

8DNA: 8D Neural Asset Light Transport by Distribution Learning

LIWEN WU, University of California San Diego, USA
HAOLIN LU, University of California San Diego, USA
BING XU, University of California San Diego, USA and NVIDIA, USA
MILOŠ HAŠAN, NVIDIA, USA
RAVI RAMAMOORTHY, University of California San Diego, USA

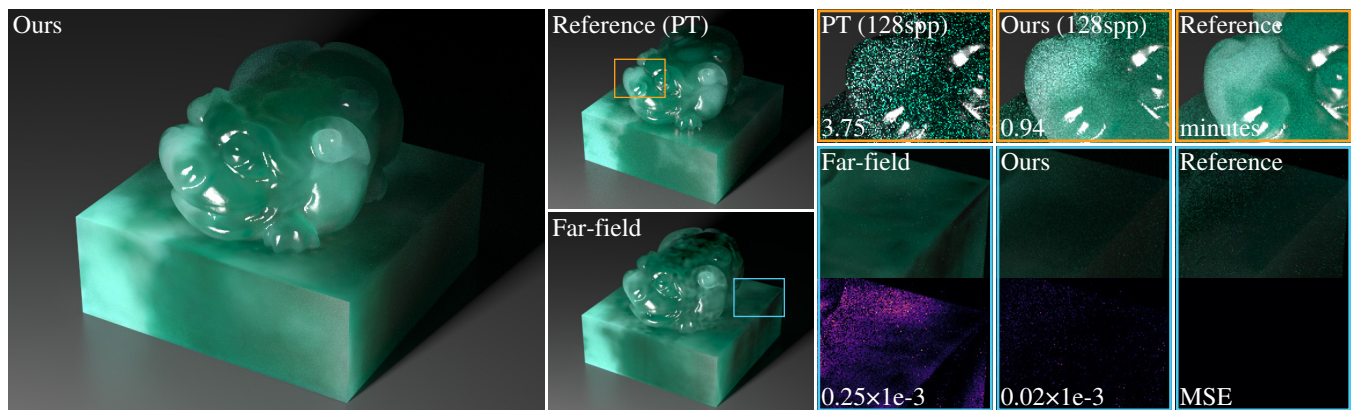


Fig. 1. **Our method vs. the baselines.** Our method pre-bakes neural assets with complex light transport—including any types of scatterings from volumes to surfaces—that can be imported between renderers for physically based rendering. This yields much lower rendering variance (insets on top) and faster inference speed (numbers in minutes) than simulating the light transport online with standard path tracing (PT). In contrast to previous regression-based pre-baking (far-field) that is limited to a far-field approximation of the light transport, our representation learns the distribution of full 8D light transport from path-traced samples, correctly reproducing the occlusion effects on the back of the seal (insets at bottom).

High-fidelity 3D assets exhibit intriguing global illumination effects like subsurface scattering, glossy interreflections, and fine-scale fiber scatterings, which often involve long scattering paths that are expensive to simulate. We introduce 8D neural assets (8DNA) to pre-bake these light transport effects into neural representations. Unlike prior methods that assume far-field lighting and precompute light transport into 6D functions, 8DNA learns the full 8D light transport, enabling accurate rendering under near-field illumination. Our training leverages a distribution-learning formulation that learns light transport from forward path-traced samples, which produces less optimization variance with lower training budget than the prior regression-based approaches. Experiments show our 8DNA rendering closely matches path-traced results under various scene configurations, yet it achieves improved variance reduction and fast inference speeds on challenging assets.

CCS Concepts: • **Computing methodologies** → **Ray tracing; Reflectance modeling.**

Additional Key Words and Phrases: Neural rendering, global illumination

Authors' Contact Information: Liwen Wu, University of California San Diego, USA, liw026@ucsd.edu; Haolin Lu, University of California San Diego, USA, hal128@ucsd.edu; Bing Xu, University of California San Diego, USA and NVIDIA, USA, b4xu@ucsd.edu; Miloš Hašan, NVIDIA, USA, milos.hasan@gmail.com; Ravi Ramamoorthy, University of California San Diego, USA, ravir@cs.ucsd.edu.



This work is licensed under a Creative Commons Attribution 4.0 International License. *SIGGRAPH Conference Papers '26, Los Angeles, CA, USA*
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2554-8/2026/07
<https://doi.org/10.1145/3799902.3811094>

ACM Reference Format:

Liwen Wu, Haolin Lu, Bing Xu, Miloš Hašan, and Ravi Ramamoorthy. 2026. 8DNA: 8D Neural Asset Light Transport by Distribution Learning. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '26)*, July 19–23, 2026, Los Angeles, CA, USA. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/3799902.3811094>

1 Introduction

Real-world 3D objects often exhibit interesting and complex light transport effects, such as multiple scattering within media enclosed by refractive boundaries, glossy interreflections, fine-scale fiber scattering in hair, fur or fabrics, and more. These effects are challenging to render in digital 3D assets: they often involve long light paths that are non-trivial to trace from either the camera or the light source, making them expensive to simulate with standard Monte Carlo methods. Furthermore, the appearance models responsible for the above effects can be non-trivial to consistently implement and precisely match between different rendering systems.

Recent work [Mullia et al. 2024; Tg et al. 2024a] has explored a promising alternative: precompute the light transport within such 3D assets into a neural representation, which can be rendered more efficiently, and transferred between renderers more easily. These methods propose to bake all light paths within a 3D asset into a relightable neural network, which can be evaluated for any point on the asset and any view and light direction. However, such parameterizations are 6-dimensional: they fundamentally assume distant

illumination. While they can still be approximately used under near-field lighting, we show that this can lead to inaccuracy.

The true light transport operator describing light paths through the 3D asset is an 8D function $F(\mathbf{x}_o, \omega_o, \mathbf{x}_i, \omega_i)$: it depends on both outgoing (camera) direction ω_o and position \mathbf{x}_o , as well as their incoming (light) counterparts ω_i and \mathbf{x}_i . This 8D function is not only challenging to compress, but its queries are non-trivial to estimate with low variance, especially in the most interesting cases: appearances that feature highly specular surface or fiber interactions, and/or forward-scattering phase functions. Therefore, unlike the 6D functions in Mullia et al. [2024] and Tg et al. [2024a], it is difficult to train the full 8D transport with a simple regression loss.

In this paper, we propose to recover the full 8D light transport in a distribution-learning framework. From the forward path samples through the asset, we estimate the global scattering distribution using a normalizing flow and predict the survival probability (albedo term), together giving a natural decomposition of the global light transport distribution F .

In contrast to the challenging direct evaluations of F , sampling F is as easy as standard path tracing through the asset without next-event estimation. Such efficiency allows our model to train faster than regression-based baselines while better capturing 8D light transport including accurate near-field illumination. As a further benefit, the learned representation can be evaluated *and* importance-sampled, with the sampling distribution by construction precisely proportional to the transport. Similar arguments are made in concurrent work on precomputing multiple scattering from microgeometry [Li et al. 2025]; however, this work does not target 3D assets and still assumes 6D far-field light transport.

Figure 1 demonstrates that our 8D neural asset (8DNA) successfully reconstructs the global illumination of the jade seal lit by a nearby area emitter: a challenging configuration for the baselines. Our representation is particularly effective for objects with long scattering paths, such as translucent assets with complex interiors, achieving noticeable variance reduction compared to standard path tracing (Sec. 4). In summary, our contributions are:

- An efficient and accurate 8D neural asset representation of arbitrary 3D assets with complex internal light transport.
- A forward sampling training framework of our representation, whose implementation complexity and training cost are as low as standard forward path tracing without next-event estimation.
- Results showing 8DNA assets rendered within a final scene at an accuracy closely matching ground truth, without tracing long light paths or needing to implement the original light scattering models in the deployment renderer.

2 Preliminaries and Related Work

8D near-field light transport. Recall our goal of precomputing global light transport within a 3D asset consisting of surfaces, volumes, fibers, etc. We assume the asset is non-emissive and bounded by a 2D surface \mathcal{M} (support for volumes without boundary is discussed in Sec. 4.3). For such assets, light scattering can be encapsulated by a light transport function $F(\mathbf{x}_o, \omega_o, \mathbf{x}_i, \omega_i)$ that maps incident radiance L_i on any boundary point to outgoing radiance L_o at

another boundary point; integrations are taken over each variable’s domain of definition ($\mathbf{x}_i \in \mathcal{M}$, $\omega_i \in \text{sphere}$) unless specified:

$$L_o(\mathbf{x}_o, \omega_o; L_i) = \iint F(\mathbf{x}_o, \omega_o, \mathbf{x}_i, \omega_i) L_i(\mathbf{x}_i, \omega_i) d\mathbf{x}_i d\omega_i. \quad (1)$$

Note that L_i excludes the internal multiple-scattering within the asset, and we will use the fact that L_o is computable by forward path tracing given any L_i . Our objective is to learn the light transport F for a given asset, in the form of a neural representation, to enable efficient re-rendering under arbitrary illumination L_i .

The 8D transport F , or its variants, are sometimes called a BSSRDF (bidirectional subsurface scattering reflectance distribution function) in the literature. We will avoid this term, as it unnecessarily constrains the concept to subsurface scattering, while our solution generalizes to any other kinds of scattering.

Global light transport with a far-field assumption assumes incident light is sufficiently distant to depend only on ω_i . In this case, F can be further pre-integrated into a 6D transport $F'(\mathbf{x}_o, \omega_o, \omega_i)$, invariant to the incident position:

$$\begin{aligned} L_o(\mathbf{x}_o, \omega_o; L_i) &= \iint F(\mathbf{x}_o, \omega_o, \mathbf{x}_i, \omega_i) L_i(\omega_i) d\mathbf{x}_i d\omega_i \\ &= \int \left(\int F(\mathbf{x}_o, \omega_o, \mathbf{x}_i, \omega_i) d\mathbf{x}_i \right) L_i(\omega_i) d\omega_i \\ &= \int F'(\mathbf{x}_o, \omega_o, \omega_i) L_i(\omega_i) d\omega_i. \end{aligned} \quad (2)$$

Many far-field neural representations of this kind have been proposed in previous work. On flat materials, Kuznetsov et al. [2021] learn a spatially varying reflectance with a neural network F'_θ by minimizing an L2 regression loss between F'_θ and F' :

$$\iiint (F'_\theta(\mathbf{x}_o, \omega_o, \omega_i) - F'(\mathbf{x}_o, \omega_o, \omega_i))^2 d\omega_i d\mathbf{x}_o d\omega_o, \quad (3)$$

where the integral is estimated by sampling random $(\mathbf{x}_o, \omega_o, \omega_i)$ queries, and the ground truth F' is obtained via standard path tracing under a directional delta light source $L_i(\omega) = \delta(\omega - \omega_i)$ (using the fact $F'(\mathbf{x}_o, \omega_o, \omega_i) = \int F'(\mathbf{x}_o, \omega_o, \omega) \delta(\omega - \omega_i) d\omega$). Similar ideas have been applied to layered materials [Fan et al. 2022; Sztrajman et al. 2021; Zeltner et al. 2024], curved surfaces [Kuznetsov et al. 2022], 3D assets with interreflection and scattering effects [Mullia et al. 2024], and to full-sphere incident directions for translucent objects [Tg et al. 2024a]. Note, this paper uses near-field specifically to refer to multi-bounce light transport rather than local BSDF models.

While the far-field assumption leads to a simple and effective learning step, the resulting 6D model cannot accurately capture the near-field illumination effects encoded in the full transport F of Eq. (1). A direct extension would be to train an 8D network F_θ with the same regression scheme, now additionally sampling \mathbf{x}_i and path tracing ground truth under a doubly delta illumination $L_i(\mathbf{x}, \omega) = \delta(\mathbf{x} - \mathbf{x}_i) \delta(\omega - \omega_i)$, but the combination of higher-dimensional sampling and strongly peaked illumination dramatically increases the difficulty of designing efficient Monte Carlo estimators. While Tg et al. [2024b] partially mitigates this variance by learning a light transport function over a collection of incident illumination samples, it relies on an isotropic assumption for internal scattering (see supplementary), and its rendering requires tracing

multiple incident rays per outgoing ray that does not easily fit into the path tracing pipeline. These limitations motivate our shift to a distribution-learning-based training objective.

Distribution learning in rendering is most commonly used for importance sampling. These methods learn to sample a certain term of the rendering equation, including BSDF sampling [Rainer et al. 2020; Xie et al. 2019; Xu et al. 2023], product sampling the BSDF times emission [Clarberg and Akenine-Möller 2008; Hart et al. 2020; Herholz et al. 2016; Litalien et al. 2024], and path guiding from the approximated incident radiance [Diolatzis et al. 2020; Dodik et al. 2022; Müller et al. 2017; Vorba et al. 2014]. The distributions are constructed from analytic mixtures or warps in classic approaches. In neural methods, normalizing flows [Dinh et al. 2014; Durkan et al. 2019; Müller et al. 2019; Rezende and Mohamed 2015] are widely used, yet still predict analytic mixtures for their efficiency [Fan et al. 2022; Sztajman et al. 2021; Xu et al. 2023; Zeltner et al. 2024]. Recent works further explore flow-based models [Heitz et al. 2023; Lipman et al. 2022] for BSDF sampling [Fu et al. 2024] and precomputing multiple scattering [Li et al. 2025]. A reparameterization based neural sampling strategy is proposed in [Wu et al. 2025] without using generative neural networks.

Pre-computed radiance transfer (PRT) also builds upon the far-field light transport formulation of Eq. (2) but focuses on aggressively compressing both L_i and \mathcal{F} [Ramamoorthi et al. 2009]. This is typically achieved by basis function expansions using spherical harmonics [Ramamoorthi and Hanrahan 2001; Sloan et al. 2002], wavelets [Ng et al. 2003, 2004], polynomials [Ben-Artzi et al. 2008], Gaussians [Lu et al. 2025; Tsai and Shih 2006; Xu et al. 2013], or neural bases [Rainer et al. 2022; Xu et al. 2022]. Extensions to near-field polygonal area lights have also been derived using spherical harmonics [Wang and Ramamoorthi 2018; Wu et al. 2020]. In contrast to path tracing, PRT is primarily designed for deterministic, real-time evaluation under relatively simple illumination configurations, rather than for the high-fidelity, scalable simulation of complex light transport that we target.

Multiple-scattering by BSSRDF. BSSRDFs have been widely used to model specific multiple-scattering effects. Jensen et al. [2001] first introduce an analytic BSSRDF for subsurface scattering in homogeneous media under flat surfaces. Subsequent works refine this formulation [Donner and Jensen 2006; Donner et al. 2009; Habel et al. 2013], extend it to heterogeneous media [Donner et al. 2008; Peers et al. 2006], and employ neural representations to better capture shape-dependent behavior [Vicini et al. 2019]. Similar BSSRDFs have also been used to model scattering inside translucent objects [Deng et al. 2022] and fur and hair [Yan et al. 2017]. These methods typically adopt simplified BSSRDF forms (e.g., separable parameterizations) and focus on a single class of effects (e.g., subsurface scattering), leaving other components of the light transport (e.g., surface inter-reflections) to be handled by other methods or ignored. In contrast, our 8D light transport makes no simplifications and is designed to capture all scattering events within the asset—across both interior volumes and surfaces—up to the point where light exits the asset.

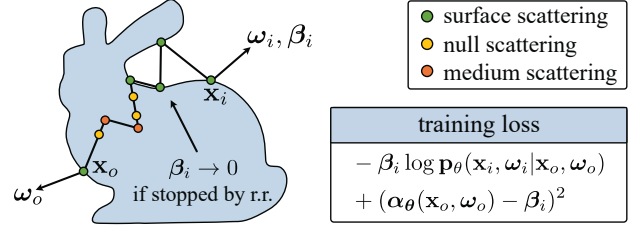


Fig. 2. **Our training** is performed by tracing random outgoing rays to the asset, encountering multiple events of surface/medium/null scattering and Russian Roulette (r.r.) until leaving the asset. The exit ray configurations and throughputs are used to compute the negative log-likelihood loss and albedo regression loss (bottom right).

3 8D Neural Asset

Because the light transport function F is non-negative, we can decompose it to a conditional distribution $\mathbf{p}(\mathbf{x}_i, \omega_i | \mathbf{x}_o, \omega_o)$ and a normalizing factor $\alpha(\mathbf{x}_o, \omega_o)$ for each color channel independently:

$$F(\mathbf{x}_o, \omega_o, \mathbf{x}_i, \omega_i) = \alpha(\mathbf{x}_o, \omega_o) \mathbf{p}(\mathbf{x}_i, \omega_i | \mathbf{x}_o, \omega_o),$$

$$\alpha(\mathbf{x}_o, \omega_o) = \iint F(\mathbf{x}_o, \omega_o, \mathbf{x}_i, \omega_i) d\mathbf{x}_i d\omega_i, \quad \mathbf{p}(\mathbf{x}_i, \omega_i | \mathbf{x}_o, \omega_o) = \frac{F}{\alpha}. \quad (4)$$

Conceptually, α corresponds to the directionally-varying albedo (survival probability) and \mathbf{p} describes the distribution of scattering events. We learn the two components separately: a normalizing flow \mathbf{p}_θ for the scattering distribution and a regular MLP α_θ for the albedo (Sec. 3.1), which together formulate $F_\theta = \alpha_\theta \mathbf{p}_\theta$. Both \mathbf{x}_i defined on the asset boundary and ω_i on the sphere do not naturally fit into standard Euclidean normalizing flows. We therefore apply a reparameterization of \mathbf{p}_θ in Sec. 3.2. The training and inference of our model are described in Sec. 3.3, and we include their pseudocode in the supplemental material.

3.1 Learning the scattering distribution and albedo

We learn both \mathbf{p}_θ and α_θ through stochastic gradient descent optimization. We first discuss the training objectives under a fixed (\mathbf{x}_o, ω_o) then the combined loss for the full 8D domain.

Scattering distribution loss. The common practice in statistical learning suggests \mathbf{p}_θ can be learned by minimizing the negative log-likelihood: $-\iint \mathbf{p} \log \mathbf{p} d\mathbf{x}_i d\omega_i = -\frac{1}{\alpha} \iint F \log \mathbf{p}_\theta d\mathbf{x}_i d\omega_i$. We drop $\frac{1}{\alpha}$ as it is only a constant factor that does not affect the gradient-descent direction, which yields the loss:

$$\mathcal{L}_p(\theta | \mathbf{x}_o, \omega_o) = -\iint F(\mathbf{x}_i, \omega_i, \mathbf{x}_o, \omega_o) \log \mathbf{p}_\theta(\mathbf{x}_i, \omega_i | \mathbf{x}_o, \omega_o) d\mathbf{x}_i d\omega_i. \quad (5)$$

Comparing the right-hand side of Eq. (5) with Eq. (1), we can interpret \mathcal{L}_p as rendering the asset with $-\log \mathbf{p}_\theta$ acting as the "incident light" (can be positive or negative):

$$-\iint F \log \mathbf{p}_\theta d\mathbf{x}_i d\omega_i = \iint F(-\log \mathbf{p}_\theta) d\mathbf{x}_i d\omega_i = L_o(\mathbf{x}_o, \omega_o; -\log \mathbf{p}_\theta).$$

Using the fact that Eq. (1) can be unbiasedly estimated through path tracing, we can compute \mathcal{L}_p in a similar path-tracing style (Fig. 2): the ray (\mathbf{x}_o, ω_o) is traced (path-sampled) for multiple bounces until leaving the asset, whose exit configuration (\mathbf{x}_i, ω_i) and throughput

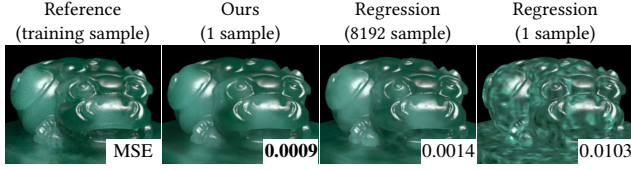


Fig. 3. **Our training vs. regression.** Regressing the ground truth light transport requires many samples per training query to estimate F' (3rd image), which fails in the 1-sample setup (4th image). In contrast, our optimization requires only 1 sample per $(\mathbf{x}_o, \boldsymbol{\omega}_o)$ query (2nd image). The numbers show the mean square error (MSE), and a far-field light is used.

β_i is then used to form the Monte Carlo estimator (\mathbb{E}^n denotes empirical expectation over n samples):

$$\mathcal{L}_p(\theta|\mathbf{x}_o, \boldsymbol{\omega}_o) = \mathbb{E}^n_{\mathbf{x}_i, \boldsymbol{\omega}_i, \beta_i \sim \text{path sampling}} [-\beta_i \log p_\theta(\mathbf{x}_i, \boldsymbol{\omega}_i|\mathbf{x}_o, \boldsymbol{\omega}_o)]. \quad (6)$$

The path-sampling probability does not need to equal \mathbf{p} since β_i gives the correction, and the throughput-based Russian roulette also seamlessly fits here to terminate low-throughput paths early, for which the terminated paths simply contribute $\beta_i = 0$.

Albedo loss. Under the same intuition, the albedo can be interpreted as rendering the asset under constant unit illumination:

$$\alpha(\mathbf{x}_o, \boldsymbol{\omega}_o) = \iint \mathbf{F}(\mathbf{x}_o, \boldsymbol{\omega}_o, \mathbf{x}_i, \boldsymbol{\omega}_i) \times 1 d\mathbf{x}_i d\boldsymbol{\omega}_i = \mathbf{L}_o(\mathbf{x}_o, \boldsymbol{\omega}_o; 1)$$

which can be estimated by averaging the throughput $\alpha_\theta(\mathbf{x}_o, \boldsymbol{\omega}_o) = \mathbb{E}^n[\beta_i]$, reusing the same samples for training p_θ . We fit α_θ against the estimated α using a simple L2 loss:

$$\begin{aligned} \mathcal{L}_\alpha(\theta|\mathbf{x}_o, \boldsymbol{\omega}_o) &= (\alpha_\theta(\mathbf{x}_o, \boldsymbol{\omega}_o) - \alpha(\mathbf{x}_o, \boldsymbol{\omega}_o))^2 \\ &= (\alpha_\theta(\mathbf{x}_o, \boldsymbol{\omega}_o) - \mathbb{E}^n[\beta_i])^2, \end{aligned} \quad (7)$$

whose gradient is linear in α , therefore the empirical expectation gives an unbiased gradient estimator.

Loss variance analysis. Figure 3 shows an example where the regression objective Eq. (3) requires thousands of samples to estimate F' for convergence, whereas our model trained with one sample per outgoing ray already recovers F . This is because the regression gradient $2 \int (\mathbf{F}'_o - \mathbf{F}') \nabla_\theta \mathbf{F}'_o d\boldsymbol{\omega}_i$ lacks an efficient way to both sample $\boldsymbol{\omega}_i$ and estimate F' for arbitrary $\boldsymbol{\omega}_i$, making the optimization especially hard for highly specular light transport induced by dielectric boundaries, fibers, etc. In contrast, our gradient $\nabla_\theta \mathcal{L}_p \propto \iint \mathbf{F} \nabla_\theta \log p_\theta d\mathbf{x}_i d\boldsymbol{\omega}_i$ and $\nabla_\theta \mathcal{L}_\alpha \propto (\alpha_\theta - \iint \mathbf{F} d\mathbf{x}_i d\boldsymbol{\omega}_i) \nabla_\theta \alpha_\theta$ are simple integrals against F . This is what is importance-sampled by the path-sampling procedure, therefore Eqs. (6) and (7) admit empirically low-variance gradient estimators suitable for stochastic optimization. This sampling efficiency ensures our model is trainable under the 8D configuration.

Overall loss. To optimize p_θ, α jointly across all outgoing configurations, we integrate $\mathcal{L}_p + \mathcal{L}_\alpha$ over $\mathbf{x}_o, \boldsymbol{\omega}_o$, estimated by sampling random outgoing rays that intersect the asset (Sec. 3.3), and sum the contributions over color channels c (superscripts denote c -th component of vectors):

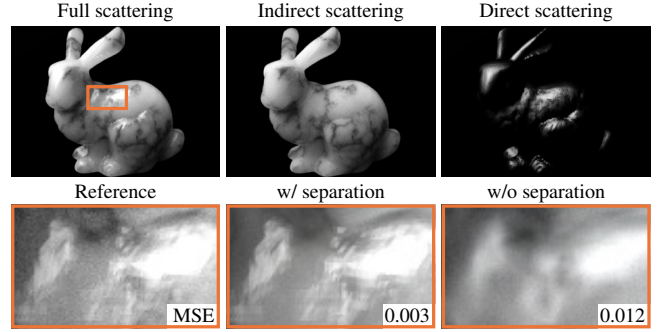


Fig. 4. **Direct-indirect separation of light transport.** The direct scattering above is nearly a delta reflection, while the indirect light transport involves smooth volumetric scattering. Such different behaviors are difficult to model by a single network (w/o separation), so we only model the indirect component combined with analytic direct scattering to improve the accuracy (w/ separation). More direct/indirect examples are in the supplement.

$$\mathcal{L}(\theta) = \sum_{\mathbf{x}_o, \boldsymbol{\omega}_o} \mathbb{E}^n [\mathcal{L}_p^c(\theta|\mathbf{x}_o, \boldsymbol{\omega}_o) + \mathcal{L}_\alpha^c(\theta|\mathbf{x}_o, \boldsymbol{\omega}_o)]. \quad (8)$$

Separating the direct scattering. Translucent objects often have simple but highly specular dielectric boundaries whose direct scattering is much sharper than the rest of the light transport (Fig. 4). To better capture the remaining complexity, we keep the (cosine-weighted) direct BSDF \mathbf{f} untouched and train the network only on the indirect transport $\mathbf{F}(\mathbf{x}_o, \boldsymbol{\omega}_o, \mathbf{x}_i, \boldsymbol{\omega}_i) - V(\mathbf{x}_o, \boldsymbol{\omega}_i)\mathbf{f}(\mathbf{x}_o, \boldsymbol{\omega}_o, \boldsymbol{\omega}_i)$, where V denotes the asset’s self-visibility. This is implemented by using the same loss Eq. (8) while setting $\beta_i = 0$ for path samples that exit the asset after one scattering event. At inference time, $V\mathbf{f}$ is still evaluated analytically, which is usually cheap, yet this direct-indirect separation leads to more accurate matching of the ground truth appearance.

3.2 Realizing the distribution network

Parameterization of network inputs. To obtain a Euclidean parameterization of $(\mathbf{x}_i, \boldsymbol{\omega}_i)$, we trace the incident ray further to its intersection \mathbf{u}_i on the asset’s axis-aligned bounding box (assume centered at origin), inducing the transformed pdf involving the surface normal \mathbf{n}_\square (the subscript indicates the query location):

$$p_\theta(\mathbf{x}_i, \boldsymbol{\omega}_i|\mathbf{x}_o, \boldsymbol{\omega}_o) = p_\theta(\mathbf{u}_i, \boldsymbol{\omega}_i|\mathbf{x}_o, \boldsymbol{\omega}_o) \left| \frac{\mathbf{n}_{\mathbf{x}_i} \cdot \boldsymbol{\omega}_i}{\mathbf{n}_{\mathbf{u}_i} \cdot \boldsymbol{\omega}_i} \right|. \quad (9)$$

This is a valid one-to-one reparameterization as explained in Fig. 5; other proxies such as a bounding sphere could be used similarly. The proxy uv-map gives a general Euclidean parameterization of the intersection, but for a bounding box, we can simply encode \mathbf{u}_i and $\boldsymbol{\omega}_i$ in the asset’s local space to their cylindrical coordinates:

$$\begin{aligned} p_\theta(\mathbf{u}_i, \boldsymbol{\omega}_i|\mathbf{x}_o, \boldsymbol{\omega}_o) &= p_\theta(\mathbf{s}|\mathbf{x}_o, \boldsymbol{\omega}_o) \frac{|\mathbf{n}_{\mathbf{u}_i} \cdot \mathbf{u}_i|}{(\sqrt{\mathbf{u}_i \cdot \mathbf{u}_i})^3}, \\ \mathbf{s} &= (\mathbf{u}_i^3 / \sqrt{\mathbf{u}_i \cdot \mathbf{u}_i}, \arctan(\mathbf{u}_i^1 / \mathbf{u}_i^2), \boldsymbol{\omega}_i^3, \arctan(\boldsymbol{\omega}_i^1 / \boldsymbol{\omega}_i^2)). \end{aligned} \quad (10)$$

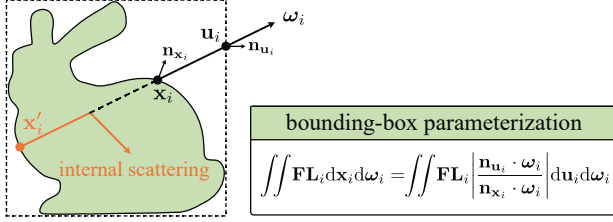


Fig. 5. **Bounding-box parameterization of x_i .** We map x_i along ω_i to its bounding-box intersection u_i and reparameterize Eq. (1) accordingly. While tracing the ray back from u_i may hit multiple points (e.g., x_i, x'_i), only the outermost intersection (x_i) is visible to the incident light and lies in the original integral domain; other points contribute only to internal scattering. Thus, the mapping from (x_i, ω_i) to (u_i, ω_i) is injective, assuming incident illumination originates outside the asset convex hull (Sec. 4.4).

We do not use spherical coordinates as they have zero Jacobian determinants at the poles which introduces singularities. Derivations of the projection factors above are described in the supplement.

Vector-valued normalizing flow. Given the Euclidean parameter s , we model the scattering distribution with an autoregressive normalizing flow $\mathbf{p}_\theta(s|\mathbf{x}_o, \omega_o) = \prod_{j=1}^4 \mathbf{p}_\theta(s^j | s^{<j}, \mathbf{x}_o, \omega_o)$. As demonstrated in Fig. 6, each conditional factor is realized by an MLP-predicted rational quadratic spline (rqs) [Durkan et al. 2019] of knots $\mathbf{g}_j = \text{MLP}_{j,\theta}(s^{<j}, \mathbf{x}_o, \omega_o)$, used for both pdf evaluation and sampling per color channel c :

$$\begin{aligned} \mathbf{p}_\theta^c(s^j | s^{<j}, \mathbf{x}_o, \omega_o) &= \frac{d \text{rqs}^{-1}(s^j, \mathbf{g}_j^c)}{ds^j} \quad (\text{pdf evaluation}) \\ s^j &= \text{rqs}(u, \mathbf{g}_j^c) \quad u \sim \text{Uniform}(0, 1) \quad (\text{pdf sampling}). \end{aligned} \quad (11)$$

To sample s , we draw s^1 then s^2 conditioned on s^1 and so on, and the full pdf evaluation is the product of the four derivatives. Unlike standard normalizing flows that output a single scalar density, our \mathbf{p}_θ is vector-valued over RGB, so each \mathbf{g}_j encodes three splines accordingly. The autoregressive structure further implies a factorization of the distribution $\mathbf{p}_\theta(u_i, \omega_i | \cdot) = \mathbf{p}_\theta(u_i | \cdot) \mathbf{p}_\theta(\omega_i | u_i, \cdot)$, which we later exploit for multiple importance sampling (MIS) in Sec. 3.3.

3.3 Training and inference

Network architecture. Each of the 4 MLPs in the distribution network uses ReLU activations, 4 hidden layers, and 128 hidden features. The j -th MLP takes $(\mathbf{x}_o, \omega_o, s^1, \dots, s^{j-1})$ as input and predicts the 2D knot positions and derivatives of a rational quadratic spline. We use 32 spline knots per RGB channel, resulting in $3 \times 32 \times (2+1) = 288$ output size. The albedo MLP takes (\mathbf{x}_o, ω_o) as input and outputs an RGB color. It also uses ReLU activations and 128 hidden features but with 2 hidden layers.

Input encoding. To capture high-frequency details, \mathbf{x}_o in all networks is encoded with a $3 \times 64 \times 64$ triplane feature grid [Chan et al. 2022], and ω_o is encoded with a $6 \times 32 \times 32$ cubemap feature grid [Wu et al. 2024]. In the distribution network, s^1, s^3 in the first and third MLPs are encoded with a 1D feature grid of resolution 32. For the third and fourth MLPs, the pair (s^1, s^2) is first mapped to a

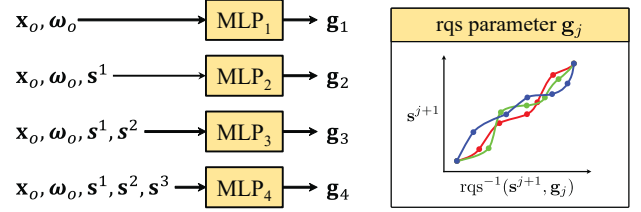


Fig. 6. **Vector-valued normalizing flow for \mathbf{p}_θ .** Left: the distribution of s is constructed autoregressively using rational quadratic splines (rqs). Right: the rqs parameter \mathbf{g}_j contains K knots for each RGB channel.

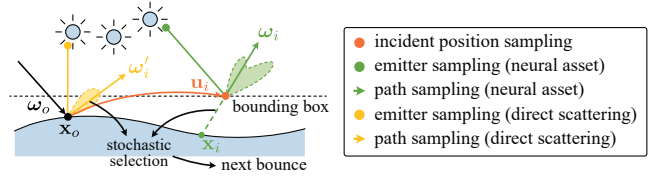


Fig. 7. **Path tracing our model** is similar to the standard scheme, except we perform MIS at the sampled incident location (orange dot). If direct scattering is separated, we further emitter-sample the direct lobe at x_o (yellow dot) and stochastically choose between direct samples (yellow arrow) and neural asset samples (green arrow) in path sampling.

unit direction then encoded with a $6 \times 32 \times 32$ cubemap grid. All the encodings use feature vectors of size 8 per grid vertex.

Training data generation. We optimize Eq. (8) using a buffer of path-sampled tuples $(\mathbf{x}_o, \omega_o, \mathbf{x}_i, \omega_i, \beta_i)$ that are regenerated once being fully consumed. Each tuple is obtained by first uniformly sampling a point on the asset’s bounding sphere then cosine-weighted sampling ω_o in the local tangent frame. We trace this ray to find the first intersection \mathbf{x}_o and continue path tracing to obtain $(\mathbf{x}_i, \omega_i, \beta_i)$. Tuples whose initial ray misses the asset are excluded from the training. Compared to precomputing the training data [Mullia et al. 2024], the online sampling prevents overfitting to a static set of paths. The buffer stores 128^4 such tuples in 15GB RAM.

Optimization. Our code is implemented using Pytorch [Paszke et al. 2019] and Mitsuba 3 [Jakob et al. 2022]. All networks are trained with the Adam optimizer [Kingma and Ba 2014] at a 0.0005 learning rate for 240K steps, using a batch size of 32768 samples per step. This leads to $(240K \times 32768) / 128^4 = 30$ buffer refreshes.

Integrating with path tracing. Path tracing with our model follows a standard emitter-path sampling scheme (Fig. 7): each path-tracing bounce begins by sampling an incident location $u_i \sim \mathbf{p}_\theta^c(u_i | \mathbf{x}_o, \omega_o)$ using the pdf factorization in Sec. 3.2 over a uniformly sampled color channel c ; MIS is then performed at u_i between the emitter-sampling technique and $\mathbf{p}_\theta^c(\omega_i | u_i, \mathbf{x}_o, \omega_o)$. When F_θ is trained on the indirect scattering, we also emitter-sample the direct lobe f at x_o and stochastically choose between a direct path sample and a neural asset sample as the next-bounce ray. Details of our path tracing is provided in the supplemental material. We use a custom Mitsuba integrator for path tracing and a CUDA kernel for the pdf sampling



Fig. 8. **Assets used in our experiments.** From left to right, top to bottom, the first 6 assets exhibit strong subsurface scattering from homogeneous (*Candle*, *Milk*) and heterogeneous media (the rest), all enclosed by glossy dielectric boundaries. *CurlHair*, *Hair*, and *Fabric* are modeled by a hair BSDF [Chiang et al. 2016], and *Teaset* uses a conductor BSDF.

Table 1. **Quantitative rendering error.** Our model achieves lower MSE (scaled by 100) on all assets than the far-field baseline.

Method	Candle	Milk	Cat	Seal	Dragon	Bunny	CurlHair	Hair	Fabric	Teaset
Far-field	5.760	0.430	0.162	0.065	0.505	0.103	0.228	1.381	3.379	0.707
Ours	2.855	0.156	0.009	0.057	0.182	0.057	0.126	0.950	2.541	0.075

and evaluation (Eq. (11)). We did not find a benefit of writing CUDA kernels for the MLPs given their relatively large sizes.

4 Experiments

For each asset in Fig. 8, we prebake its light transport to our model then evaluate its Monte Carlo rendering under novel scenes by comparing reconstruction accuracy and variance separately. The reconstruction accuracy is measured against the non-prebaked path tracing reference in MSE at high spp to suppress variance. The rendering variance is estimated by the MSE between the rendering and a high-spp rendering of the same method, and it is considered together with rendering speed to evaluate sampling efficiency. We also measure the time required to prebake each asset. All images are rendered in 800×800 resolution on an NVIDIA 3090 GPU.

4.1 Baselines.

We mainly compare with the standard path tracing (PT) and a far-field light transport baseline (Far-field) constructed following Tg et al. [2024a] and Mullia et al. [2024]. While [Tg et al. 2024b] is also a near-field method, it is restricted to isotropic subsurface scattering and does not generalize to all of our test scenes. We therefore include only a partial comparison to this method in the supplement.

Far-field. We model the 6D F' with an MLP of 128 hidden units and 8 hidden layers trained over the regression loss Eq. (3); and we learn the importance sampling of ω_i following Xu et al. [2023], using a 16-knot rqs normalizing flow parameterized by two MLPs (each with 64 hidden units and 2 hidden layers). The overall network capacity is chosen to be comparable to ours excluding $p_\theta(\mathbf{u}_i|\mathbf{x}_o, \omega_o)$ (as it is not presented in F'), to ensure the performance gain is from the representation rather than over-parameterization. All networks use the same input encodings and optimization settings as described

Table 2. **Rendering variance and speed at 128 spp.** Far-field model achieves the lowest variance and fastest inference speed but suffers from biased light transport. Our rendering achieves accuracy comparable to path tracing while reducing variance and rendering time: roughly 2-20 \times faster on volumetric assets and 1.4-4 \times faster on fine-scale surface scattering (*CurlHair*, *Hair*, *Fabric*). All methods render simple assets (*Teaset*) efficiently.

Method	Candle	Milk	Cat	Seal	Dragon	Bunny	CurlHair	Hair	Fabric	Teaset
	100\timesVariance\downarrow (128spp)									
Far-field	12.20	0.445	0.093	0.011	0.134	0.014	0.134	0.067	0.090	0.110
PT	54.34	33.41	0.115	2.561	5.723	3.673	0.655	2.219	0.414	0.102
Ours	27.44	2.491	0.094	0.084	1.975	0.588	0.572	2.023	0.573	0.107
	minutes\downarrow (128spp)									
Far-field	0.35	0.39	0.33	0.30	0.35	0.35	0.29	0.30	0.40	0.40
PT	4.13	15.6	1.28	2.04	4.27	4.97	0.80	2.12	4.49	0.82
Ours	0.53	0.69	0.57	0.55	0.52	0.57	0.57	0.58	0.93	0.65

in Sec. 3.3. We do not use the visibility hint [Müller et al. 2019] but apply our direct-scattering separation that serves the same purpose. To obtain ground-truth F' for supervision, we find 4096 spp are required for convergence on purely surface-scattering assets and 8192 spp for assets with volumetric interiors. Generating and resampling such data online would take several days [Tg et al. 2024a], so we precompute a static dataset of 1024×256^2 training samples using the strategy of Mullia et al.

Path tracing. We disable emitter sampling for media enclosed in dielectric boundaries, as the rays inside are never visible to light sources. For fair comparison, we also disable the Mitsuba megakernel backend when recording rendering speed that is not available for the other methods involving PyTorch calls.

4.2 Results

Accuracy of light transport. Figure 9 shows each asset rendered under area or environment light with background geometry contributing indirect illumination. A rendering of multiple assets composed into a single scene is provided in the supplement. Although this is a near-field setup, the diffuse background essentially smooths the illumination, so the far-field baseline still provides a reasonable approximation of the path-traced reference. On closer inspection, however, the 6D light transport that ignores \mathbf{x}_i overestimates incoming radiance especially near the shadowed region of *Candle*, *Seal*, and *CurlHair*; additional comparisons under purely far-field and purely near-field lighting in the supplement highlight when these artifacts are least and most visible. Due to the variance issue of the regression loss, the far-field model also fails to reproduce strong view-dependent effects, including the interreflections of the *Teaset* and the *Cat* in the mirror, where the cat’s interior white-red-blue volumetric slabs appear purple in the front view. Benefiting from the correct 8D light transport modeling and the low-variance distribution-learning objective, our model accurately handles these effects and achieves consistently lower MSE with respect to the reference renderings (Tab. 1).

Sampling efficiency. Table 2 reports the rendering variance and speed under 128 spp with the influence of rendering bias removed. Because the far-field model pre-integrates over \mathbf{x}_i and does not need

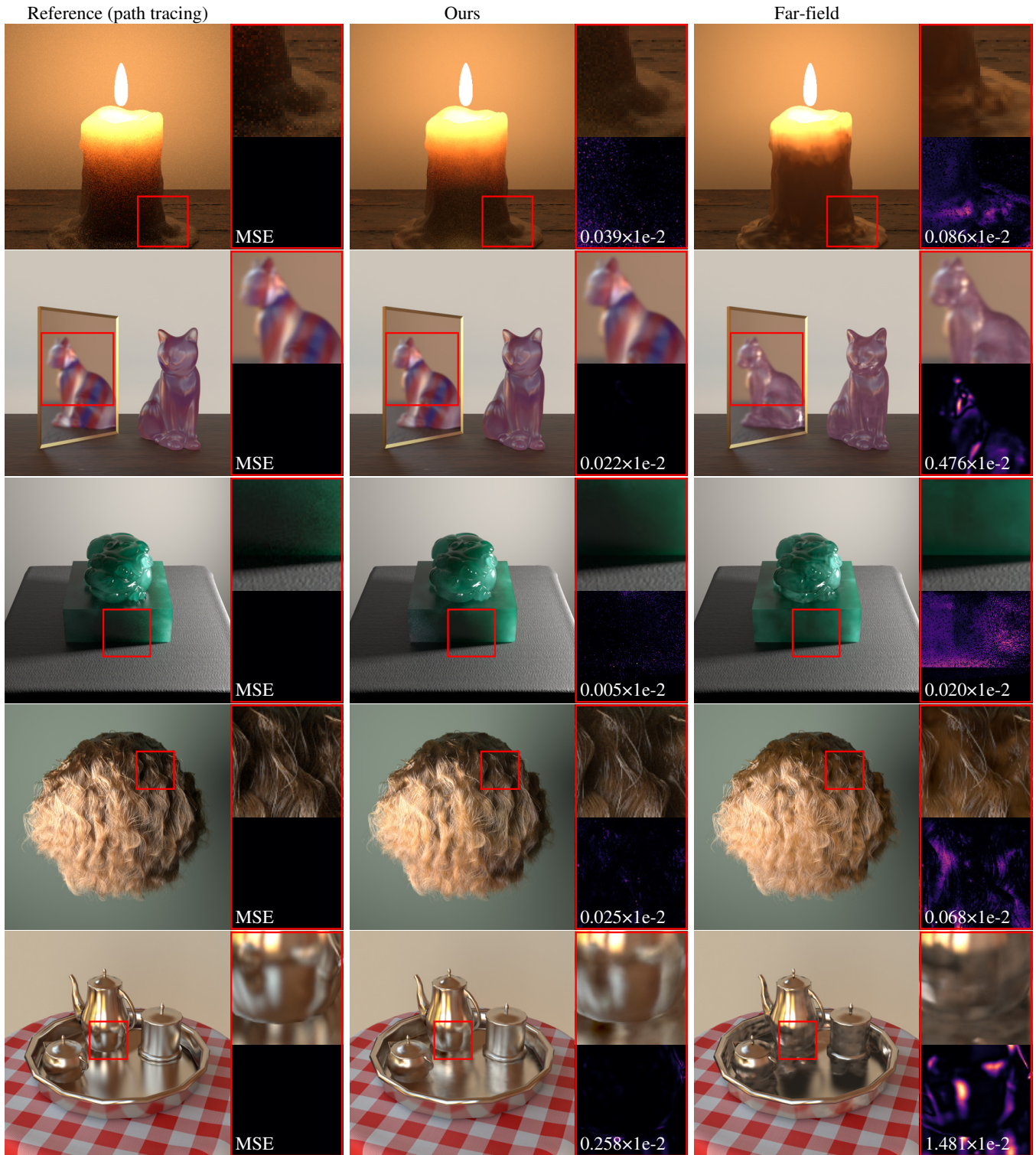


Fig. 9. **Qualitative rendering comparison.** The assets shown in Fig. 8 are baked into neural light transport models, and the background geometry uses standard material models. We use 8192 spp except for *Milk*, *Seal*, and *Dragon* where the path tracing uses 65536 spp. Our method more accurately captures the light transport of the assets under near-field illumination. The numbers show the MSEs of the insets.

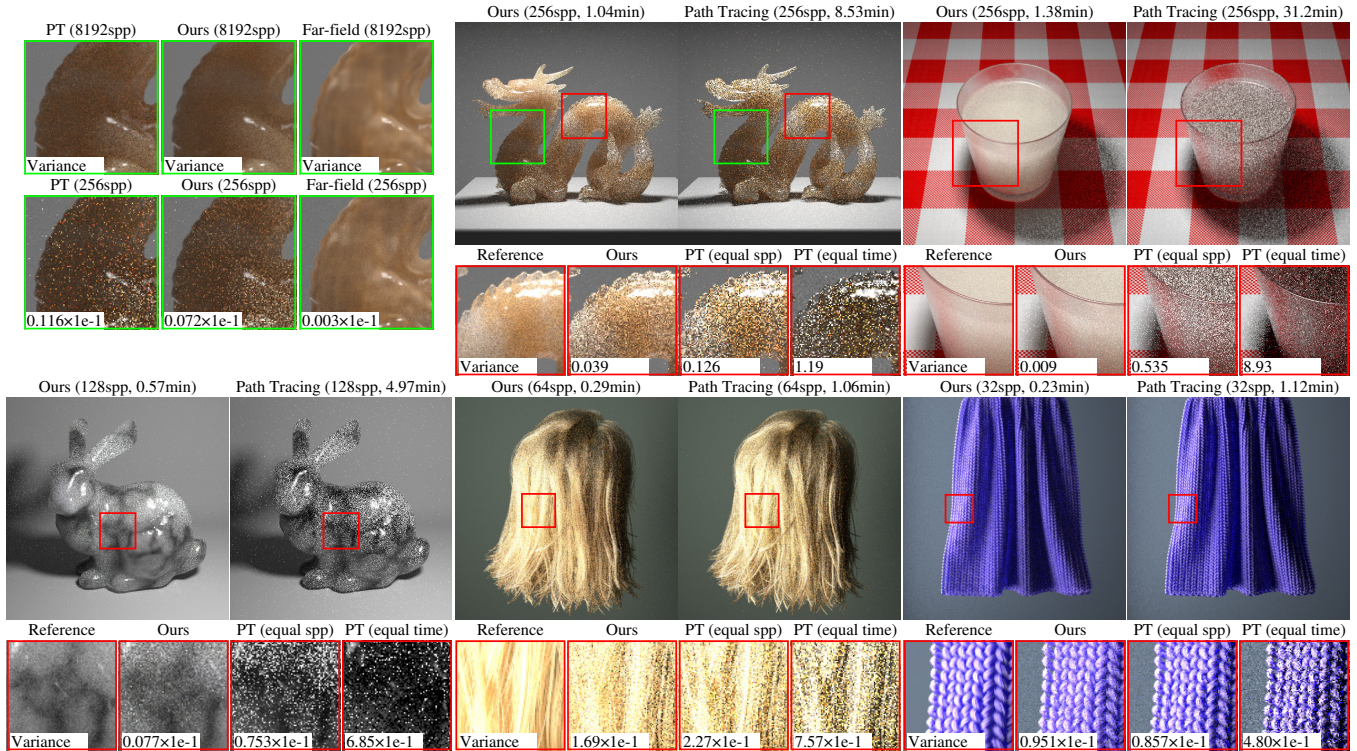


Fig. 10. **Qualitative rendering variance comparison.** Far-field demonstrates the least variance but its rendering does not match the path-traced reference (top left). Overall, our method achieves less variance than path tracing in both equal spp and equal time rendering with comparable rendering quality. The numbers show the variance of the insets.

to evaluate an x_i -dependent network, it exhibits much less variance than our method and is around twice faster in inference. However, this is at the cost of inaccurate light transport as discussed above. Our pre-baking noticeably reduces the variance on assets with volumetric interiors, for which naive path tracing cannot exploit emitter sampling to reduce variance (Fig. 10). On surface-only assets where the MIS is effective, we achieve similar variance or are slightly worse due to the lack of MIS over x_i (Sec. 4.4). But path-tracing over long scattering paths can be expensive especially for high-albedo assets (*Seal*, *Milk*, and *Hair*), so we still achieve less variance in equal time rendering. As shown in the convergence plot (Fig. 11), the advantage of our method grows with the asset’s complexity; we do not gain too much performance on simple assets like *Teaset* that are already efficient to render with path tracing.

Training time. As shown in Tab. 3, the far-field model is faster to optimize as it mainly evaluates MLPs. In contrast, pdf evaluations dominate our normalizing flow training, which does not have an efficient built-in implementation. However, generating each far-field training data sample requires thousands of spp, while our data is produced online with only 1 spp. Taking both factors into account, our full training is $3\times$ faster on volumetric assets and roughly comparable ($1.1\times$) on surface-only assets. Note our data are resampled for 30 times (Sec. 3.3), which would make the far-field training take days if using the same strategy.

Table 3. **Timing of each training stage.** Our networks are slower to optimize than the far-field model (including both light transport and the importance sampling module), while our data generation is substantially faster. The time is averaged over all volumetric and surface-only assets separately.

Asset type	Method	Time (hours)		
		Data generation	Optimization	Total
Volumetric	Far-field	5.91	0.84	6.75
	Ours	0.55	1.68	2.23
Surface	Far-field	1.26	0.72	1.98
	Ours	0.33	1.45	1.78

4.3 Ablation study

Network architecture. Table 4 shows the performance trade-off between different variants of our model. Reducing the network size lowers the rendering time but increases reconstruction error. The direct-scattering separation improves rendering quality without introducing a noticeable decrease in rendering speed. All the variants have similar rendering variance.

Bounding geometry proxy. Besides the bounding box, we can also take intersections on other geometry proxies as the parameterization of incident rays. As shown in Fig. 12, the rendering variance is reduced with tighter bounding geometry like convex hull but

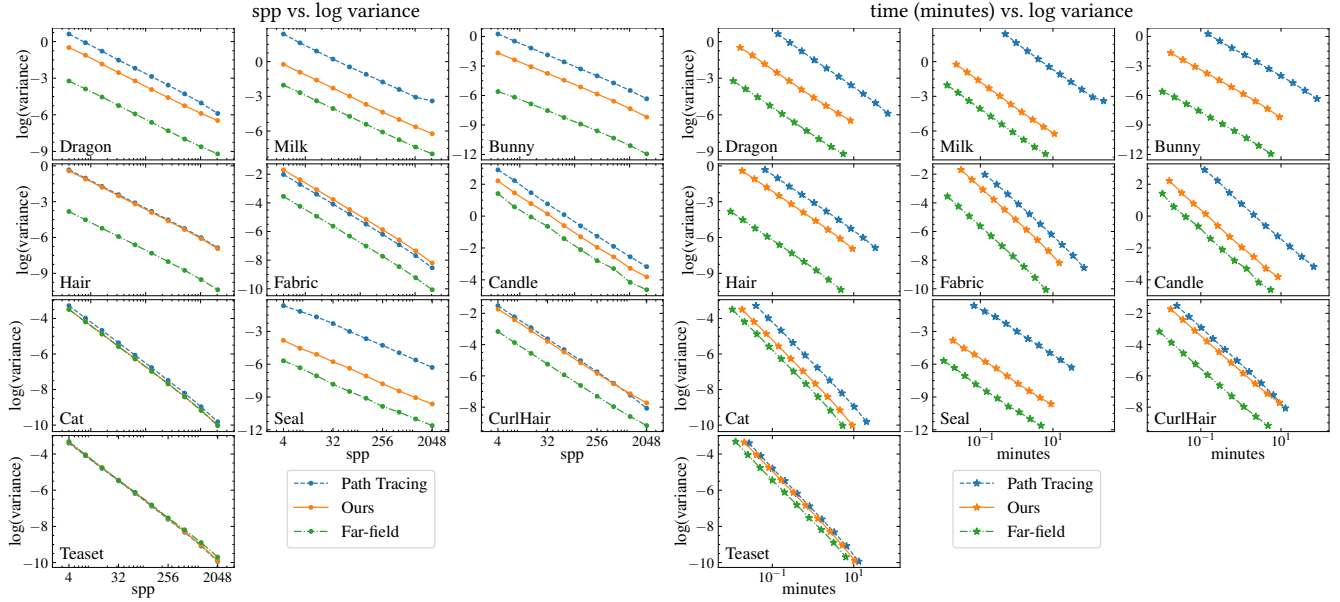


Fig. 11. **Convergence graphs.** Columns 1-3 show the log-log plot of the rendering variance with respect to the spp. Columns 4-6 show the log-log plot of the variance with respect to the rendering time. Our method achieves better variance reduction than path tracing in both comparisons, and the improvement is more significant on more complex scenes. The variance is computed with respect to a high spp rendering for each method separately *without* considering reconstruction accuracy (Tab. 1). Far-field does converge faster but is not consistent with the path-traced ground truth (see Tab. 1 and Fig. 9).

Table 4. **Ablation on network architectures.** ‘Smaller network’ decreases the MLP width to 64 in both p_θ and α_θ and uses 16-knot rqs in normalizing flows, which speeds up the rendering while introducing more reconstruction error. The inference speed is similar without the direct-scattering separation, yet the rendering quality is decreased.

Model variants	100×MSE↓	100×Variance↓	seconds↓
Ours	0.200	0.240	12.21
Smaller network	0.416	0.210	7.16
Without separation	0.365	0.227	12.17

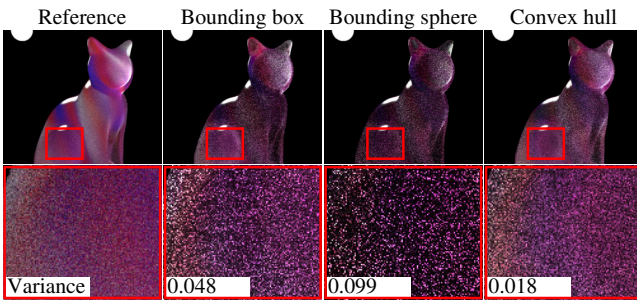


Fig. 12. **Bounding geometry for incident ray parameterization.** Bounding geometries with smaller surface area like convex hull tend to exhibit less variance. The renderings use 128 spp. Intersections on each geometry proxy above are mapped to cylindrical coordinates to fit into the normalizing flow.

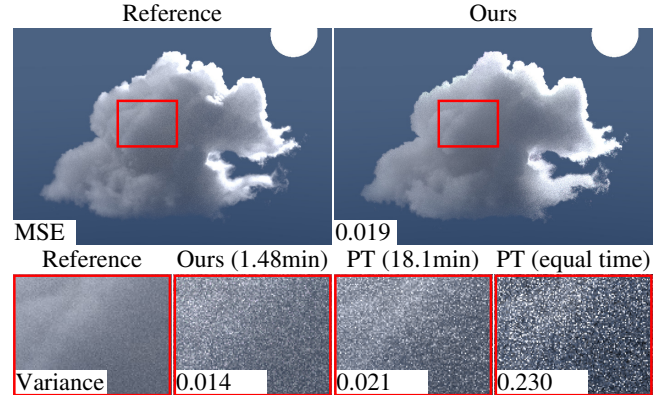


Fig. 13. **Our method extended to a purely volumetric asset.** The *Cloud* is illuminated by a sphere area light and an environment light. Our rendering closely matches the path-traced reference while it achieves similar equal-spp variance but faster inference (insets rendered at 128 spp).

increased with bounding sphere that has larger surface area for *Cat* than bounding box. This is because the variance of x_i -sampling correlates to the surface area of bounding geometry. Neural assets using bounding boxes already demonstrate low rendering variance in most of our scenes, so we choose it for the simplicity of its implementation.

Extension to purely volumetric asset. For assets without a solid boundary, we can define x_o by transmittance-sampling the location of the first scattering event. Our model then learns the remaining

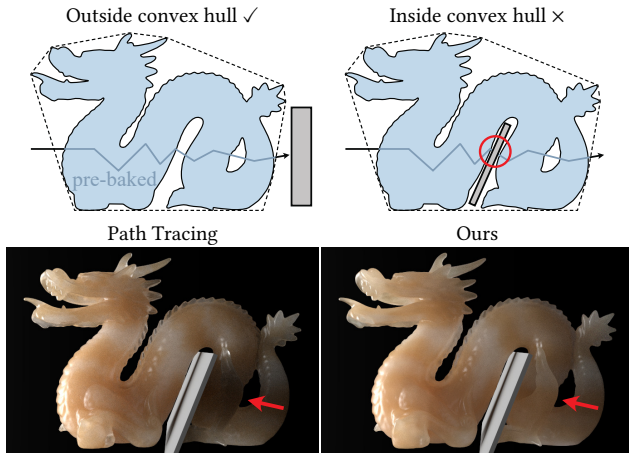


Fig. 14. **Failure case of our model.** An area emitter is placed on the right of *Dragon*. The diffuse slab blocks the pre-baked light paths (top right), causing our model to miss occlusion effects (red arrow). Our method assumes no additional geometry lies inside the asset’s convex hull (top left).

light transport inside the volume. Figure 13 shows the *Cloud* can be reconstructed using this approach. We do not observe significant variance reduction with our pre-baking, as path tracing can apply MIS to the pure volume. However, our model is faster to evaluate than explicit multi-scattering, which leads to improved equal-time variance compared to path tracing.

4.4 Limitations

We pre-bake light transport for each asset in isolation, so the internal scattering is essentially bounded by the asset’s convex hull. When inserting the asset into a new scene, our model remains accurate only if the internal scattering structure does not change. This means no other objects can intersect with the asset’s convex hull (Fig. 14). Applying convex decomposition to the asset then pre-baking each component separately can resolve this issue. Meanwhile, we currently do not use MIS over x_i , which may lead to high rendering variance under small emitters like point lights. Sampling incident locations visible to the emitters may help.

The usage of normalizing flows also introduces limitations. On the one hand, the normalizing flow evaluation is less efficient than MLP, so an MLP-based light transport model remains preferable if the far-field approximation is sufficiently accurate. On the other hand, normalizing flows have an inductive bias toward smooth distributions, making it difficult to reproduce high-frequency phenomena, such as specular interreflections between complex shapes or caustics and glints that lie in a sub-manifold of the incident domain.

5 Conclusion and Future Work

We have demonstrated an 8D neural representation that learns global light transport inside 3D assets from forward path-traced samples. This distribution-learning framework provides low-variance gradient estimates for optimizing our light transport model, while the 8D parameterization accurately pre-bakes internal scattering even under near-field illumination. Together, our method noticeably

reduces rendering variance and time compared to standard path tracing while preserving high visual fidelity. Looking forward, the model has the potential to deliver fast and accurate global illumination in real-time applications.

Acknowledgments

This work was supported in part by NSF grants 2212085, 2341952, 2105806, NSF Chase-CI grants 2100237, 2120019, and ONR grant N00014-23-1-2526. We also acknowledge an NVIDIA Fellowship, gifts from Adobe, Google and Qualcomm, the Ronald L. Graham Chair and the UC San Diego Center for Visual Computing. Ramamoorthi acknowledges a part-time appointment at NVIDIA.

References

- Aner Ben-Artzi, Kevin Egan, Frédo Durand, and Ravi Ramamoorthi. 2008. A precomputed polynomial representation for interactive BRDF editing with global illumination. In *ACM ToG*.
- Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. 2022. Efficient geometry-aware 3d generative adversarial networks. In *CVPR*.
- Matt Jen-Yuan Chiang, Benedikt Bitterli, Chuck Tappan, and Brent Burley. 2016. A practical and controllable hair and fur model for production path tracing. In *Computer Graphics Forum*. 275–283.
- Petrik Clarberg and Tomas Akenine-Möller. 2008. Practical product importance sampling for direct illumination. In *Computer Graphics Forum*.
- Xi Deng, Fujun Luan, Bruce Walter, Kavita Bala, and Steve Marschner. 2022. Reconstructing translucent objects using differentiable rendering. In *SIGGRAPH*.
- Laurent Dinh, David Krueger, and Yoshua Bengio. 2014. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516* (2014).
- Stavros Diolatzis, Adrien Gruson, Wenzel Jakob, Derek Nowrouzezahrai, and George Drettakis. 2020. Practical product path guiding using linearly transformed cosines. In *Computer Graphics Forum*.
- Ana Dodik, Marios Papas, Cengiz Öztireli, and Thomas Müller. 2022. Path Guiding Using Spatio-Directional Mixture Models. In *Computer Graphics Forum*.
- Craig Donner and Henrik Wann Jensen. 2006. A Spectral BSSRDF for Shading Human Skin. In *Rendering techniques*.
- Craig Donner, Jason Lawrence, Ravi Ramamoorthi, Toshiya Hachisuka, Henrik Wann Jensen, and Shree Nayar. 2009. An empirical BSSRDF model. In *ACM ToG*.
- Craig Donner, Tim Weyrich, Eugene d’Eon, Ravi Ramamoorthi, and Szymon Rusinkiewicz. 2008. A layered, heterogeneous reflectance model for acquiring and rendering human skin. In *ACM ToG*.
- Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. 2019. Neural spline flows. In *NeurIPS*.
- Jiahui Fan, Beibei Wang, Milos Hasan, Jian Yang, and Ling-Qi Yan. 2022. Neural layered brdfs. In *SIGGRAPH*.
- Ziyang Fu, Yash Belhe, Haolin Lu, Liwen Wu, Bing Xu, and Tzu-Mao Li. 2024. BSDF importance sampling using a diffusion model. In *SIGGRAPH Asia*.
- Ralf Habel, Per H Christensen, and Wojciech Jarosz. 2013. Photon beam diffusion: A hybrid monte carlo method for subsurface scattering. In *Computer Graphics Forum*.
- David Hart, Matt Pharr, Thomas Müller, Ward Lopes, Morgan McGuire, and Peter Shirley. 2020. Practical product sampling by fitting and composing warps. In *Computer Graphics Forum*.
- Eric Heitz, Laurent Belcour, and Thomas Chambon. 2023. Iterative α -(de) blending: A minimalist deterministic diffusion model. In *SIGGRAPH*.
- Sebastian Herholz, Oskar Elek, Jiří Vorba, Hendrik Lensch, and Jaroslav Krivánek. 2016. Product importance sampling for light transport path guiding. In *Computer Graphics Forum*.
- Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, and Delio Vicini. 2022. Dr. jit: A just-in-time compiler for differentiable rendering. In *ACM ToG*.
- Henrik Wann Jensen, Stephen R Marschner, Marc Levoy, and Pat Hanrahan. 2001. A practical model for subsurface light transport. In *ACM ToG*.
- Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
- Alexandr Kuznetsov, Krishna Mullia, Zexiang Xu, Miloš Hašan, and Ravi Ramamoorthi. 2021. Neumip: Multi-resolution neural materials. In *ACM ToG*.
- Alexandr Kuznetsov, Xuezheng Wang, Krishna Mullia, Fujun Luan, Zexiang Xu, Milos Hasan, and Ravi Ramamoorthi. 2022. Rendering neural materials on curved surfaces. In *SIGGRAPH*.
- Zixuan Li, Zixiong Wang, Jian Yang, Miloš Hašan, and Beibei Wang. 2025. Pure-Sample: Neural Materials Learned by Sampling Microgeometry. *arXiv preprint arXiv:2508.07240* (2025).

- Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. 2022. Flow Matching for Generative Modeling. In *ICLR*.
- Joey Litalien, Miloš Hašan, Fujun Luan, Krishna Mullia, and Iliyan Georgiev. 2024. Neural Product Importance Sampling via Warp Composition. In *SIGGRAPH Asia*.
- Haolin Lu, Yash Belhe, Gurprit Singh, Tzu-Mao Li, and Toshiya Hachisuka. 2025. Gaussian Integral Linear Operators for Precomputed Graphics. In *ACM ToG*.
- Thomas Müller, Markus Gross, and Jan Novák. 2017. Practical path guiding for efficient light-transport simulation. In *Computer Graphics Forum*.
- Thomas Müller, Brian McWilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. 2019. Neural importance sampling. In *ACM TOG*.
- Krishna Mullia, Fujun Luan, Xin Sun, and Miloš Hašan. 2024. RNA: Relightable Neural Assets. In *ACM ToG*.
- Ren Ng, Ravi Ramamoorthi, and Pat Hanrahan. 2003. All-frequency shadows using non-linear wavelet lighting approximation. In *ACM ToG*.
- Ren Ng, Ravi Ramamoorthi, and Pat Hanrahan. 2004. Triple product wavelet integrals for all-frequency relighting. In *ACM ToG*.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*.
- Pieter Peers, Karl Vom Berge, Wojciech Matusik, Ravi Ramamoorthi, Jason Lawrence, Szymon Rusinkiewicz, and Philip Dutré. 2006. A compact factored representation of heterogeneous subsurface scattering. In *ACM ToG*.
- Gilles Rainer, Adrien Bousseau, Tobias Ritschel, and George Drettakis. 2022. Neural precomputed radiance transfer. In *Computer graphics forum*.
- Gilles Rainer, Abhijeet Ghosh, Wenzel Jakob, and Tim Weyrich. 2020. Unified neural encoding of BTFs. In *Computer Graphics Forum*.
- Ravi Ramamoorthi et al. 2009. Precomputation-based rendering. *Foundations and Trends® in Computer Graphics and Vision* 3, 4 (2009), 281–369.
- Ravi Ramamoorthi and Pat Hanrahan. 2001. An efficient representation for irradiance environment maps. In *ACM ToG*.
- Danilo Rezende and Shakir Mohamed. 2015. Variational inference with normalizing flows. In *ICML*.
- Peter-Pike Sloan, Jan Kautz, and John Snyder. 2002. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *ACM ToG*.
- Alejandro Sztrajman, Gilles Rainer, Tobias Ritschel, and Tim Weyrich. 2021. Neural brdf representation and importance sampling. In *Computer Graphics Forum*.
- Thomson Tg, Jeppe Revall Frisvad, Ravi Ramamoorthi, and Henrik W Jensen. 2024a. NeuPreSS: Compact Neural Precomputed Subsurface Scattering for Distant Lighting of Heterogeneous Translucent Objects. In *Computer Graphics Forum*.
- Thomson Tg, Duc Minh Tran, Henrik W Jensen, Ravi Ramamoorthi, and Jeppe Revall Frisvad. 2024b. Neural SSS: Lightweight object appearance representation. In *Computer Graphics Forum*.
- Yu-Ting Tsai and Zen-Chung Shih. 2006. All-frequency precomputed radiance transfer using spherical radial basis functions and clustered tensor approximation. In *ACM ToG*.
- Delio Vicini, Vladlen Koltun, and Wenzel Jakob. 2019. A learned shape-adaptive subsurface scattering model. In *ACM ToG*.
- Jiří Vorba, Ondřej Karlík, Martin Šik, Tobias Ritschel, and Jaroslav Krivánek. 2014. On-line learning of parametric mixture models for light transport simulation. In *ACM ToG*.
- Jingwen Wang and Ravi Ramamoorthi. 2018. Analytic spherical harmonic coefficients for polygonal area lights. In *ACM ToG*.
- Liwen Wu, Sai Bi, Zexiang Xu, Fujun Luan, Kai Zhang, Iliyan Georgiev, Kalyan Sunkavalli, and Ravi Ramamoorthi. 2024. Neural directional encoding for efficient and accurate view-dependent appearance modeling. In *CVPR*.
- Liwen Wu, Sai Bi, Zexiang Xu, Hao Tan, Kai Zhang, Fujun Luan, Haolin Lu, and Ravi Ramamoorthi. 2025. Neural BRDF Importance Sampling by Reparameterization. In *SIGGRAPH*.
- Lifan Wu, Guangyan Cai, Shuang Zhao, and Ravi Ramamoorthi. 2020. Analytic spherical harmonic gradients for real-time rendering with many polygonal area lights. In *ACM ToG*.
- Feng Xie, Anton Kaplanyan, Warren Hunt, and Pat Hanrahan. 2019. Multiple scattering using machine learning. In *SIGGRAPH*.
- Bing Xu, Liwen Wu, Milos Hasan, Fujun Luan, Iliyan Georgiev, Zexiang Xu, and Ravi Ramamoorthi. 2023. NeuSample: Importance sampling for neural materials. In *SIGGRAPH*.
- Kun Xu, Wei-Lun Sun, Zhao Dong, Dan-Yong Zhao, Run-Dong Wu, and Shi-Min Hu. 2013. Anisotropic spherical gaussians. In *ACM ToG*.
- Zilin Xu, Zheng Zeng, Lifan Wu, Lu Wang, and Ling-Qi Yan. 2022. Lightweight neural basis functions for all-frequency shading. In *SIGGRAPH Asia*.
- Ling-Qi Yan, Weilun Sun, Henrik Wann Jensen, and Ravi Ramamoorthi. 2017. A BSSRDF model for efficient rendering of fur with global illumination. In *ACM ToG*.
- Tizian Zeltner, Fabrice Rousselle, Andrea Weidlich, Petrik Clarberg, Jan Novák, Benedikt Bitterli, Alex Evans, Tomáš Davidovič, Simon Kallweit, and Aaron Lefohn. 2024. Real-time neural appearance models. In *ACM ToG*.