized and whitened (linearly transformed to the local space of the Gaussian shell's ellipsoid, where it is a linear sphere, we can compute the cumulative distribution function (CDF) of a single weighted 3D Gaussian primitive along the given ray as

$$\tau_i(\mathbf{x}_{-t_s}, \mathbf{x}_{t_s}) = \sigma_i \int_{-t_s}^{t_s} g_i(\mathbf{x}_t)\, \mathrm{d}t = G e^{-\frac{1}{2}(a-b^2)} \left[\mathrm{erf}\left(t_s \sqrt{\frac{1}{2}} \right) \right] \tag{27}$$

with

$$a = \mathbf{x}_0 \times \mathbf{x}_0, \quad b = \mathbf{x}_0 \times \omega, \quad G = \frac{\sigma_i}{2\pi \sqrt{v^T \Sigma_i^{-1} v |\Sigma_i|}} \tag{28}$$

This equation will be plugged in Equations (9) and (11) to compute the contribution of each overlapping primitive to the accumulated transmittance along the ray. However, to simplify this equation and accelerate rendering, we can resort to integrating along the whole domain of the ray $t_0 = -\infty, t_1 = \infty$ when no overlaps are detected between the integration limits (shell) of a given primitive. This further simplifies the integral to

$$\tau_i(\mathbf{x}_{-\infty}, \mathbf{x}_{\infty}) = G e^{-\frac{1}{2}(a-b^2)} \tag{29}$$

Epanechnikov kernel Again, assuming our ray direction normalized, we can compute the CDF of a single 3D Epanechnikov primitive along the given ray as

$$\tau_i(\mathbf{x}_{-t_s}, \mathbf{x}_{t_s}) = \frac{15}{8\pi (7^3 |\Sigma|)^{\frac{1}{2}}} \int_{t_0}^{t_1} \left[1 - \frac{1}{7} \left(\frac{(x_{0,x} + \omega_x t)^2}{S_x^2} + \frac{(x_{0,y} + \omega_y t)^2}{S_y^2} + \frac{(x_{0,z} + \omega_z t)^2}{S_z^2} \right) \right] \mathrm{d}t = K_\mathrm{norm} (\mathcal{K}_3 (t_0^3 - t_1^3) + \mathcal{K}_2 (t_0^2 - t_1^2) + \mathcal{K}_1 (t_0 - t_1)), \tag{30}$$

with

$$\mathcal{K}_1 = 3((x_{0,x}^2 - 7S_x^2) S_y^2 + x_{0,y}^2 S_x^2) S_z^2 + x_{0,z}^2 S_x^2 S_y^2),$$
$$\mathcal{K}_2 = 3(x_{0,z} S_x^2 S_y^2 \omega_z + x_{0,y} S_x^2 S_z^2 \omega_y + x_{0,x} S_y^2 S_z^2 \omega_x),$$
$$\mathcal{K}_3 = S_x^2 S_y^2 \omega_z^2 + S_x^2 S_z^2 \omega_y^2 + S_y^2 S_z^2 \omega_x^2$$
$$\mathcal{K}_\mathrm{norm} = \frac{15}{168\pi 7^{\frac{3}{2}} |\Sigma|^2},$$

and t_0, t_1 defining the segment along the ray being integrated.

C INTEGRATOR ADJOINTS

Here we include the adjoints for both the VPPT and VPRF integrators. For simplicity, in the following formulas we omit the set of kernel primitives for all summations and products, as well as the positional dependence.

Adjoint VPPT. The derivative of Equation (21) is

$$\begin{aligned} \delta L = {} & \delta(\beta) \cdot \alpha_S \cdot \mathrm{T}_S \cdot \mu_S \cdot L_i + \beta \cdot \delta(\alpha_S) \cdot \mathrm{T}_S \cdot \mu_S \cdot L_i \\ & + \beta \cdot \alpha_S \cdot \delta(\mathrm{T}_S) \cdot \mu_S \cdot L_i + \beta \cdot \alpha_S \cdot \mathrm{T}_S \cdot \delta(\mu_S) \cdot L_i \\ & + \beta \cdot \alpha_S \cdot \mathrm{T}_S \cdot \mu_S \cdot \delta(L_i) \end{aligned}$$

with

$$\delta\alpha_S = \sum_i \left(\delta(\alpha_i p_i) \cdot \frac{1}{\sum_j p_j} - \delta(p_i) \cdot \frac{1}{(\sum_j p_j)^2} \cdot \sum_j \alpha_j p_j \right),$$
$$\delta T_S = \sum_i \delta(e^{-\tau_i}) \cdot \frac{(\prod_j e^{-\tau_j})}{e^{\tau_i}},$$
$$\delta\mu_S = \sum_i \delta\mu_i,$$

and $p_i = \sigma_i K_i(\mathbf{x})$.

Adjoint VPRF. The derivative of Equation (7) is

$$\delta L = \sum_{k=1}^{M} \delta(\mathrm{T}_{k-1}) L_k + \mathrm{T}_{k-1} \delta(L_k),$$

with for the VPRF integrator

$$\delta \mathrm{T}_k = \sum_{j=0}^{k} \delta(\mathrm{T}_j) \frac{\prod_{l=0}^{k} \mathrm{T}_l}{\mathrm{T}_j},$$
$$\delta L_k = \delta \left(1 - \prod_{i \in S_k} e^{-\tau_i} \right) \sum_{i \in S_k} \left(\overline{Q}_i \frac{\tau_i}{\sum_j \tau_j} \right) + \left(1 - \prod_{i \in S_k} e^{-\tau_i} \right) \sum_{i \in S_k} \delta \left(\overline{Q}_i \frac{\tau_i}{\sum_j \tau_j} \right),$$

where

$$\delta\left(1-\prod_{i\in S_k} e^{-\tau_i}\right) = \sum_{i\in S_k} \frac{-\delta(e^{-\tau_i})}{e^{-\tau_i}} \prod_{j\in S_k} \delta(e^{-\tau_j}),$$

$$\delta\left(\bar{Q}_i \frac{\tau_i}{\sum_j \tau_j}\right) = \delta\bar{Q}_i \frac{\tau_i}{\sum_j \tau_j} + \bar{Q}_i \frac{\delta(\tau_i)}{\sum_j \tau_j} + \bar{Q}_i \frac{\tau_i \sum_j \delta(\tau_j)}{(\sum_j \tau_j)^2}.$$

Received 17 July 2024; revised 1 November 2024; accepted 2 December 2024