Neural Parametric Mixtures for Path Guiding

Honghao Dong
Peking University
Beijing, China
cuteday@pku.edu.cn

Guoping Wang
Peking University
Beijing, China
wgp@pku.edu.cn

Sheng Li*
Peking University
Beijing, China
lisheng@pku.edu.cn

ABSTRACT

Previous path guiding techniques typically rely on spatial subdivision structures to approximate directional target distributions, which may cause failure to capture spatio-directional correlations and introduce parallax issue. In this paper, we present Neural Parametric Mixtures (NPM), a neural formulation to encode target distributions for path guiding algorithms. We propose to use a continuous and compact neural implicit representation for encoding parametric models while decoding them via lightweight neural networks. We then derive a gradient-based optimization strategy to directly train the parameters of NPM with noisy Monte Carlo radiance estimates. Our approach efficiently models the target distribution (incident radiance or the product integrand) for path guiding, and outperforms previous guiding methods by capturing the spatio-directional correlations more accurately. Moreover, our approach is more training efficient and is practical for parallelization on modern GPUs.

CCS CONCEPTS

• **Computing methodologies** → **Ray tracing**; **Neural networks**.

KEYWORDS

Ray Tracing, Global Illumination, Sampling and Reconstruction, Neural Networks, Mixture Models

ACM Reference Format:
Honghao Dong, Guoping Wang, and Sheng Li. 2023. Neural Parametric Mixtures for Path Guiding. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Proceedings (SIGGRAPH '23 Conference Proceedings), August 06–10, 2023, Los Angeles, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3588432.3591533

1 INTRODUCTION

The efficiency of path tracing relies heavily on the sampling strategy. To further improve its efficiency and robustness, path guiding algorithms leverage the knowledge gained during rendering to facilitate the process of light-path construction, thereby reducing noise. To acquire better importance sampling distribution, local path guiding techniques employ previous radiance estimates to learn an approximation of spatial incident radiance fields, which are then used to guide the construction of paths. In practice, current methods typically use some representation (e.g., Gaussian mixtures [Herholz et al. 2016; Vorba et al. 2014], quadtrees [Müller et al. 2017]) to approximate the directional distribution of incident radiance. A spatial subdivision structure (e.g., kd-tree [Dodik et al. 2022], or octree [Bus and Boubekeur 2017]) is then used to store these distributions, thus accounting for the spatial variations.

However, several key deficiencies remain in their paradigm. Most methods learn the marginalized incident radiance distribution within each subdivided spatial region. This fails to capture the spatio-directional correlations within the spatial discretizations, and could cause artifacts (e.g., parallax error, Fig 1(a)). Moreover, their spatial subdivision structures are subject to frequent reconstruction for finer-grained spatial resolution, which needs extra overhead and require a long training time to converge. Meanwhile, it is challenging to efficiently fit these specific directional distributions from noisy samples, especially in an online manner [Ruppert et al. 2020].

While an adaptive and robust spatial representation is difficult to achieve with manually designed subdivision schemes, we saw the recent success of neural implicit representation in compactly modeling spatially varying functions with fine-grained and high-frequency details [Mildenhall et al. 2020]. In this work, we exploit the great expressiveness of neural implicit representation while preserving the desirable properties of parametric mixture models (e.g. efficient importance sampling) for path guiding algorithms. We thereby present Neural Parametric Mixtures (NPM), which use a continuous and compact implicit representation to encode spatio-directional target distributions, and decode them into PMMs with lightweight neural networks for fast importance sampling. We show that our NPM representation, without explicit spatial subdivision schemes, can be efficiently trained simply using gradient-based optimization techniques. Specifically, our method has advantages in the following aspects:

First, our continuous implicit representation of spatial radiance fields naturally captures the correlations between spatial positions and directional target distributions. By smoothly interpolating and decoding the implicit representations with neural networks, our method inherently avoids the issues due to spatial discretization, thus resulting in higher performance.

Second, our compact representation avoids the extra overhead and long training time caused by the iterative reconstruction strategies applied to the explicit spatial subdivision structures. Combined with our simple optimization based on stochastic gradient descent, our method outperforms other guiding methods even with fewer training samples. In addition, our method is practical and performant for parallelization on GPU.

Lastly, our method can learn the product distribution (i.e., multiplied by the BSDF and the cosine term). This further reduces the

*Corresponding author.
Project URL: https://neuropara.github.io.

SIGGRAPH '23 Conference Proceedings, August 06–10, 2023, Los Angeles, CA, USA

ACM ISBN 979-8-4007-0159-7/23/08…$15.00
https://doi.org/10.1145/3588432.3591533

noise with a modest computational overhead while not requiring the extra effort of previous solutions (e.g., fitting each BSDF with pre-computed parametric models).

2 RELATED WORK

Path Guiding. To achieve better sampling strategies, local path guiding techniques leverage previous radiance estimates (either online or during a pre-computation process) to build an approximation of the incident radiance fields, which is used to guide subsequent sampling. Early approaches used simple bases such as histograms for importance sampling, e.g. built from a photon map [Jensen 1995] or collected radiance estimates with 5-D tree structures [Lafortune and Willems 1995]. Subsequent work has developed various techniques to construct the guiding distribution, e.g., Gaussian mixtures [Vorba et al. 2014], quad-trees [Müller et al. 2017], which is often stored in spatial data structures (e.g., kd-tree and octree) to account for spatial variations of the distributions.

Deep learning techniques have also been explored recently, achieving improvements while often with less practical performance. For example, convolutional networks could be used to reconstruct the learned noisy radiance field [Huo et al. 2020; Zhu et al. 2021]. Specially designed neural networks could also model complex manifolds [Dinh et al. 2017], while allowing samples to be drawn directly from the learned distribution [Müller et al. 2019]. However, the prohibitive computational cost prevents its practical application [Müller et al. 2019; Vorba et al. 2019]. Instead of directly importance sampling using neural networks, we encode the target distribution into implicit neural representation, and use only lightweight MLPs to decode it into parametric mixtures for efficient sampling. We show that our method can be efficiently trained (< 10s per scene on a single GPU) while being sufficiently robust and practical.

Parametric Mixture Models. Parametric mixture models (PMMs) are convex combinations of parametric distributions, and are often used to approximate directional distributions in graphics applications. They have many desirable properties, e.g., fast sampling, and closed-form solutions for products, convolutions and integrals. Several types of PMMs (e.g., Gaussian mixtures [Dodik et al. 2022; Vorba et al. 2014] and von Mises-Fisher mixtures [Ruppert et al. 2020]) are widely used in the recently developed path guiding algorithms. Several recent works also use PMMs to fit BSDFs with precomputation [Herholz et al. 2016; Ruppert et al. 2020], and multiply them with the learned incident radiance to achieve product sampling.

Parametric models can also be predicted by neural networks, enabling new possibilities for e.g. lighting [Currius et al. 2020] and reconstruction [Yu et al. 2021] tasks. In this work, we use neural representations to encode parametric mixtures for efficient sampling. Our method is also naturally extensible to product sampling.

Implicit Neural Representation. Following the success of using neural networks to represent 3D scenes implicitly [Mildenhall et al. 2020], the concept of neural representation has been popularized and applied to various tasks. They use sparse input images to optimize the spatial radiance fields via a differentiable volume rendering procedure, thus enabling novel view synthesis. Inspired by its recent successful applications [Diolatzis et al. 2022; Müller et al. 2022], we exploit a continuous and compact implicit neural representation to encode the spatio-directional target distributions for path guiding algorithms. While the ground truth target distribution (i.e., the incident radiance or product distribution) is unknown, our NPM representation can be optimized online using minibatch stochastic gradient descent (SGD), where the gradients for training are estimated by Monte Carlo integration using noisy radiance estimates.

3 PRELIMINARY

Monte Carlo Integration. Light transport algorithms are generally based on the rendering equation [Kajiya 1986]:

$$L_{\mathrm{o}}\left(\mathbf{x}, \omega_{\mathrm{o}}\right)=L_{\mathrm{e}}\left(\mathbf{x}, \omega_{\mathrm{o}}\right)+\int_{\Omega} f_{\mathrm{s}}\left(\mathbf{x}, \omega_{\mathrm{o}}, \omega_{\mathrm{i}}\right) L_{\mathrm{i}}\left(\mathbf{x}, \omega_{\mathrm{i}}\right)\left|\cos \theta_{\mathrm{i}}\right| \mathrm{d} \omega_{\mathrm{i}}, \quad (1)$$

which defines the relationship between the outgoing radiance L_{o}, emitted radiance L_e, and the integrated incident radiance L_{i}, at shading point $\mathbf{x}$. Monte Carlo integration is used to obtain an estimate of the reflection integral L_r using an average of N samples. In the case where $N = 1$:

$$\left\langle L_{\mathrm{r}}\left(\mathbf{x}, \omega_{\mathrm{o}}\right)\right\rangle=\frac{f_{\mathrm{s}}\left(\mathbf{x}, \omega_{\mathrm{o}}, \omega_{\mathrm{i}}\right) L_{\mathrm{i}}\left(\mathbf{x}, \omega_{\mathrm{i}}\right)\left|\cos \theta_{\mathrm{i}}\right|}{p\left(\omega_{\mathrm{i}} \mid \mathbf{x}, \omega_{\mathrm{o}}\right)}, \quad (2)$$

where $\langle L_{\mathrm{r}}\left(\mathbf{x}, \omega_{\mathrm{o}}\right)\rangle$ is an unbiased estimate of the outgoing radiance $L_{\mathrm{r}}\left(\mathbf{x}, \omega_{\mathrm{o}}\right)$, and ω_i is the incident direction sampled with some directional probability distribution $p(\omega_{\mathrm{i}} \mid \mathbf{x}, \omega_{\mathrm{o}})$. The variance of this estimator $V[\langle L_{\mathrm{r}}\rangle]$ can be reduced if the sampling distribution resembles the shape of the integrand, and could even reach zero variance if being proportional to it (i.e., $p \propto f_s \cdot L_{\mathrm{i}} \cos\theta_i$). This, however, is difficult to achieve with only BSDF importance sampling, leaving the remaining part of the integrand (i.e., the incident radiance) unknown, resulting in a relatively high variance of the MC estimator. Path guiding algorithms, on the other hand, manage to obtain better importance sampling strategies often by using previous radiance samples to approximate the incident radiance L_{i} or the full integrand $f_s \cdot L_{\mathrm{i}} \cos\theta_i$, which will be discussed later.

Von Mises-Fisher Mixtures. We use the *von Mises-Fisher* (vMF) distribution as the basis of NPM. The vMF distribution is defined as:

$$v(\omega \mid \mu, \kappa)=\frac{\kappa}{4 \pi \sinh \kappa} \exp\left(\kappa \mu^{T} \omega\right), \quad (3)$$

where $\mu \in \mathbb{S}^2$ and $\kappa \in [0, +\infty)$ defines the direction and precision (sharpness) of the vMF distribution. The vMF mixture model (VMM) is thus a convex combination of K vMF components/lobes:

$$\mathcal{V}(\omega \mid \Theta)=\sum_{i=1}^{K} \lambda_i \cdot v\left(\omega \mid \mu_i, \kappa_i\right), \quad (4)$$

where Θ contains the parameters (μ_i, κ_i) and weights (λ_i) of each vMF component. The vMF mixtures have many desirable properties, e.g., fewer parameters (4 floats per component), efficient importance sampling, and closed-form product and integration, which together constitute the reason for choosing it as the basis of NPM.

Our key is to encode the vMF mixtures with our implicit neural representation, then decode them with lightweight MLPs, and train them to effectively model the target distributions for path guiding algorithms. Other parametric basis functions (e.g., Gaussian mixtures) could be integrated into our method using a similar paradigm.

4 NEURAL PARAMETRIC MIXTURES

In this section, we present our Neural Parametric Mixtures (NPM) technique for local path guiding. We first show how to encode/decode target distributions with NPM in a simple setup (i.e., learning incident radiance fields, Sec. 4.1), then we derive the optimization method for NPM based on minibatch stochastic gradient descent (Sec. 4.2). Finally, we show how our NPM could naturally benefit from learning the full integrand (to account for the BSDF term), as well as the other extensions for better learning target distributions (Sec. 4.3). An overview of our method is illustrated in Fig. 2.

4.1 Radiance-based NPM

In order to acquire a better importance sampling strategy, we should obtain an approximation of the incident radiance distribution using previous radiance estimates, known as the radiance-based local path guiding [Herholz et al. 2016; Rath et al. 2020]. Specifically, we want to use the vMF mixtures to be approximately proportional to the incident radiance, at a given shading position $\mathbf{x}$:

$$\mathcal{V}\left(\omega_i \mid \Theta\left(\mathbf{x}\right)\right) \propto L_{\mathrm{i}}(\mathbf{x}, \omega_i), \tag{5}$$

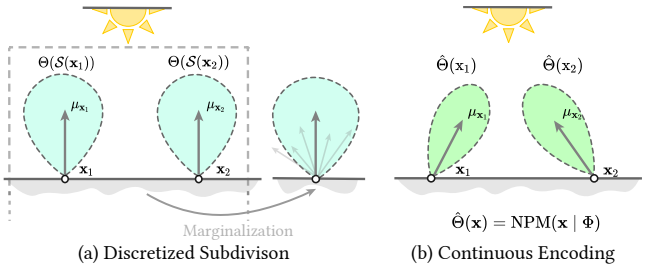
where Θ is conditioned on $\mathbf{x}$ to account for the spatial variation of the target distribution. Previous work achieves this with specific spatial subdivision strategies (e.g., kd-tree, octree). However, this spatial discretization introduces artifacts (e.g., resulting from parallax, Fig. 1 (a)), and is subject to frequent reconstruction to converge to a fine grained spatial subdivision, as discussed in Sec. 1.

Instead, we use an implicit neural representation to encode the target distribution compactly. This allows the spatial variation of the distribution to be continuously accounted for, thus better capturing spatio-directional correlations. Technically, given a shading position $\mathbf{x}$ in the scene, our NPM would output the guiding distribution that approximates the target distribution (Eq. 5). The output guiding distribution is defined using a set of parameters $\hat{\Theta}(\mathbf{x})$:

$$\mathbf{NPM}(\mathbf{x} \mid \Phi) = \hat{\Theta}(\mathbf{x}), \tag{6}$$

where Φ are the trainable parameters of the implicit representation, and $\hat{\Theta}$ are the output decoded parameters, defining a vMF mixture $\mathcal{V}(\omega_i \mid \hat{\Theta}(\mathbf{x}))$ that is trained to approximate $L_{\mathrm{i}}(\mathbf{x}, \omega_i)$ (Eq. 5). By continuously conditioning the learned distribution Θ on spatial positions $\mathbf{x}$, our method inherently avoids the above issues caused by spatial discretizations. We achieve the above mapping by using a lightweight network to decode this parametric distribution from the implicit neural representation. To make sure that we get a valid vMF mixture (i.e., $\lambda_i, \kappa_i > 0$, $\mu_i \in \mathbb{S}^2$, and $\sum_{j=1}^{K} \lambda_j = 1$), we must additionally regularize the raw network output with appropriate mapping functions (see Tab. 1). Specifically, we apply exponential activation to λ_i and κ_i. Logistic activation is applied to θ_i and φ_i, which form the spherical coordinates of μ_i. Most importantly, we apply the softmax function to all λs to ensure that the outputs model a valid PDF (i.e., satisfy $\sum_{i=1}^{K} \lambda_i = 1$).

Discussion. It is possible to implement different forms of implicit neural representation with trainable parameters Φ. While it is straightforward to use a monolithic network to model $\mathbf{NPM}_\Phi : \mathbf{x} \rightarrow \Theta$, we find it difficult to fit the high-frequency variations of the target distribution. Thereby, we use a trainable multi-resolution spatial embedding for encoding the distributions, and additionally a lightweight neural network for decoding the parameters. This is crucial for our method to achieve better modeling capacity while remaining performant, as will be discussed later.



(a) Discretized Subdivison (b) Continuous Encoding

Figure 1: Parallax issue caused by spatial discretizations (a). For a subdivided volume $\mathcal{S}(\mathbf{x})$ in (a), the guiding distribution is marginalized with training samples scattered over the volume $\mathcal{S}(\mathbf{x})$, and is shared by different positions (e.g., $\mathbf{x}_1$ and $\mathbf{x}_2$). Our method will not suffer from parallax due to NPM implicitly representing a monolithic function, continuously mapping from spatial positions to parametric guiding distributions, as shown in (b).

4.2 Optimizing NPM

We show how to optimize the divergence between the decoded distribution $\hat{\Theta}(\mathbf{x})$ and the target distribution using minibatch stochastic gradient descent. To achieve this, the gradients of a training objective (or loss function) with respect to the network parameters are necessary. However, it is non-trivial to define such a loss function, given the ground truth output parameters $\Theta_{\mathrm{gt}}(\mathbf{x})$ are unknown. Previous works typically use design optimization algorithms (e.g., expectation-maximization) that iteratively use batches of samples to fit a given set of parameters Θ, which often parameterize a marginalized distribution shared by the spatial region covering the samples [Herholz et al. 2016; Ruppert et al. 2020]. However, their methods are applied to explicitly parameterized models, and are therefore not applicable to our method, which models the implicit representation of the function $\mathbf{NPM}_\Phi : \mathbf{x} \rightarrow \hat{\Theta}$.

We minimize the KL divergence between the decoded vMF mixtures and the target distribution via minibatch stochastic gradient descent, where its gradients with respect to the trainable parameters are estimated using Monte Carlo integration. Other divergence metrics are also available following a similar derivation. Let us start

Table 1: Detailed mapping functions we use to regularize network outputs, where λ', κ' θ', φ' denote the raw outputs, and (θ, φ) is the normalized spherical coordinate of $\mu \in \mathbb{S}^2$. Left: parameter notations and their valid ranges; middle: type of activation; right: specific mappings.

Parameter	Activation	Mapping
$\kappa \in [0, +\infty)$	Exponential	$\kappa_i = \exp(\kappa'_i)$
$\lambda \in [0, +\infty)$	Softmax	$\lambda_i = \exp(\lambda'_i) / \sum_{j=1}^{K} \exp(\lambda'_j)$
$\theta,\ \varphi \in [0, 1]$	Logistic	$\theta_i = 1/(1 + \exp(-\theta'_i))$

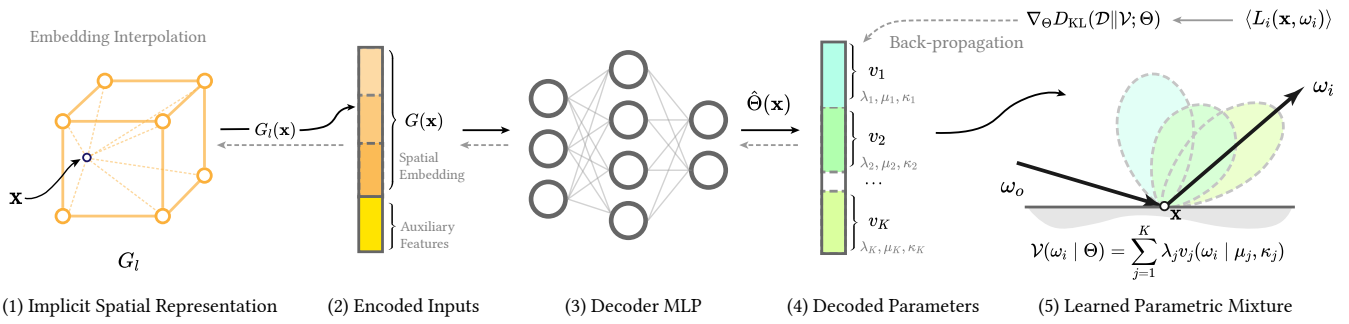


Figure 2: High-level illustration of our Neural Parametric Mixtures (NPM). We implicitly encode the spatially varying target distributions with the multi-resolution embedding. When the distribution of a spatial location $\mathbf{x}$ is queried, (1) the features assigned to the nearby grid points surrounding $\mathbf{x}$ are interpolated at each level, and concatenated with other levels to obtain the spatial embedding $G(\mathbf{x})$. (2) the spatial embedding is then combined with other inputs to (3) feed into the lightweight MLP for (4) decoding the parameters Θ of the vMF mixture $\mathcal{V}(\omega_i \mid \Theta)$ with K components. We then (5) use this parametric distribution for importance sampling the scattering direction. The result MC radiance estimate $\langle L_i(\mathbf{x},\omega_i)\rangle$ is used to estimate the training gradient $\nabla_\Theta D_{\mathrm{KL}}$ (Sec. 4.2), which is then back-propagated through these differentiable stages to optimize our NPM representation (dashed lines).

by assuming that the shading position $\mathbf{x}$ is fixed, thus omitting the dependency of Θ on $\mathbf{x}$ in the equations. For a given position, the KL divergence between the target distribution $\mathcal{D}$ and our output distribution $\mathcal{V}$ is defined as:

$$D_{\mathrm{KL}}(\mathcal{D}\|\mathcal{V};\Theta) = \int_\Omega \mathcal{D}(\omega)\log\frac{\mathcal{D}(\omega)}{\mathcal{V}(\omega \mid \hat{\Theta})}\,\mathrm{d}\omega, \tag{7}$$

where $\mathcal{D} \propto L_\mathrm{i}$ in radiance-based path guiding. This integral could now be estimated with the Monte Carlo estimator with N samples:

$$D_{\mathrm{KL}}(\mathcal{D}\|\mathcal{V};\Theta) \approx \frac{1}{N}\sum_{j=1}^{N}\frac{\mathcal{D}(\omega_j)}{\tilde{p}(\omega_j \mid \hat{\Theta})}\log\frac{\mathcal{D}(\omega_j)}{\mathcal{V}(\omega_j \mid \hat{\Theta})}, \tag{8}$$

where $\tilde{p}$ is the distribution from which the samples are drawn, which in our case is a combination of the BSDF importance sampling and guiding distribution. By taking its derivative with respect to Θ, we obtain the MC estimate of the gradient $\nabla_\Theta D_{\mathrm{KL}}(\mathcal{D}\|\mathcal{V};\Theta)$:

$$\nabla_\Theta D_{\mathrm{KL}}(\mathcal{D}\|\mathcal{V};\Theta) \approx -\frac{1}{N}\sum_{j=1}^{N}\frac{\mathcal{D}(\omega_j)\nabla_\Theta\mathcal{V}(\omega_j \mid \hat{\Theta})}{\tilde{p}(\omega_j \mid \hat{\Theta})\mathcal{V}(\omega_j \mid \hat{\Theta})}, \tag{9}$$

where the derivatives of the vMF mixtures $\mathcal{V}$ with respect to their parameters Θ are straightforward. The gradients for the trainable NPM parameters Φ could then be obtained via back propagation. Since we use the unbiased MC estimate of the training gradients, the parameters are guaranteed to converge to a local minimum.

In practice, our training sample pairs $(\mathbf{x},\omega_i) \rightarrow L_\mathrm{i}$ are distributed in different spatial positions $\mathbf{x}$, efficiently learning a spatially varying target distribution $\mathcal{D}(\mathbf{x})$. This results in the training objective accounting for the divergence of multiple positions. The expected solution for Φ is thus:

$$\Phi^* = \arg\min_\Phi \mathbb{E}_\mathbf{x}\left[D_{\mathrm{KL}}\left(\mathcal{D}\left(\mathbf{x}\right)\|\mathcal{V};\Theta\left(\mathbf{x}\right)\right)\right]. \tag{10}$$

For our implicit spatial embedding (i.e., grids of latent features, discussed later), this results in the embedding being optimized with all (and only) its nearby samples. When using the gradient descent method, the samples with the largest gradients (i.e., the most important ones for reducing divergence) would dominate, forming a reasonable design choice for better adaptivity.

4.3 Full Integrand Learning

Using path guiding to sample the full integrand $f_s \cdot L_\mathrm{i}\cos\theta_i$ can achieve even better performance, which should incorporate the BSDF term and the cosine term into the target distribution. This is challenging since the guiding distribution is now conditioned on 5D inputs (i.e., outgoing direction ω_o and spatial coordinate $\mathbf{x}$). Previous works fit BSDFs with precomputed parametric models and multiply them with the learned incident radiance distribution to achieve product sampling. However, this often relies on scene-dependent precomputation, discretization over ω_o, and extra computational overhead [Herholz et al. 2016; Ruppert et al. 2020].

Our neural design can naturally handle the conditions with the extra input of ω_i. This is essential since a neural network could approximate arbitrary conditional models if being expressive enough. We later show this improves performance through learning a better guiding distribution, with only modest performance overhead. For clarity, we denote the previous radiance-based method as NPM-radiance, and this version as NPM-product.

Specifically, by supplementing input ω_o, we reformulate the learned distribution (Eq. 6) with the outgoing directions. This enables learning the full integrated as:

$$\mathbf{NPM}_{\mathrm{product}}(\mathbf{x},\omega_\mathrm{o} \mid \Phi) = \hat{\Theta}(\mathbf{x},\omega_\mathrm{o}), \tag{11}$$

where $\hat{\Theta}$ now parameterizes the vMF mixture $\mathcal{V}$ that is trained to approximate the full integrand in Eq. 1, i.e.,

$$\mathcal{V}\left(\omega_i \mid \hat{\Theta}\left(\mathbf{x},\omega_\mathrm{o}\right)\right) \propto f_\mathrm{s}\left(\mathbf{x},\omega_\mathrm{o},\omega_\mathrm{i}\right) L_\mathrm{i}\left(\mathbf{x},\omega_\mathrm{i}\right)\left|\cos\theta_\mathrm{i}\right|, \tag{12}$$

where the cosine term could be approximated with a constant vMF lobe [Ruppert et al. 2020], leaving NPM to focus on the remaining part of the integral. Nonetheless, it is still challenging for neural networks to model a 2D directional distribution conditioned on

5D spatio-directional inputs. We further use the following simple extensions to help the network learn these spatially varying distributions:

Auxiliary Feature Inputs. Following the practices in prior work [Hadadan et al. 2021; Müller et al. 2021], we additionally input the surface normal and roughness as auxiliary features to help the network better correlate the target distribution with e.g., local shading frame (normal) and spatially varying BSDFs (roughness). Experimentally, we find this helps the network to better capture the spatio-directional correlations, while with a small computational overhead due to additional memory traffic.

Input Encoding. It is challenging for a neural network to model the non-linearity between multidimensional inputs and outputs, especially when our outputs are distributions with high-frequency spatial variations. Therefore, we replace the spatial input $\mathbf{x}$ with our trainable multi-resolution spatial embedding (discussed in Sec. 5.1). For the other inputs (e.g., outgoing direction ω_o and surface normals $\mathbf{n}(\mathbf{x})$), we encode them using the spherical harmonics basis, which is previously established in NeRF [Verbin et al. 2022].

5 IMPLEMENTATION

In this section, we provide the technical details that are crucial to the performance and practicality of our NPM implementation.

5.1 Multi-resolution Spatial Embedding

Our implicit NPM representation learns a continuous mapping $\mathbf{NPM}_\Phi : \mathbf{x} \rightarrow \hat{\Theta}$ (with the additional input $\omega_o \in \mathbb{S}^2$ in the extended version), where $\Theta \in \mathbb{R}^{4\times K}$ defines the learned target distribution. While a straightforward solution would be using a multi-layer perceptron (MLP) as the universal function approximator to model $\mathbf{NPM}_\Phi$, we experimentally found it difficult to capture the high-frequency spatial variations of the target distributions.

Therefore, we use a learnable spatial embedding to implicitly encode the learned parametric mixtures. Similar approaches are found successful in recent NeRF-like applications [Müller et al. 2022; Munkberg et al. 2022]. Specifically, we define L 3D uniform grids G_l, each covering the entire scene with a spatial resolution of D_l^3, where G_l denotes the l-th embedding grid. D_l grows exponentially, resulting in multiple resolutions of the embedding. We then assign a learnable embedding (a latent feature vector $v \in \mathbb{R}^F$) to each lattice point of G_l. To query the spatial embedding for $\mathbf{x}$, we bilinearly interpolate the features *nearby* $\mathbf{x}$ for each resolution, and concatenate them to obtain the final embedding $G(\mathbf{x})$. More formally:

$$G(\mathbf{x} \mid \Phi_\mathrm{E}) = \bigoplus_{l=1}^{L} \text{bilinear}\left(\mathbf{x}, V_l\left[\mathbf{x}\right]\right), \; G : \mathbb{R}^3 \rightarrow \mathbb{R}^{L\times F}, \tag{13}$$

where $V_l[\mathbf{x}]$ is the set of features at the eight corners of the cell enclosing $\mathbf{x}$ within G_l. The spatial embedding $G(\mathbf{x})$ is then concatenated with other inputs (e.g., ω_o and auxiliary features) to the MLP for decoding the parameters Θ. We thus formulate the desired mapping (taking Eq. 6 for example) as a two-step procedure:

$$\mathbf{MLP}\Big(G(\mathbf{x} \mid \Phi_\mathrm{E}) \;\Big|\; \Phi_\mathrm{M}\Big) = \hat{\Theta}(\mathbf{x}), \tag{14}$$

where the parameters of the spatial embedding (Φ_E) and the MLP (Φ_M) together constitute the trainable parameters Φ of our implicit representation for NPM. Intuitively, a spatial embedding implicitly encodes the target distribution within a specific spatial region, while the multi-resolution design efficiently accounts for different levels of detail (LOD). By smoothly interpolating between the spatial embedding around positions and decoding them using neural networks, we naturally account for the spatial variations of the target distribution. This also lessens the burden of using a single monolithic MLP as the implicit representation, leaving it mainly focusing on decoding it into parametric models Θ. This significantly accelerates training/inference with a larger memory footprint.

5.2 Online Training Scheme

Renderer Integration. We implement our method on a custom GPU-accelerated renderer based on OptiX [Parker et al. 2010], where the training and inference procedures are integrated into a *wavefront*-style path tracer [Laine et al. 2013]. This design choice allows ray casting, importance sampling, and BSDF evaluation to be performed in coherent chunks over large sets of traced paths by splitting the traditional megakernel path tracer into multiple specialized kernels. This improves GPU thread utilization by reducing the control flow divergence. Most importantly, this allows us to efficiently sample and evaluate the guiding distributions at each vertex along the path in parallel, thus significantly accelerating network training/inference.

Specifically, we place the training/inference samples into *queues*, where the structure-of-arrays (SoA) memory layout is applied to improve memory locality. At each ray intersection of the chunk of traced paths, the queries for guiding distributions within the queue are processed via batched network inference. The sampling and evaluation procedures are then performed, also using specialized kernels, before entering the next ray-cast kernel. This provides our method with maximum parallelism through large-batch training and inference, minimizing the latency caused by waiting network queries, while avoiding inefficient single-sample inference.

Training Scheme. We use the same configuration to train each scene online during rendering, without any scene-specific fine-tuning or pre-computation. During training, we collect MC radiance estimates along each traced path, and split them into mini-batches for training. The optimization step is performed for each spp, which allows drawing samples to be drawn from the latest guiding distribution. The distribution of the samples (for both rendering and training) is thus gets refined as training proceeds. We stop the training process after a fixed fraction of the total rendering budget (either time or sample count). While we always set this to 25% in our experiments, we find our NPM technique converges quickly during training, generally reaching a local minimum after about 150spp, which amounts to about 1000 training steps/batches and 15s (including the runtimes of both training and rendering) on GPU.

5.3 Guiding Network

We implement our network on the *tiny-cuda-nn* framework [Müller 2021] and integrate it into our renderer. The MLP we used (for both NPM-radiance and NPM-product) contains 3 linear layers of width 64. Each layer with ReLU activation, except for the last layer with

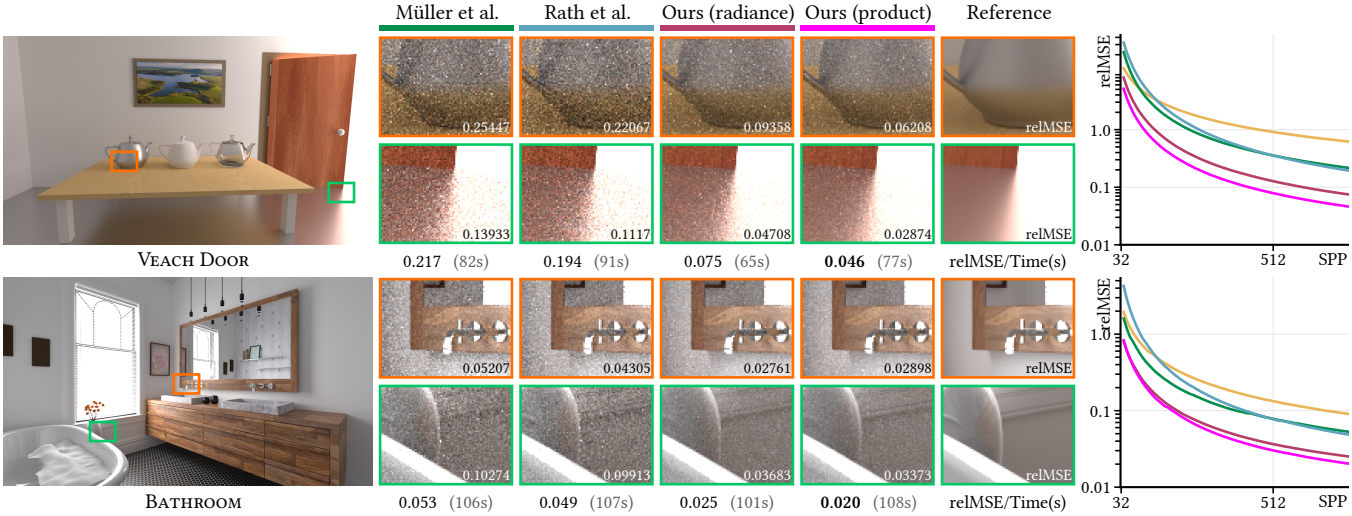


Figure 3: Equal-sample-count (750spp) comparisons for two scenes. We show the error (for both the zoom-in areas and whole images) and time cost of different methods. The yellow plots (as well as the other figures) refer to the results obtained by unidirectional path tracing.

our custom mapping functions (Tab. 1). We let the network output $K = 8$ vMF components, i.e., $\Theta \in \mathbb{R}^{8\times4}$. For the multi-resolution spatial embedding, we use $L = 8$ grids with increasing resolutions for each level. The coarsest level has a resolution of $D_1 = 8$ while the finest level has $D_8 = 86$. The feature of each level contains $F = 4$ floats, resulting in the final spatial embedding $G(\mathbf{x}) \in \mathbb{R}^{8\times4}$. In practice, we find that the performance of the network could be improved by enlarging the capacity of the MLP or the spatial embedding, leaving this a trade-off between quality and speed.

For training, we use a fixed learning rate of 0.005 that is large enough to acquire a fast convergence speed. Adaptive momentum techniques like Adam [Kingma and Ba 2015] are used for more robust training and better convergence. For importance sampling the decoded mixtures, we use the numerically stable strategy for vMF [Jakob 2012]. When inference, we also apply exponential moving average (EMA) to the weights of previous training steps, which better reduces the noise of the MC estimated gradients (Eq. 9).

6 RESULTS AND DISCUSSION

We run all the experiments on an Intel Core i9-11900 CPU and an NVIDIA RTX3070 GPU. Following the similar practices of previous works [Müller 2019; Rath et al. 2020], we disable NEE and Russian roulette for all methods and set the maximum path length to 10. All methods are implemented upon a GPU path tracing renderer.

We render all images at the resolution of 1280×720, and evaluate image quality using mean relative squared error (relMSE). All the images, additional metrics (MAPE and MRSE), and the false-color maps can be interactively inspected with our supplementary viewer.

6.1 Comparisons

Our method is compared against improved PPG [Müller 2019] (an enhanced version of Practical Path Guiding [Müller et al. 2017]), and Variance-aware Path Guiding [Rath et al. 2020]. For the experimental configuration of the compared methods, we use the same as [Rath et al. 2020], except for fixing the BSDF selection probability to 50% (for both ours and the compared methods). Both compared methods used an iteratively reconstructed subdivision structure (i.e., the spatio-directional trees) to account for spatial variations. A total of 10 different scenes were tested.

We first show equal-spp comparisons on two representative scenes. The Veach Door scene features strong indirect illumination that is difficult to handle with BSDF importance sampling, while the Bathroom scene contains many specular and glossy surfaces. As shown in Fig. 3, our proposed method outperforms the other two methods even when only learning incident radiance L_i (NPM-radiance). The noise is alleviated further with our full integrand learning method (NPM-product), since both of the scenes contain glossy surfaces, where the contribution of samples is strongly influenced by the BSDF term. We also note that our method quickly becomes effective at the very beginning of the training process (see the convergence plots in Fig. 3). This indicates a better training efficiency over classical guiding methods, which will be discussed later. Additional results on more test scenes are shown in Fig. 4 and Tab. 2, as well as the convergence plots in Fig. 5.

We then show the results of equal-time comparisons between our method and [Rath et al. 2020] in Fig. 6. Since they do not explicitly learn the product sampling distribution (i.e., conditioned on 5D inputs ω_o and $\mathbf{x}$), we only use our radiance-based method (NPM-radiance) for fair comparisons. Instead of simply learning the incident radiance distribution (L_i), they use an improved target distribution to account for the variance and BSDF (marginalized over ω_o). Our method, on the other hand, achieves better performance by learning L_i only. We attribute this superiority of our method to both the better capacity of capturing spatio-directional correlation and more parallelism.

Table 2: Practical Path Guiding (PPG) [Müller 2019], Variance-aware Path Guiding [Rath et al. 2020], unidirectional path tracing and our method on 10 test scenes. We report relMSE, render time, and speedup using PPG as the baseline. Our NPM technique consistently reduces the error in the test scenes.

			[Müller 2019]			[Rath et al. 2020]			Ours					
	PT (BSDF)		PPG (baseline)			Variance. PG			NPM (radiance)			NPM (product)		
Bathroom	0.0905	48s	0.0530	1.0 ×	106s	0.0485	1.09 ×	107s	0.0251	2.11 ×	101s	**0.0203**	2.61 ×	108s
Bedroom	0.0383	40s	0.0201	1.0 ×	105s	0.0161	1.26 ×	109s	0.0150	1.35 ×	84s	**0.0146**	1.38 ×	90s
Breakfast Room	0.0094	48s	0.0069	1.0 ×	100s	0.0047	1.46 ×	103s	0.0038	1.80 ×	63s	**0.0035**	1.96 ×	71s
Living Room	0.0273	32s	0.0184	1.0 ×	74s	0.0146	1.26 ×	80s	0.0157	1.17 ×	47s	**0.0132**	1.39 ×	54s
Pink Room	0.0046	37s	0.0082	1.0 ×	74s	0.0061	1.34 ×	76s	0.0033	2.42 ×	53s	**0.0026**	3.21 ×	62s
Salle de Bain	0.0819	38s	0.0223	1.0 ×	116s	0.0346	0.64 ×	116s	0.0196	1.14 ×	79s	**0.0140**	1.59 ×	86s
Staircase	0.1812	34s	0.0298	1.0 ×	80s	0.0261	1.14 ×	86s	0.0194	1.54 ×	72s	**0.0172**	1.74 ×	76s
Veach Door	0.6208	33s	0.2167	1.0 ×	82s	0.1945	1.11 ×	91s	0.0750	2.89 ×	65s	**0.0461**	4.69 ×	77s
Veach Egg	8.2918	33s	0.8379	1.0 ×	82s	0.7870	1.07 ×	85s	0.5984	1.40 ×	62s	**0.5352**	1.56 ×	69s
White Room	0.0301	38s	0.0278	1.0 ×	107s	0.0253	1.10 ×	103s	0.0124	2.25 ×	76s	**0.0100**	2.75 ×	87s

6.2 Evaluation

Trainable Spatial Embedding. We analyze the performance of different forms of spatial input encoding in terms of convergence and quality (Fig. 8). The spatial embedding (i.e. parametric encoding) uses trainable latent vector grids to model the spatially-varying target distributions, leaving the MLP to focus on decoding this implicit representation into valid vMF mixtures. The other two variants do not explicitly separate these two tasks by using a monolithic MLP. The addition of spatial embedding significantly improves convergence, and the multi-resolution design further reduces error by better modeling finer-grained spatio-directional correlations. Furthermore, this does not introduce noticeable computational overhead, as only a small fraction of parameters are involved in each training/inference.

Training Efficiency. The effectiveness of guiding methods under small training budgets is important, especially for applications such as preview rendering or even interactive rendering. We analyze the training efficiency of different guiding methods by comparing their performance under different training budgets (31 spp, 63 spp, 127 spp, respectively) in Fig. 7. Our method quickly converges to a good sampling distribution with only a few training samples and less training time cost (e.g., 31 spp with about 3s), thus outperforming previous guiding methods even with much fewer training samples.

6.3 Discussion

Path Guiding Extensions. Our method can be extended with many well-established extensions suggested by previous path guiding algorithms. They are straightforward to be integrated and are promising to further improve our performance. For example: (1) the BSDF selection probability could also be learned by our network or by some other caching strategies [Müller et al. 2020], thus better handling the near-specular surfaces; and (2) the improved variance-aware target distribution [Rath et al. 2020] could be learned to account for the variance within the noisy MC estimates.

Performance Analysis. Our method serves effective means for path guiding while remaining performance practical. Specifically, the measured time cost per NPM evaluation (including both network inference and importance sampling the decoded mixture models) at 1280×720 is about 3ms. Meanwhile, a training step (i.e., a batch of 2^{18} samples) costs about 10ms, indicating that a typical training process (about 1000 training steps) takes about 10s to converge on a single GPU. NPM contains a total of about 2M learnable parameters, resulting in a memory consumption of $<$ 10MB. The compact design of our implicit NPM representation results in less control flow divergence, better memory locality, and better caching performance. Together, this makes our method practical for modern GPU parallelization, which is often harder to achieve with the tree-like spatial subdivision schemes used by most of the previous guiding methods.

Alternative Solutions. Several studies also aim to tackle the parallax issue. Dodik et al. [2022] use spatio-directional mixtures (i.e., conditioned on $\mathbf{x}$ and ω_o) to correlate target distributions with spatial positions. Ruppert et al. [2020] design strategies to warp the guiding distributions in the spatial subdivisions to resemble the true distribution. However, these methods adopt sophisticated strategies that are difficult to parallelize efficiently on GPUs (e.g., batched expectation-maximization (EM) applied to a varying number of mixtures) while requiring extra efforts to fit scene BSDFs for product sampling. In contrast, our method exploits trainable spatial embedding to encode the target distributions while using a decoder MLP to model the non-linearity between spatial features and PMMs in a GPU-friendly manner. Nevertheless, incorporating ideas from these studies, such as adaptively controlling the granularity of learned distributions, may further enhance our method.

7 CONCLUSION, LIMITATIONS AND FUTURE WORK

We present Neural Parametric Mixtures, a novel method for learning the target distributions for path guiding techniques. We use a compact implicit neural representation to encode the spatio-directional parametric distributions. Compared to previous non-neural methods that use explicit spatial subdivision structures to store directional distributions, our continuous implicit representation is simpler and more efficient while naturally avoiding the artifacts (e.g.,

parallax) caused by their discretized subdivision schemes. Our NPM technique could be efficiently trained with stochastic gradient descent to minimize the divergence from the target distribution.

Despite the simplicity and effectiveness of our method, the main limitation resides in the lack of flexibility of our directional distribution representation, i.e., a fixed number of vMF components. While a similar issue exists in classical methods using PMMs [Dodik et al. 2022; Herholz et al. 2016], recent methods achieve more accurate directional distributions by adaptively merging and splitting the vMF components [Ruppert et al. 2020]. This, however, is non-trivial to apply to our NPM technique.

In future work, we will investigate more accurate approaches to implicitly encode parametric distributions while remaining efficient. Finding better basis functions or adaptively controlling the number of output components are two possible but challenging directions. Meanwhile, we would like to improve the efficiency of our method by using either novel architectural designs for neural networks, optimized implementation, or adapting previous extensions to path guiding algorithms. We believe these are important steps to make our method more practical for interactive or even real-time rendering pipelines, as well as other related applications that require fitting distributions with high-frequency spatial variations. In addition, applying our method to bidirectional path tracing [Popov et al. 2015], especially subspace probabilistic connections [Su et al. 2022], will also be an interesting future avenue.

ACKNOWLEDGMENTS

This project was supported by the National Key R&D Program of China (No.2022YFB3303400) and NSFC of China (No. 62172013). We also thank the test scenes providers: Mareck (Bathroom), Slyk-Drako (Bedroom), Wig42 (Breakfast Room, Living Room, Pink Room, Staircase), nacimus (Salle de Bain), Jaakko Lehtinen (Veach Door), Jay-Artist (White Room), as well as the efforts for converting scene formats by Benedikt Bitterli [2016].

REFERENCES

Benedikt Bitterli. 2016. Rendering resources. https://benedikt-bitterli.me/resources/.

Norbert Bus and Tamy Boubekeur. 2017. Double Hierarchies for Directional Importance Sampling in Monte Carlo Rendering. *Journal of Computer Graphics Techniques (JCGT)* 6, 3 (28 August 2017), 25–37. http://jcgt.org/published/0006/03/02

R. R. Currius, D. Dolonius, U. Assarsson, and E. Sintorn. 2020. Spherical Gaussian Light-field Textures for Fast Precomputed Global Illumination. *Computer Graphics Forum* 39, 2 (2020), 133–146.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. 2017. Density estimation using Real NVP. In *International Conference on Learning Representations*.

Stavros Diolatzis, Julien Philip, and George Drettakis. 2022. Active Exploration for Neural Global Illumination of Variable Scenes. *ACM Transactions on Graphics* (2022).

Ana Dodik, Marios Papas, Cengiz Öztireli, and Thomas Müller. 2022. Path Guiding Using Spatio-Directional Mixture Models. In *Computer Graphics Forum*, Vol. 41. Wiley Online Library, 172–189.

Saeed Hadadan, Shuhong Chen, and Matthias Zwicker. 2021. Neural radiosity. *ACM Transactions on Graphics (TOG)* 40, 6 (2021), 1–11.

Sebastian Herholz, Oskar Elek, Jiří Vorba, Hendrik Lensch, and Jaroslav Křivánek. 2016. Product importance sampling for light transport path guiding. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 67–77.

Yuchi Huo, Rui Wang, Ruzahng Zheng, Hualin Xu, Hujun Bao, and Sung-Eui Yoon. 2020. Adaptive incident radiance field sampling and reconstruction using deep reinforcement learning. *ACM Transactions on Graphics (TOG)* 39, 1 (2020), 1–17.

Wenzel Jakob. 2012. Numerically stable sampling of the von Mises-Fisher distribution on S^2 (and other tricks). *Interactive Geometry Lab, ETH Zürich, Tech. Rep* (2012), 6.

Henrik Wann Jensen. 1995. Importance driven path tracing using the photon map. In *Eurographics Workshop on Rendering Techniques*. Springer, 326–335.

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

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. *ICLR* (2015).

Eric P Lafortune and Yves D Willems. 1995. A 5D tree to reduce the variance of Monte Carlo ray tracing. In *Eurographics Workshop on Rendering Techniques*. Springer, 11–20.

Samuli Laine, Tero Karras, and Timo Aila. 2013. Megakernels considered harmful: Wavefront path tracing on GPUs. In *Proceedings of the 5th High-Performance Graphics Conference*. 137–143.

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 *ECCV*.

Thomas Müller. 2019. "Practical Path Guiding" in Production. In *ACM SIGGRAPH 2019 Courses (SIGGRAPH '19)*. ACM, New York, NY, USA, Article 18, 77 pages.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM Trans. Graph.* 41, 4, Article 102 (July 2022), 15 pages.

Thomas Müller, Markus Gross, and Jan Novák. 2017. Practical path guiding for efficient light-transport simulation. In *Computer Graphics Forum*, Vol. 36. Wiley Online Library, 91–100.

Thomas Müller, Brian McWilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. 2019. Neural importance sampling. *ACM Transactions on Graphics (TOG)* 38, 5 (2019), 1–19.

Thomas Müller, Fabrice Rousselle, Alexander Keller, and Jan Novák. 2020. Neural control variates. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–19.

Thomas Müller, Fabrice Rousselle, Jan Novák, and Alexander Keller. 2021. Real-Time Neural Radiance Caching for Path Tracing. *ACM Trans. Graph.* 40, 4, Article 36 (jul 2021), 16 pages.

Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Mueller, and Sanja Fidler. 2022. Extracting Triangular 3D Models, Materials, and Lighting From Images. *CVPR* (2022).

Thomas Müller. 2021. *tiny-cuda-nn*. https://github.com/NVlabs/tiny-cuda-nn

Steven G Parker, James Bigler, Andreas Dietrich, Heiko Friedrich, Jared Hoberock, David Luebke, David McAllister, Morgan McGuire, Keith Morley, Austin Robison, et al. 2010. Optix: a general purpose ray tracing engine. *ACM Transactions on Graphics (TOG)* 29, 4 (2010), 1–13.

S. Popov, R. Ramamoorthi, F. Durand, and G. Drettakis. 2015. Probabilistic Connections for Bidirectional Path Tracing. *Computer Graphics Forum* 34, 4 (07 2015), 75–86.

Alexander Rath, Pascal Grittmann, Sebastian Herholz, Petr Vévoda, Philipp Slusallek, and Jaroslav Křivánek. 2020. Variance-aware path guiding. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 151–1.

Lukas Ruppert, Sebastian Herholz, and Hendrik PA Lensch. 2020. Robust fitting of parallax-aware mixtures for path guiding. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 147–1.

Fujia Su, Sheng Li, and Guoping Wang. 2022. SPCBPT: Subspace-Based Probabilistic Connections for Bidirectional Path Tracing. *ACM Trans. Graph.* 41, 4, Article 77 (jul 2022), 14 pages. https://doi.org/10.1145/3528223.3530183

Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. 2022. Ref-NeRF: Structured View-Dependent Appearance for Neural Radiance Fields. *CVPR* (2022).

Jiří Vorba, Johannes Hanika, Sebastian Herholz, Thomas Müller, Jaroslav Křivánek, and Alexander Keller. 2019. Path Guiding in Production. In *ACM SIGGRAPH 2019 Courses* (Los Angeles, California) *(SIGGRAPH '19)*. ACM, New York, NY, USA, Article 18, 77 pages.

Jiří Vorba, Ondřej Karlík, Martin Šik, Tobias Ritschel, and Jaroslav Křivánek. 2014. On-line learning of parametric mixture models for light transport simulation. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–11.

Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. 2021. PlenOctrees for Real-time Rendering of Neural Radiance Fields. In *ICCV*.

Shilin Zhu, Zexiang Xu, Tiancheng Sun, Alexandr Kuznetsov, Mark Meyer, Henrik Wann Jensen, Hao Su, and Ravi Ramamoorthi. 2021. Hierarchical neural reconstruction for path guiding using hybrid path and photon samples. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–16.

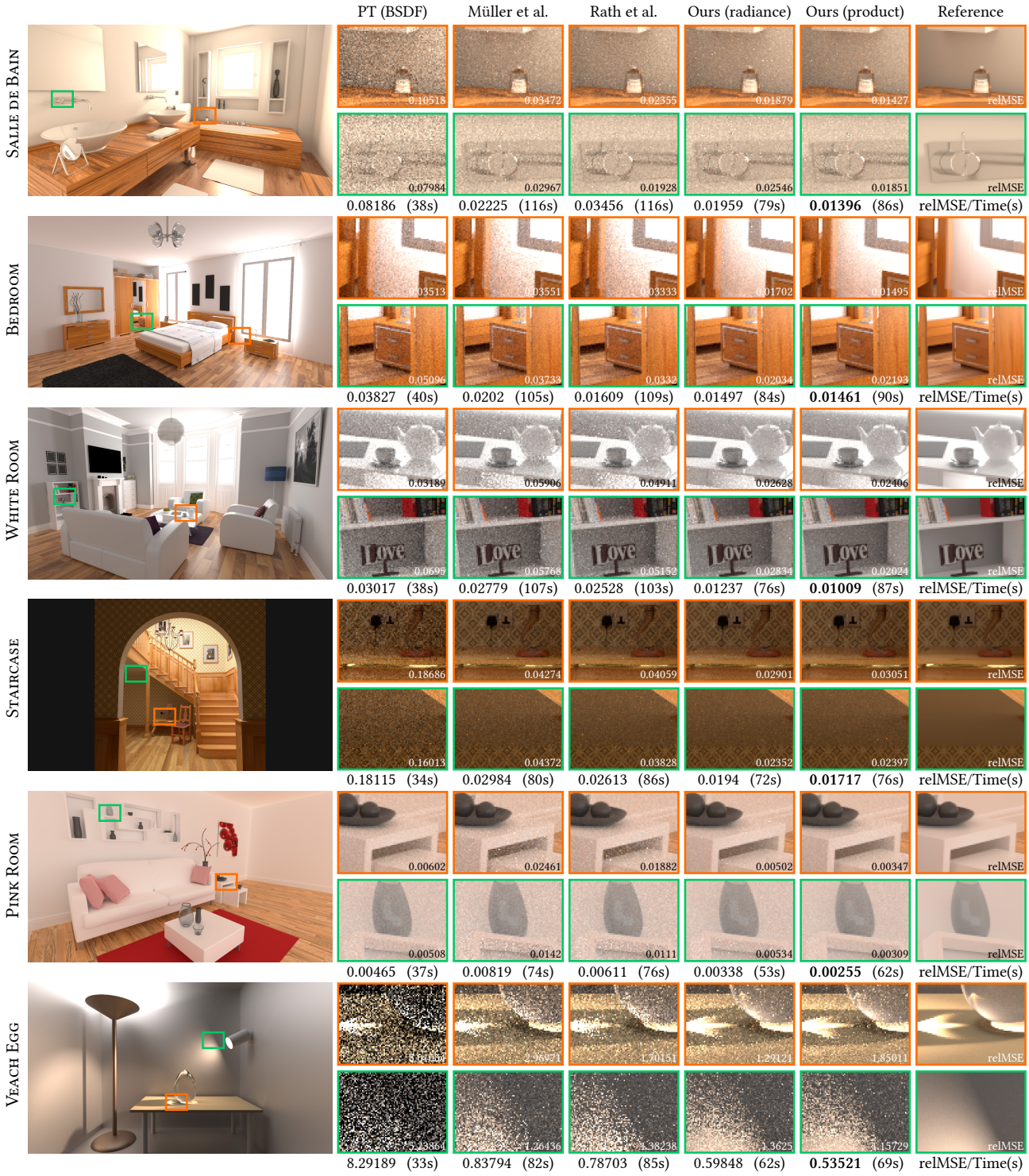


Figure 4: Visual comparisons using the same experimental setup with Fig. 3, all are rendered with 750spp at 1280×720. We use the online training setup for all the guiding methods, i.e., all the samples are included in the final rendering. Our method exhibits better performance than other guiding methods in most scenes by only learning the incident radiance term while further reducing the error by incorporating the BSDF term (i.e., product sampling). More results on other test scenes, additional error metrics and false-color visualizations are provided in our supplementary interactive viewer.

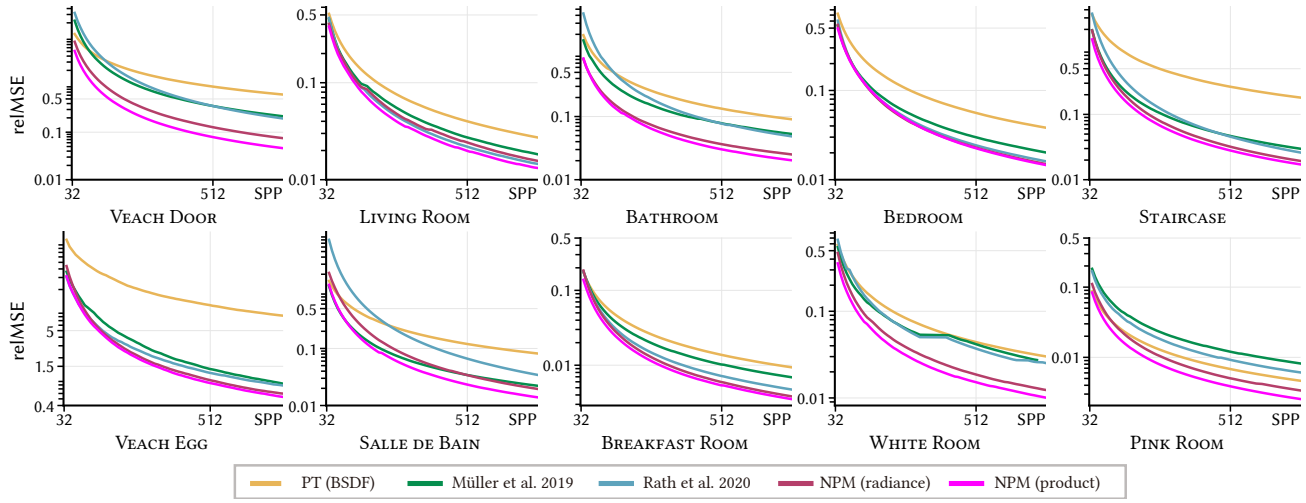


Figure 5: Convergence plots correspond to Fig. 3 and Fig. 4. Unidirectional path tracing with BSDF importance sampling (PT-BSDF), Practical Path Guiding [Müller 2019], Variance-aware Path Guiding [Rath et al. 2020] and our method with different target distributions (NPM-radiance and NPM-product). Our methods consistently outperform these classical guiding methods, and quickly become effective even with a few training samples and short training time (e.g., 30spp, amounting to about 3 seconds on GPU), indicating practicality for preview or even interactive rendering. We attribute this success to the compact implicit representation and better spatial resolution of our method. The image results and detailed statistics could be inspected in the supplemental materials.

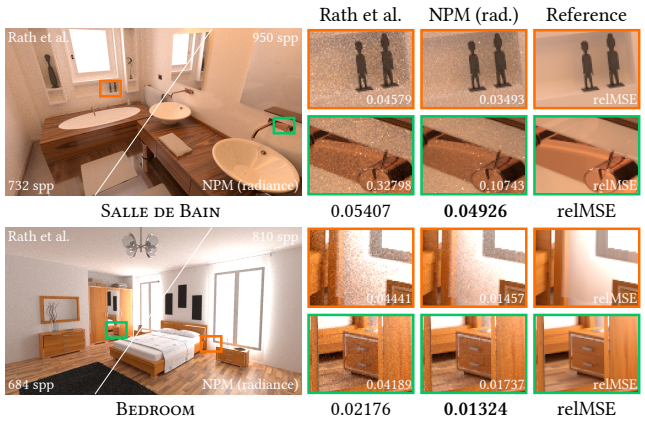


Figure 6: Equal-time comparisons (80s) on two test scenes between NPM(radiance) and Variance-aware Path Guiding [Rath et al. 2020].

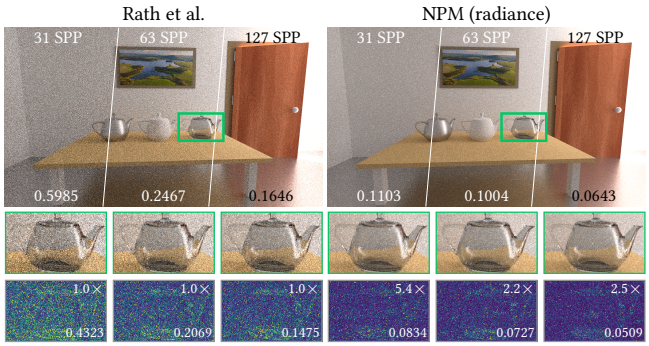


Figure 7: We train each guiding method with small training budgets (31 spp, 63 spp, 127 spp, respectively) and render the scene with 500 spp. Our method outperforms previous methods even with much fewer training samples.

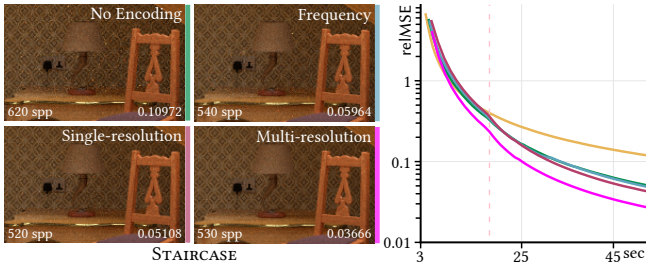


Figure 8: Equal-time comparison (50s) of different input encoding. We report the sample count and error (relMSE) of each method. The dashed line in the plot marks the end of the training phase. The multi-resolution spatial embedding outperforms other methods while remaining training-efficient. Yellow plot refers to path tracing with BSDF importance sampling.