Efficient Neural Path Guiding with 4D Modeling

HONGHAO DONG, Peking University, China
RUI SU, Peking University, China
GUOPING WANG, Peking University, China
SHENG LI[*], Peking University, China

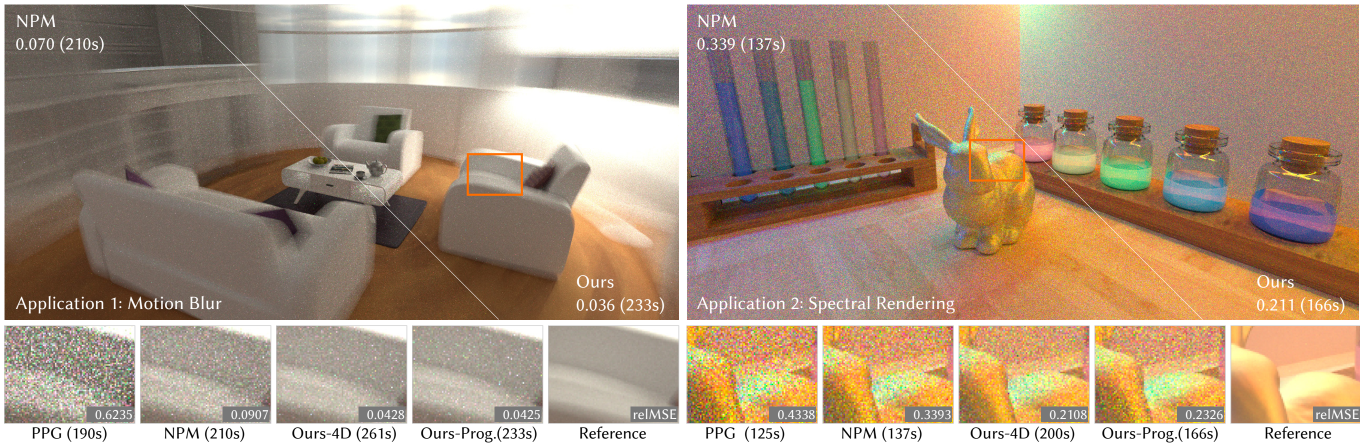

Fig. 1. Our 4D neural path guiding can efficiently model the local target distribution conditioned on additional domains (e.g., time and wavelength). We investigate the benefits of our method on typical distributed ray tracing applications, including motion blur rendering on dynamic scenes, as well as spectral rendering. Our method achieves better importance sampling quality at a reasonable computational overhead.

Previous local guiding methods used 3D data structures to model spatial radiance variations but struggled with additional dimensions in the path integral, such as temporal changes in dynamic scenes. Extending these structures to higher dimensions also proves inefficient due to the curse of dimensionality. In this study, we investigate the potential of compact neural representations to model additional scene dimensions efficiently, thereby enhancing the performance of path guiding in specialized rendering applications, such as distributed effects including motion blur. We present an approach that models a higher dimensional spatio-temporal distribution through neural feature decomposition. Additionally, we present a cost-effective approximate with lower-dimensional representation to model only subspace by progressive training strategy. We also investigate the benefits of modeling correlations with the additional dimensions on typical distributed ray tracing scenarios, including the motion blur effect in dynamic scenes, as well as spectral rendering. Experimental results demonstrate the effectiveness of our method in these applications.

[*]Sheng Li is the corresponding author.

Authors' addresses: Honghao Dong, Peking University, Beijing, China, donghonghao@pku.edu.cn; Rui Su, Peking University, Beijing, China, susurui@pku.edu.cn; Guoping Wang, Peking University, Beijing, China, wgp@pku.edu.cn; Sheng Li, Peking University, Beijing, China, lisheng@pku.edu.cn.

CCS Concepts: • **Computing methodologies** → **Ray tracing**; **Neural networks**.

Additional Key Words and Phrases: Ray Tracing, Global Illumination, Path Guiding, Neural Networks, Feature Decomposition

ACM Reference Format:
Honghao Dong, Rui Su, Guoping Wang, and Sheng Li. 2024. Efficient Neural Path Guiding with 4D Modeling. 1, 1 (November 2024), 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Physically-based rendering requires Monte Carlo (MC) integration to solve difficult numerical integration problems. To enhance the efficiency of light transport simulation, path guiding collects the sample statistics during rendering, and then uses them to fit a better distribution for sample decisions (e.g., scattering direction sampling). To model the target distribution across different shading points, local path guiding techniques typically store them into explicit 3D spatial structures [Müller et al. 2017], and more recently, they have utilized implicit neural representations [Dong et al. 2023; Huang et al. 2024; Müller et al. 2019].

Path guiding techniques effectively reduce the variance for solving the rendering equation [Kajiya 1986], especially in complex scenes with intricate visibility. However, some photorealistic rendering applications involve additional dimensions in the path integral, e.g., distributed ray tracing. For example, motion blur rendering and spectral rendering require integrating the additional time domain $\mathcal{T}$ and the wavelength domain Λ in the path integral, respectively. This also makes the zero-variance objective of path guiding even harder to achieve, as the ideal target distribution now conditions on extra variables from additional dimensions. In practice, previous local path guiding generally neglects these cases [Vorba et al. 2019], i.e., the learned target distribution is marginalized on these additional dimensions. In this work, we show that improvements could be achieved when the correlation between the target distribution and the additional dimensions is taken into consideration.

Directly modeling higher dimensionality is often impractical due to the curse of dimensionality. In this work, we extend path guiding algorithms by practically modeling the additional integral domains in distributed ray tracing, taking a step closer towards the zero-variance objective in these applications. We first extend the 3D grid-based representations [Müller et al. 2022; Takikawa et al. 2023] into high dimensional representation, then present a neural feature decomposition approach to represent it with multiple low dimensional representations for practical performance. We also present a less costly variant that uses a low-dimensional representation to model only a subspace of the high-dimensional representation. Combined with our progressive training strategy, this approximation achieves similar sampling quality throughout the full rendering process with less computational overhead. Finally, we validate the effectiveness of our method under distributed rendering applications that involve extra integral domains, including motion blur for dynamic scenes and spectral rendering.

Overall, our main contributions include:

- We extend the local path guiding to model the target distribution conditioned on 4D input spaces, and evaluate its effectiveness on typical distributed ray-tracing applications.
- We propose to decompose the high-dimensional feature into multiple lower-dimensional representations, providing practical means to model the target distributions conditioned on additional dimensionalities.
- We present a cost-effective variant that uses reduced representations to model only subspaces of the high-dimensional correlations using a progressive training strategy.

2 RELATED WORK

Path Guiding. Path guiding techniques collect radiance samples to construct better sampling distributions. Most guiding methods learn the target distribution of local shading points, and fit them into directional representations (e.g., directional trees or mixture models) [Müller et al. 2017; Vorba et al. 2019, 2014]. They are then used to sample the scattering directions during forward or bidirectional path construction, namely the local guiding approaches. To account for the spatial variations of the local target distributions, they are generally stored in spatial acceleration structures, e.g., kd-tree and octree.

Recently, a new trend of research leverages neural networks to facilitate path guiding. Some works use deep learning to refine the quality of conventional methods, e.g., using convolutional networks to reconstruct the noisy distribution within spatio-directional trees. Another type of method directly encodes the target distributions using networks without involving explicit data structures. The networks are shown to be capable of learning the full importance sampling process [Müller et al. 2019; Zheng and Zwicker 2019], or fitting the target distribution into explicit mixture models for efficient sampling [Dong et al. 2023; Huang et al. 2024]. These explicit distributions provides attractive alternatives when lightweight computation is in demand [Huang et al. 2024].

Some physically-based rendering applications require extra nested integration on additional domains, e.g., motion blur and spectral rendering. However, previous local guiding methods generally neglect these effects, i.e., they learn the marginalized target distribution over the extra domains [Vorba et al. 2019]. In this work, we extend local guiding methods to model the additional integral domains and demonstrate its benefits on typical rendering applications.

Neural Representation for Rendering. Recently, deep learning techniques have been widely applied to facilitate rendering applications, i.e., neural rendering. Among various research avenues, neural scene representation focuses on encoding the scene with neural networks, and has been gaining popularity after the success of neural radiance fields (NeRF) [Mildenhall et al. 2020]. Neural scene representation has also been applied to enhance ray-traced photorealistic rendering, e.g., for efficient caching and querying of spatial radiance [Hadadan et al. 2021; Müller et al. 2021], or encoding the target distribution for path guiding [Dong et al. 2023; Huang et al. 2024].

Research has also found that input encoding is crucial for network performance. Early methods use frequency encoding to project the 3D coordinate inputs to high dimensional space [Mildenhall et al. 2020]. More recent studies use grid-based representations to encode the spatial features, which are stored in 3D grids or hash tables [Müller et al. 2022; Takikawa et al. 2023] for efficient querying before feeding into the network. The trainable encoding technique lessens the burden of the pure MLP methods, resulting in lower computational cost by keeping the size of MLP sufficiently small [Takikawa et al. 2023]. The spatial feature embedding technique has later been extended to model dynamic scenes. Li et al. [2022] use temporal feature embedding to network inputs, while Fridovich-Keil et al. [2023] and Cao and Johnson [2023] model dynamic scenes with 4D feature voxels. The 4D feature grids are decomposed into multiple orthogonal feature grids/planes for practical computational and storage overhead, namely the k-plane method. Inspired by these works, we use feature decomposition to extend previous local guiding methods, so as to model the correlations of the target distribution with additional dimensions beyond the 3D spatial coordinates.

3 PRELIMINARY

Monte Carlo Rendering. 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 $N = 1$:

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

where $\langle L_r(\mathbf{x}, \omega_o) \rangle$ is an unbiased estimate of the outgoing radiance $L_r(\mathbf{x}, \omega_o)$, and ω_i is the incident direction sampled with some directional probability distribution $p(\omega_i \mid \mathbf{x}, \omega_o)$. The variance of this estimator $V[\langle L_r \rangle]$ can be reduced if the sampling distribution p resembles the shape of the integrand, where zero-variance is achieved when $p \propto f_s \cdot L_i \cos\theta_i$. Local path guiding algorithms approximate the zero-variance goal by fitting a local sampling distribution for $\mathbf{x}$ using previous radiance samples. Typically, they seek to learn the incident radiance L_i or the full integrand $f_s \cdot L_i \cos\theta_i$, which is used to facilitate the construction of the path at each vertex.

Distributed Ray Tracing. Generally, the value of a pixel I_{px} could be obtained by integrating the outgoing radiance $L_o(\mathbf{x}, \omega_o)$ (Eq. 1) from the visible surfaces to the pixel, where L_o is conditioned on the primary intersection $\mathbf{x}$ and the view direction ω_o. However, some physically-based visual effects require additional integration on other domains, which is often hard to approximate with a single path sample. Typical examples include motion blur and spectral effects (the so-called distributed effects [Cook et al. 1984]), which require integration along time t and wavelength λ domains, respectively. This also makes the path integral have higher dimensionality, and extra paths need to be sampled along the additional dimensions.

This also poses challenges to path guiding algorithms since the zero-variance target distribution now depends on additional variables, a factor not commonly addressed in previous path guiding research. In this work, we introduce a practical method to extend path guiding algorithms to account for these effects and explore the advantages of modeling the target distribution across these additional domains.

Parametric Network Encoding. Trainable feature grid is a widely used network encoding, and is found to be effective for augmenting the capability of the MLP, while allowing it to be small for faster inference [Takikawa et al. 2023]. Following the practices of previous work [Dong et al. 2023; Hadadan et al. 2021], we also leverage the feature-grid-based representation with multiple resolutions to implicitly encode the guiding distributions. Taking the 3D case for example, a feature grid consists of L 3D grids with different voxel resolutions. For each grid G_l of the l-th level, a trainable latent embedding $y \in \mathbb{R}^F$ is assigned to each of its grid point. To efficiently query the encoded coordinate $\mathbf{x}$, the features on the corners of the voxel encapsulating $\mathbf{x}$ is linearly interpolated, and then aggregated with other levels to get the final encoding $G(\mathbf{x})$:

$$G(\mathbf{x}) = \bigoplus_{l=1}^{L} \text{trilinear}\left(\mathbf{x}, V_l[\mathbf{x}]\right), \tag{3}$$

where $V_l[\mathbf{x}]$ is the set of features nearby $\mathbf{x}$ within the l-th grid G_l, and $G : \mathbb{R}^3 \to \mathbb{R}^{L\times F}$ is the parametric encoding function.

Neural Mixture Fields. Recent study found coordinate-based neural representation could be used to encode the spatial-varying target distribution, thus facilitating path guiding by providing a more continuous neural spatial representation [Dong et al. 2023; Huang et al. 2024]. Specifically, they use lightweight MLPs and optionally a spatial feature grid [Müller et al. 2022] to decode them into parametric mixtures:

$$\mathbf{MLP}\Big(G(\mathbf{x} \mid \Phi_E) \mid \Phi_M\Big) = \hat{\Theta}(\mathbf{x}), \tag{4}$$

where $\hat{\Theta}$ is the parameter defining a vMF mixture, while Φ_E and Φ_M are the parameters of the feature grids (Eq. 3) and the MLP, respectively. The parameters could be optimized by minimizing the learned distribution $\mathcal{V}$ and the target distribution $\mathcal{D}$:

$$D_{KL}(\mathcal{D} \| \mathcal{V}; \Theta) = \int_\Omega \mathcal{D}(\omega) \log \frac{\mathcal{D}(\omega)}{\mathcal{V}(\omega \mid \hat{\Theta})} \, d\omega, \tag{5}$$

where $\mathcal{D}$ is defined by the network output $\hat{\Theta}$. Sparse estimates of $\mathcal{V}$ can be obtained during the rendering process, which could then be used to estimate the divergence (Eq. 5) with MC integration and for efficient back-propagation on the fly. Concurrent research also proposed a similar technique while proposing a new mixture model to better account for anisotropy [Huang et al. 2024]. Compared to the expressive implicit density modeling techniques (e.g., normalizing flow [Müller et al. 2019]), these explicit distributions provides attractive alternatives when lightweight computation is in demand [Huang et al. 2024]. In this work, we extend their method to higher dimensionality, thereby modeling the correlation of the target distribution over the extra domains.

4 METHOD

Based on the previous work [Dong et al. 2023; Huang et al. 2024], our goal is to model the correlation of the local target distribution on extra dimensionality, which is required to better approximate the zero-variance goal in rendering distributed effects. More formally, the ideal target distribution $\hat{p}$ is now conditioned on variables $\mathbf{u}$ from the extra domains $\mathcal{U}$:

$$p(\omega_i \mid \mathbf{x}, \omega_o, \mathbf{u}) \propto f_s\left(\mathbf{x}, \omega_{o,}, \omega_i, \mathbf{u}\right) L_i\left(\mathbf{x}, \omega_i, \mathbf{u}\right) \left|\cos\theta_i\right|, \tag{6}$$

where $\mathbf{u} = \{u_1, \cdots, u_N\}$ is the parameterized vector consisting of the variables, and u_i could be shutter time t for motion blur rendering, or wavelength λ for spectral rendering, for examples.

It is non-trivial to extend the representation used in previous work. For the grid-based representation, modeling one extra dimension needs about 2× querying overhead for interpolating nearby features and $D\times$ storage cost (D is the resolution of the extra dimension). A similar situation exists for the traditional explicit data structures, where directly extending the kd-tree needs $D\times$ storage cost and $\log D\times$ computational cost. Both are impractical for a resolution D needed to model the fine-grained features within the new dimension.

In this section, we first propose decomposing the high-dimensional grid-based neural representation into multiple low-dimensional feature grids, which have practical applicability to model the target distribution for guiding distributed effects. We then show that in certain scenarios where modeling the full parameter space is unnecessary (e.g., offline rendering), low dimensional representations could be used to model a subspace once at a time while combined with progressive training to guide the full rendering process.

In our practice, we mainly focus on applications with one extra integral domain ($N = 1$), which is sufficient for motion blur in dynamic scenes and spectral rendering. It is also possible to extend our method to model a higher number of additional domains. However,

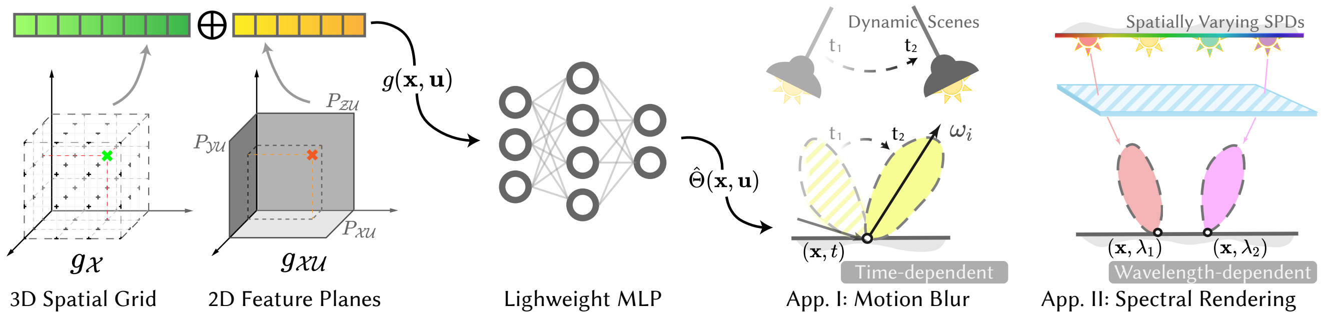

Fig. 2. Schematic of our method. To model the target distributions within the 4D space $\{\mathcal{X}, \mathcal{U}\}$, we propose to decompose the high dimensional representation into a 3D spatial grid $g_{\mathcal{X}}$ (for modeling spatial-only features), as well as three 2D feature planes $g_{\mathcal{XU}}$. The queried features $g(\mathbf{x}, \mathbf{u})$ from both representations are then concatenated as input of the MLP, which is used to decode the parameterized guiding distribution conditioned $\hat{\Theta}(\mathbf{x}, \mathbf{u})$ on $\mathbf{u}$. We show two typical applications, including motion blur rendering for dynamic scenes, as well as spectral rendering. The target distribution is conditioned on time (t) and wavelength (λ) in two applications, respectively.

our experiments show that the computational cost often outweighs the performance gains in most cases, as discussed later.

4.1 Neural Feature Decomposition

We first focus on extending modeling one extra integral domain required by the distributed effects by extending the 3D grid-based representation used in previous methods [Dong et al. 2023]. A straightforward extension would be using a 4D grid, which would result in impractical computational and storage costs. Inspired by the feature decomposition technique initiated by NeRF applications [Fridovich-Keil et al. 2023], we use additional low-dimensional feature grids to model the correlations introduced by the extra dimensions, while letting the original 3D feature grid focus on modeling the spatial-only features decoupled from extra dimensions. Specifically, we decompose the high dimensional grid into two orthogonal components with low dimensional representations:

- A 3D feature grid $g_{\mathcal{X}}$ for modeling spatial features decoupled from the extra domains.
- Three 2D feature planes $g_{\mathcal{XU}}$ for modeling the correlation between the extra domains and the spatial coordinates.

where we formulate each component respectively as:

$$\begin{aligned} g_{\mathcal{X}}(\mathbf{x}) &= G(x, y, z) \\ g_{\mathcal{XU}}(\mathbf{x}, \mathbf{u}) &= P_{XU}(x, u) \oplus P_{YU}(y, u) \oplus P_{ZU}(z, u), \end{aligned} \tag{7}$$

where $\mathbf{x}$ denotes the 3D spatial coordinates and $\mathbf{u}$ denotes the variable from extra dimensions, both normalized to $[0, 1]$ ranges. $G(\cdot, \cdot, \cdot)$ is the 3D grid for modeling decoupled spatial features, while $P(\cdot, \cdot)$ is the 2D grid for modeling the correlations between the extra variable u and the spatial coordinates $\mathbf{x}$. $\oplus$ is the aggregation operator for feature reduction between different feature planes, where we found simple concatenations perform sufficiently well in our experiments. The features are then concatenated together to obtain the final encoding $g(\mathbf{x}, \mathbf{u})$ as the input of the decoder MLP, which predicts the target distribution $\hat{\Theta}(\mathbf{x}, \mathbf{u})$ conditioned on $\mathbf{u}$ (Eq. 4):

$$g(\mathbf{x}, \mathbf{u}) = g_{\mathcal{X}}(\mathbf{x}) \oplus g_{\mathcal{XU}}(\mathbf{x}, \mathbf{u}). \tag{8}$$

Discussion. We introduce a practical grid-based representation for encoding correlations within a 4D scene space. While it is possible to extend this approach to model additional domains beyond the 4D space by introducing extra 2D feature planes, $g_{\mathcal{U}}(\mathbf{u})$, to capture correlations between different variables u_i from other domains, we found that the additional computational cost often outweighs the quality gains when modeling more than two extra dimensions. Therefore, in this work, we focus on modeling just one extra dimension $N = 1$, which is sufficient to guide distributed effects.

Using a 2D plane decomposition strategy (as used in K-planes [Fridovich-Keil et al. 2023]) to encode spatial features could be an alternative. However, considering that spatial correlations are more crucial to overall reconstruction quality compared to correlations from additional dimensions, employing a 3D grid $g_{\mathcal{X}}$ is more practically effective.

4.2 Progressive Training Scheme

For typical applications in rendering distributed effects, a key observation is that the target distribution $p(\omega_\mathrm{i} \mid \mathbf{x}, \omega_\mathrm{o}, \mathbf{u})$ only *partially* changes if the extra variables $\mathbf{u}$ is perturbed. Intuitively, if a neural representation Φ_1 is trained to encode the target distribution $p_{\mathbf{u}_1}$ at $\mathbf{u}_1$, then the learning target $p_{\mathbf{u}_2}$ at $\mathbf{u}_2$ should be similar to $p_{\mathbf{u}_1}$. By leveraging Φ_1 as the initial weights, we can use transfer learning to efficiently fine-tune the network to the new state Φ_2 with potentially few training samples.

To further exploit the correlations of $p_{\mathbf{u}}$ between neighboring $\mathbf{u}$, we reorganize both the rendering and the training process in a progressive manner along the extra dimension. Specifically, we use stratified sampling to solve the integral of the extra dimension $\mathcal{U}$, thus allowing splitting the rendering process by subdividing the domain into M small strata $[u_i, u_{i+1}]$. During rendering the current stratum, only the target distribution within this interval is modeled. By assuming the target distribution $p_{\mathbf{u}}$ have no drastic changes in $[u_i, u_{i+1}]$, we do not use grid-based representation to model the variation of $p_{\mathbf{u}_1}$. More formally, we disable $g_{\mathcal{XU}}(\mathbf{x}, \mathbf{u})$ (Eq. 7), while letting $g_{\mathcal{X}}(\mathbf{x})$ to model the marginalized target distribution over the small interval $[u_i, u_{i+1}]$:

$$p_{\mathbf{u}_i}(\omega_\mathrm{i} \mid \mathbf{x}, \omega_\mathrm{o}) \propto \int_{\mathbf{u}_i}^{\mathbf{u}_{i+1}} f_\mathrm{s}\left(\mathbf{x}, \omega_{\mathrm{o},}, \omega_\mathrm{i}, \mathbf{u}\right) L_\mathrm{i}\left(\mathbf{x}, \omega_\mathrm{i}, \mathbf{u}\right) \left|\cos\theta_\mathrm{i}\right| \mathrm{d}u, \tag{9}$$

where $|u_i - u_{i+1}|$ is designed to be small. By assuming no high-frequency changes within an individual stratum, we only provide $\mathbf{u}$ as auxiliary network inputs to model the effect of $\mathbf{u}$ on the target distribution, rather than using the grid-based representation, as discussed in Sec. 4.1.

We combine the above strategy for modeling the target distribution in individual strata with progressive training to achieve guidance for the full rendering process. Specifically, we iteratively render each stratum. At the beginning of each stratum, we use a fixed amount of samples to fine-tune the network, where the samples are generated on the fly during rendering the current stratum. We let the first K strata be the burn-in phase, where a larger amount of samples are used for training, as illustrated in Fig. 3.

Discussion. We introduce a progressive training approach that guides the full rendering process using a more compact representation than the method described in Sec. 4.1, yet achieves comparable sampling quality. This essentially exploits the fast-adaption intrinsic of the neural representation, which the traditional tree-like structures struggle to achieve. However, this method also brings higher training cost while limiting the applicability to where the whole scene space $\{\mathcal{X}, \mathcal{U}\}$ needs to be modeled, e.g., for interactive rendering and fast previewing, which will also be discussed later.

We also note that it is straightforward to combine this rendering scheme with adaptive sampling techniques in the extra domains [Hachisuka et al. 2008; West et al. 2020]. Intuitively, this reduces the variance of the nested integral, while alleviating the burden of the network by allocating more samples to the important subspaces.

4.3 Optimization

We aim to efficiently train our neural representation in an online manner. Training strategies for density estimation using neural networks have been studied in previous work [Dong et al. 2023; Huang et al. 2024; Müller et al. 2019], and are shown to have faster convergence than explicit structures [Dong et al. 2023], e.g., directional trees [Müller et al. 2017]. We adopt a similar optimization technique by estimating the K-L divergence, but extend the training objective to additional dimensions expanded by $\mathbf{u}$ ($\mathbf{u}$ is 1D in our

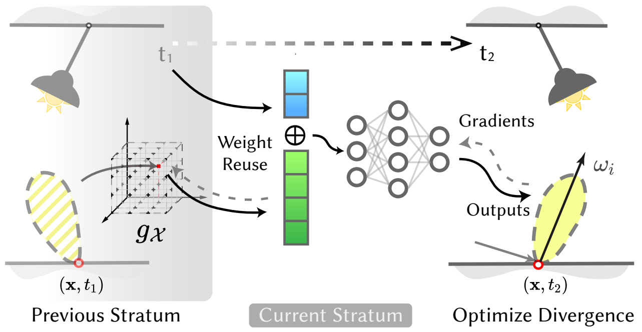

Fig. 3. Illustration of our progressive training method using motion blur as an example. We use stratified sampling to sample paths along the time dimension. To handle the dynamic changes of geometry and lighting, we reuse the network weights of the previous stratum, and exploit newly generated samples to fine-tune the network after the stratum is switched.

main experiments). Specifically, for given shading position $\mathbf{x}$ and the extra scene variable $\mathbf{u}$, we use MC integration to optimize the directional guiding distribution $\mathcal{V}$ predicted by the network:

$$\nabla_{\Theta} D_{\mathrm{KL}}(\mathcal{D} \| \mathcal{V}; \Theta) \approx -\frac{1}{N} \sum_{j=1}^{N} \frac{\mathcal{D}(\omega_j) \nabla_{\Theta} \mathcal{V}(\omega_j \mid \hat{\Theta})}{\tilde{p}(\omega_j \mid \hat{\Theta}) \mathcal{V}(\omega_j \mid \hat{\Theta})}, \tag{10}$$

where $\mathcal{V}$ is defined by the network output parameters $\hat{\Theta}$, and is conditioned on $(\mathbf{x}, \mathbf{u})$, and other inputs and network parameters Φ.

We exploit the samples generated on-the-fly for network optimization. When learning the full space of distribution (Sec. 4.1), the network's loss is defined by the integrated divergence over the surface $\mathcal{S}$ and the space expanded by the extra dimensions $\mathcal{U}$:

$$\mathcal{L}(\Phi) = \int_{\mathcal{U}} \int_{\mathcal{S}} p_{XU}(\mathbf{x}, \mathbf{u}) \nabla_{\Phi} D_{\mathrm{KL}}(\mathcal{D} \| \mathcal{V}; \hat{\Theta}(\mathbf{x}, \mathbf{u}))\, \mathrm{d}\mathbf{x}\, \mathrm{d}\mathbf{u}, \tag{11}$$

where p_{XU} is the unknown joint PDF of the samples over $\{\mathcal{S}, \mathcal{U}\}$.

As for the case where only a subspace of $\mathcal{U}$ is modeled within a small stratum (Sec. 4.2), we adapt transfer-learning to reuse the trained network parameters Φ from the last stratum by assuming that the target distributions only have slight changes for similar $\mathbf{u}$ in neighboring strata, as shown in Fig. 3.

5 IMPLEMENTATION

We implemented our feature decomposition method (Sec. 4.1), as well as the progressive training and stratified scheme (Sec 4.2) upon neural parametric mixtures [Dong et al. 2023] on a GPU renderer using CUDA and OptiX [Parker et al. 2010].

Network Architecture. We use a similar network architecture as used in neural parametric mixtures (NPM), while implementing the feature decomposition by modifying the `tiny-cuda-nn` framework [Müller 2021]. We use an MLP with 3 linear layers and 64 neurons, with ReLU activation after each layer. The last layer has a width of 32 and is mapped to 8 vMF components using the custom mapping as described in [Dong et al. 2023]. For the 3D grid $g_{\mathcal{X}}(\mathbf{x})$ modeling decoupled spatial features, we use a multi-resolution grid $G(\cdot,\cdot,\cdot)$ with 8 levels, where the coarsest resolution is 8 with a growth factor of 1.4× at each level. For 2D grids $g_{\mathcal{X}\mathcal{U}}(\mathbf{x}, \mathbf{u})$ modeling correlations with extra dimensions (Sec. 4.1), we make each feature plane $P(\cdot,\cdot)$ consisting of 8 levels, with the coarsest resolution at 8 and the growth factor set to 2×. The 3D grid G and the 2D plane P have a latent vector of length $F = 4$ and $F = 2$ at each of their grid points per level, respectively.

Training and Rendering. We leverage a similar training scheme as [Dong et al. 2023], where training and inference are interleaved at each frame. For the variant that models the full 4D correlation (Sec. 4.1), we use the first 35% of the sample count for training, where each frame is trained for `n_batches` = 4 steps with `batch_size` = 2^{18} samples. As for the low dimensional variant with progressive training (Sec. 4.2), we only train `n_batches` = 2 steps per frame with the same `batch_size`. Due to the reduced total sample count, we only gather samples from 50% of the paths that we randomly chose. During rendering each frame, we collect the radiance samples along the vertices of each path parallelly and gather them into a queue with a maximum size of `n_batches`×`batch_size`. Additional samples that exceed the queue size are discarded for less memory traffic. Both variants are optimized with Adam [Kingma and Ba 2015] with a 0.008 learning rate. The exponential moving average (EMA) strategy is further applied to network weights with a decay rate of 0.95 and 0.9 for the two variants, respectively. We also note that the training samples with zero target value $\mathcal{D}$ could be culled to accelerate training, as they have zero derivatives (Eq. 10) and thus not contribute to the gradient accumulation process. While we did not include this specific optimization in current implementation, it is expected to provide efficiency gains in complex scenes where many light paths have zero contribution and could be culled.

MIS Integration. Experimentally, the learned mixtures struggle to fit the case where the BSDF lobe is extremely sharp (e.g., when roughness is very small). Following previous practices of guiding methods [Müller et al. 2017; Ruppert et al. 2020], we combine our sampling technique with a defensive strategy using multiple importance sampling (MIS) [Veach 1998]. Specifically, we randomly choose to sample from the network predicted mixtures or BSDF with a fixed ratio of $\lambda_B = 0.5$, and do not perform guiding for specular surfaces. In practice, we also apply MIS compensation [Karlík et al. 2019] to the target distribution, as proposed by Rath et al. [2020]. This allows the network to learn effects that are not handled well by BSDF sampling. Specifically, the target distribution becomes:

$$p_{\mathrm{g}}(\omega_{\mathrm{i}} \mid \mathbf{x}, \omega_{\mathrm{o}}, \mathbf{u}) \propto w_{\mathrm{g}} f_{\mathrm{s}} \left(\mathbf{x}, \omega_{\mathrm{o}}, \omega_{\mathrm{i}}, \mathbf{u}\right) L_{\mathrm{i}} \left(\mathbf{x}, \omega_{\mathrm{i}}, \mathbf{u}\right) \left|\cos\theta_{\mathrm{i}}\right|, \tag{12}$$

where $w_{\mathrm{g}} = \frac{(1-\lambda_B)p_{\mathrm{g}}}{(1-\lambda_B)p_{\mathrm{g}}+\lambda_B p_B}$ is the MIS weight of the guiding distribution. The same strategy is also applied to NPM for fair comparisons.

6 EXPERIMENTS AND RESULTS

We compare our method with Practical Path Guiding (PPG) [Müller et al. 2017] and Neural Parametric Mixtures (NPM) [Dong et al. 2023]. All the compared methods are implemented on GPU, where we use the implementation of NPM from authors, and port the CPU implementation of PPG to CUDA. For simplicity of comparisons, we let the guiding methods learn the incident radiance $p \propto L_{\mathrm{i}}$, i.e., we disable the learnable selection probability of PPG, while disabling the product distribution learning for NPM. We do not use pixel weighting schemes for all the guiding methods. We render the test scenes at 1280×720, and report the mean relative squared error (RelMSE) for each method. We limit the maximum path length to 10 and disabled Russian roulette and NEE for all scenes following prior practices [Dodik et al. 2022; Müller et al. 2019]. We use spectral rendering for all experiments and enabled HeroMIS [Wilkie et al. 2014] (simultaneously evaluating 4 wavelengths per path), although the heavy use of dispersive materials (i.e., BSDFs with wavelength-dependent IoR) in our spectral test scenes often makes it ineffective. Both training and rendering are performed on a single RTX4070 Laptop (8GB) GPU, or an RTX 4080S GPU if otherwise mentioned.

6.1 Rendering Distributed Effects

We compare the performance of our method with previous guiding methods in typical distributed ray-tracing applications including motion blur and spectral rendering, as these effects require the target distribution changes correlated with other dimensions. For other distributed effects like depth of field (DoF) and soft shadows, the local target distribution (Eq. 6) does not change along extra integral domains, and thus is sufficient to model with 3D representations, as used in previous local guiding methods [Müller et al. 2017].

Table 1. Equal-time comparisons of our method against previous guiding methods. For each scene, we report the RelMSE metric and the sample count of each method. Our method produces lower errors in most scenes.

Scene / RMSE	PT	NPM[2023]	Ours(4D)	Ours(Prog.)
Living-Room	0.1896	0.0469 🟤	0.0347 ⚪	0.0311 🟡
	3220 spp	1910 spp	1420 spp	1500 spp
Tower	0.1338	0.0704 🟤	0.0696 ⚪	0.0359 🟡
	3540 spp	1790 spp	1400 spp	1500 spp
Veach-Egg	1.2370	0.3979 🟤	0.3408 🟡	0.3466 ⚪
	7430 spp	1910 spp	1470 spp	1500 spp
Interior	0.5296	0.3905 ⚪	0.4239 🟤	0.3105 🟡
	4180 spp	1940 spp	1380 spp	1500 spp
Liquids	0.2170 ⚪	0.2679	0.2319 🟤	0.2054 🟡
	4310 spp	1850 spp	1360 spp	1500 spp

Motion Blur. Motion blur is a typical distributed effect involving dynamic scenes. This requires the path integral with an extra dimension of time t, where the local target distribution (Eq. 6) is easy to change drastically due to dynamic lighting, materials, geometry visibility, etc. Oftentimes, it is inefficient to learn a marginalized local radiance distribution over t, which fails to capture transient effects that occur within a short time interval.

We let our method learn a local target distribution $p_{\mathrm{g}}(\omega_{\mathrm{i}} \mid \mathbf{x}, t)$ that is additionally conditioned on time. In Figure 5, we compare our method with previous guiding methods, where the target distribution is only conditioned on 3D spatial coordinates $\mathbf{x}$. In the Tower scene, we use rigid body physics to simulate the process of an object crushing onto a tower of cubes. The moving cube with emissive material is the major illuminant in the scene, resulting in drastic changes in light visibility and, thus, highly correlated target distribution with time. For the Room scene, we make the perimeter of the room rotate, leading to changing lighting including the occluded lamp and the incident radiance from the environment. By additionally modeling the correlation over time, our methods increase the probability of finding high-contribution paths at different time points. The progressive training variant of our method (`Ours-Progressive`) successfully adapts to new time strata by reusing the weights from the previous time point using transfer learning, achieving a reasonable computational cost by using a reduced grid-based representation than `Ours-4D`.

We further visualize the animated scene and the learned target distribution in Fig. 7. For each scene, we show the distribution predicted by NPM and Ours(4D), respectively. NPM learns a marginalized distribution over temporal scene changes, which tends to be blurry. On the other hand, our method produces time-dependent distributions that are more fine-grained and accurate among different time points.

Spectral Rendering. Spectral rendering extends the path integral over the wavelength domain Λ, thereby achieving more physically accurate light-scene interaction. Multiple causes in spectral scenes result in the target distribution being correlated with the wavelength λ, e.g., (1) wavelength-dependent BSDFs (e.g., dispersive dielectrics); (2) non-constant spectra of the texture reflectance; and (3) colored lights with different spectral power distributions (SPDs). Previous guiding methods generally omit this correlation by learning the marginalized (averaged) distribution p_{g} over Λ for shading points $\mathbf{x}$:

$$p_{\mathrm{g}}(\omega_{\mathrm{i}} \mid \mathbf{x}, \omega_{\mathrm{o}}) \propto \int_{\Lambda} f_{\mathrm{s}}\left(\mathbf{x}, \omega_{\mathrm{o}},, \omega_{\mathrm{i}}, \lambda\right) L_{\mathrm{i}}\left(\mathbf{x}, \omega_{\mathrm{i}}, \lambda\right) \left|\cos\theta_{\mathrm{i}}\right| \,\mathrm{d}\lambda. \tag{13}$$

We show the benefits of our method by learning a λ-dependent target distribution. In our experiments, we use the CIE 1931 standard response curve to convert the spectrally rendered results into the sRGB color space for visualization and comparisons. For the non-spectral data (e.g., the RGB reflectance), we up-sample them into smooth spectra at runtime using the technique of [Jakob and Hanika 2019]. We show two spectrally rendered test scenes in Fig. 6. In the Interior scene, we lit a complex indoor scene with colored lights using different SPDs. All windows are equipped with slightly dispersive glass surfaces for light to pass through. In the Liquids scene, we show multiple emissive liquids modeled using homogeneous media with different albedo σ_{a}. The scene is further lit by a ceiling light with spatially varying SPDs. All the illuminants in this scene are enclosed in glass containers with wavelength-dependent IoR, thus preventing both NEE and HeroMIS from being effective. For both scenes, the local target distribution is conditioned on λ, caused by wavelength-dependent lighting and materials. We show our method produces better results for modeling these λ-dependent effects, thus resulting in noticeably less color noise.

Discussion. As shown in the experimental results, applying our method to spectral rendering yields a smaller relative improvement compared to previous guiding methods than in the case of motion blur rendering (see Fig. 5). We believe this could be attributed to the relatively smooth spectral power distribution (SPD) we used for most emitters, as well as the reflectance spectra of the textures (upsampled from RGB in the test scenes). These spectra are typically smooth and energy-conserving [Jakob and Hanika 2019], resulting in a weaker correlation between p_{g} and λ. Moreover, orthogonal spectral MIS methods like HeroMIS [Wilkie et al. 2014] could also weaken the correlation when it being sufficiently effective (the contribution from multiple different wavelengths are simultaneously considered), thereby making our method less beneficial. Consequently, learning a λ-dependent distribution often offers less significant improvements compared to using a marginalized distribution (Eq. 13) as it does in dynamic scenes.

6.2 Evaluations

Alternatives for Modeling Correlation. We also investigate the effectiveness of other alternatives for modeling additional dependencies on the variable $\mathbf{u}$ using neural networks, including (a) directly inputting the conditional variable into the network, and (b) using 4D grid-based representations, as is also discussed in Sec. 4. For these alternatives, we explored different setup on their hyper-parameters and chose the generally well behaved ones. In Fig. 4, we compare our method (Ours-4D) against these two variants (`4D+Hash` and `3D+Aux`, respectively). Specifically, for the `3D+Aux` method, we encode the

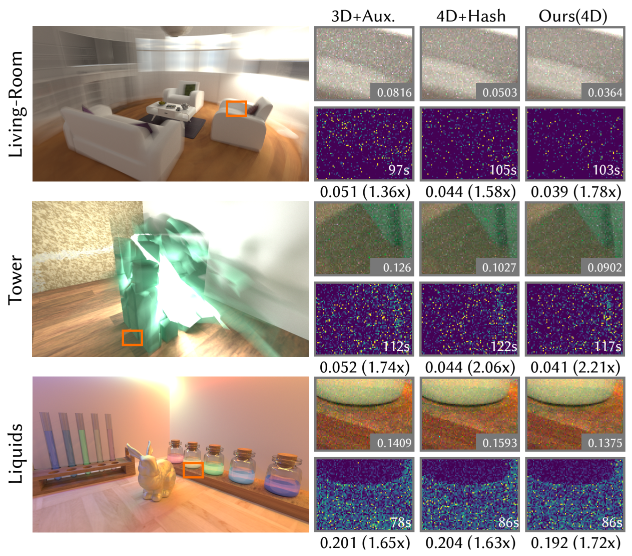

Fig. 4. Comparing different alternatives for modeling 4D correlations on a subset of scenes. The experiments are conducted on an RTX 4080S. We also report the render times and the multipliers of equal-sample error (relMSE) reduction of each method compared to NPM.

auxiliary input $\mathbf{u}$ using frequency encoding [Mildenhall et al. 2020] with 16 frequency levels; while for the `4D+Hash` method, we use the 4D multi-resolution grids with hash tables [Müller et al. 2022] (using the same resolution as our spatial grid but limiting the hash map size to 2^{22}) to avoid the impractical computational and memory cost of a full 4D grid. The parametric feature-grid-based methods generally achieves slightly better reconstruction quality, while has more computation overhead. Therefore, in simple scenarios (e.g., meshes only slightly moves or in spectral rendering scenes), simpler encoding approaches like frequency encoding (`3D+Aux`) might be feasible while maintaining performance efficient.

Performance Analysis. While our method achieves better sampling quality by modeling 4D correlations, it also introduces extra computational overhead. Therefore, we also present the equal-time results of the test scenes in Tab. 1. Generally, we find the two variants of our methods yield similar sampling quality at equal sample rates, while Ours-`4D` often has better efficiency for not using $g_{\mathcal{XU}}$ to model the correlations between $\mathbf{x}$ and $\mathbf{u}$. Both variants of our methods outperform other methods in most scenes. We present the detailed performance breakdown in Tab. 2, where we also show the memory footprint and the inference time of 1280×720 pixels per frame. Our method outperforms NPM at a reasonable memory overhead for modeling the $\mathbf{u}$-dependent guiding distributions. We also evaluate the performance of different resolutions of $g_{\mathcal{XU}}$ (denoted with `mini` and `large`).

Rendering Animated Sequences. Lastly, we validate the effectiveness of our 4D representation by rendering an animated scene with varying lighting and geometry, which is a direct application of our method. We pretrain our method (Ours-4D) and NPM for 30s, then render the animated frames without further training. We show the

Table 2. Performance breakdown of NPM and the different variants of our method. The statistics are acquired with an equal-time (97s) comparison performed on the LIVING-ROOM scene on an RTX 4080S.

Method	Encoding size $\mathcal{G}_{\mathcal{X}} + \mathcal{G}_{\mathcal{X}\mathcal{U}}$	Training per step	Inference per frame	RelMSE
NPM	34.9 MB	3.3ms	10.8ms	0.0547
Ours(4D)$^{\text{mini}}$	37.2+**16.4** MB	9.0ms	15.3ms	0.0433
Ours(4D)$^{\text{base}}$	37.2+**30.2** MB	9.3ms	15.4ms	0.0422
Ours(4D)$^{\text{large}}$	37.2+**48.5** MB	9.3ms	15.9ms	0.0441
`3D+Aux.`	34.9 MB	5.6ms	16.6ms	0.0493
`4D+Hash`	89.8 MB	9.7ms	19.2ms	0.0538
Ours(Prog.)	37.2 MB	5.1ms	11.0ms	0.0366

results of two random frames in Fig. 9. Our method reduces the noise for modeling time-dependent target distribution, while NPM learns an averaged distribution over time, leading to inaccuracies with respect to specific time points.

6.3 Discussion and Limitations

Alternative Solutions. In this work, we augment path guiding methods by modeling the local target distribution with additional integral domains, which benefits the distributed ray-tracing applications and is often neglected by previous work. However, there also exist other methods that facilitate rendering distributed effects, which are orthogonal to our method. For example, primary sample space (PSS) techniques [Crespo et al. 2021; Guo et al. 2018] can often be directly extended to model the extra domains. However, these methods often suffer from the curse of dimensionality, while having difficulty in scaling to arbitrary path lengths [Zheng and Zwicker 2019]. Another direction is to adaptively importance sample the extra domains [Hachisuka et al. 2008; West et al. 2020], which could be combined with our method directly, as also discussed in Sec. 4.2.

It is also possible to extend explicit subdivision data structures to model extra dimensionality. For example, PPG [Müller et al. 2017] can be expanded by nesting an extra binary tree over the directional trees, or by replacing the 3D spatial tree with a 4D K-D tree. However, our pilot test revealed that both designs result in a drastically increased demand of training budget, as far fewer samples fall into the subdivisions. This renders them less effective for the current task setup. Further efforts are needed to address the sparsity of training samples and the curse of dimensionality in explicit 4D structures.

Limitations. Our method achieves better sampling quality by modeling one extra domain for target distribution learning while with a notable computational overhead. However, since the performance gain of our method partially depends on how strongly the target distribution is correlated to the variates $\mathbf{u}$ from the extra domains $\mathcal{U}$, our method might fall short of previous guiding methods, or simpler parametric encoding methods (like frequency encoding) in relatively simple scenes (e.g., motion blur within a very short time period, and/or with few geometry/lighting changes). Thereby, we believe the best use of our method is to augment the path construction in non-trivial scenes featuring intricate distributed effects.

Another limitation resides in the scalability of the number of extra domains. Although our method theoretically applies to a higher number (> 1) of dimensions (discussed in Sec. 4.1), we experimentally found that the performance benefit hardly justifies its additional computational cost in most cases. Unfortunately, this makes our method best used for guiding individual distributed effects (e.g., either motion blur or spectral rendering). We leave the practical simultaneous guiding of multiple distributed effects for future work.

7 CONCLUSION AND FUTURE WORK

In this work, we extend previous local path guiding techniques based on neural mixture fields [Dong et al. 2023; Huang et al. 2024], and investigate the benefits of modeling more accurate target distributions that are conditioned on additional integral domains. We introduce several 4D modeling approaches specifically designed to condition the target distribution on additional integral domains, thereby enhancing path guiding algorithms for distributed ray tracing applications. Our focus is on exploiting the continuity and compactness of coordinate-based neural representations, as well as the mechanisms of high-dimensional representation and the strategy of lower-dimensional neural decomposition. We conduct experiments and explorations, while showing that our method can benefit those typical applications of distributed ray-tracing rendering. We conjecture that our approaches could also be applied to other types of importance sampling networks (e.g., normalizing-flow-based neural importance sampling [Müller et al. 2019]) for potentially better expressiveness, and we leave this as a future work.

In future work, we would like to investigate practical means to scale our method to multiple additional domains, which could benefit complex cases that simultaneously feature multiple distributed effects. Another interesting direction could be combining our method with adaptive sampling techniques in the extra dimensions (e.g., the adaptive stratified sample pattern used in West et al. [2020]). Last but not least, we note that our current study is limited to effectively conditioning the target distribution on 4D domains to facilitate neural path guiding techniques. Developing a practical learning and adaptive sampling method for general 4D distributions remains a more challenging and anticipated area of research, with the potential to benefit a broader range of applications.

ACKNOWLEDGMENTS

This project was supported by the National Key R&D Program of China (No.2023YFF0905103), NSFC of China (No. 62172013), and the Foundation of Shuanghu laboratory (Grant No.SH-2024JK03). The authors wanted to thank the creators for the test scenes: Blendswap user Jay-Artist [2012] (LIVING ROOM), Sketchfab user iansed [2018] (INTERIOR), and Benedict Bitterli [2016] (VEACH-EGG).

REFERENCES

Benedikt Bitterli. 2016. Rendering resources. https://benedikt-bitterli.me/resources/.
Ang Cao and Justin Johnson. 2023. Hexplane: A fast representation for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 130–141.
Robert L. Cook, Thomas Porter, and Loren Carpenter. 1984. Distributed ray tracing *(SIGGRAPH '84)*. Association for Computing Machinery, New York, NY, USA, 137–145.

Miguel Crespo, Adrian Jarabo, and Adolfo Muñoz. 2021. Primary-space adaptive control variates using piecewise-polynomial approximations. *ACM Transactions on Graphics (TOG)* 40, 3 (2021), 1–15.

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.

Honghao Dong, Guoping Wang, and Sheng Li. 2023. Neural Parametric Mixtures for Path Guiding. In *ACM SIGGRAPH 2023 Conference Proceedings*. 1–10.

Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. 2023. K-planes: Explicit radiance fields in space, time, and appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12479–12488.

Jerry Guo, Pablo Bauszat, Jacco Bikker, and Elmar Eisemann. 2018. Primary Sample Space Path Guiding. In *Eurographics Symposium on Rendering - EI & I*, Wenzel Jakob and Toshiya Hachisuka (Eds.). Eurographics, The Eurographics Association, 73–82. http://graphics.tudelft.nl/Publications-new/2018/GBBE18 doi: 10.2312/sre.20181174.

Toshiya Hachisuka, Wojciech Jarosz, Richard Peter Weistroffer, Kevin Dale, Greg Humphreys, Matthias Zwicker, and Henrik Wann Jensen. 2008. Multidimensional adaptive sampling and reconstruction for ray tracing. In *ACM SIGGRAPH 2008 papers*. 1–10.

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

Jiawei Huang, Akito Iizuka, Hajime Tanaka, Taku Komura, and Yoshifumi Kitamura. 2024. Online Neural Path Guiding with Normalized Anisotropic Spherical Gaussians. *ACM Trans. Graph.* 43, 3, Article 26 (apr 2024), 18 pages.

iansed. 2018. The Big Bang Theory Apartment. https://skfb.ly/6wwtG This work is licensed under CC-BY-4.0 (https://creativecommons.org/licenses/by/4.0/).

Wenzel Jakob and Johannes Hanika. 2019. A Low-Dimensional Function Space for Efficient Spectral Upsampling. *Computer Graphics Forum (Proceedings of Eurographics)* 38, 2 (March 2019).

Jay-Artist. 2012. The White Room Cycles. https://blendswap.com/blend/5014 This work is licensed under CC-BY-3.0 (https://creativecommons.org/licenses/by/3.0/).

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

Ondřej Karlík, Martin Šik, Petr Vévoda, Tomáš Skřivan, and Jaroslav Křivánek. 2019. MIS Compensation: Optimizing Sampling Techniques in Multiple Importance Sampling. *ACM Trans. Graph. (SIGGRAPH Asia 2019)* 38, 6 (2019), 12 pages. https://doi.org/10.1145/3355089.3356565

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

Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, et al. 2022. Neural 3d video synthesis from multi-view video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5521–5531.

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

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.

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.

Towaki Takikawa, Thomas Müller, Merlin Nimier-David, Alex Evans, Sanja Fidler, Alec Jacobson, and Alexander Keller. 2023. Compact Neural Graphics Primitives with Learned Hash Probing. In *SIGGRAPH Asia 2023 Conference Papers*. 1–10.

Eric Veach. 1998. *Robust Monte Carlo methods for light transport simulation*. Stanford University.

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.

Rex West, Iliyan Georgiev, Adrien Gruson, and Toshiya Hachisuka. 2020. Continuous Multiple Importance Sampling. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 136–1.

A. Wilkie, S. Nawaz, M. Droske, A. Weidlich, and J. Hanika. 2014. Hero Wavelength Spectral Sampling. *Computer Graphics Forum* 33, 4 (2014), 123–131. https://doi.org/10.1111/cgf.12419

Quan Zheng and Matthias Zwicker. 2019. Learning to Importance Sample in Primary Sample Space. *Computer Graphics Forum* 38, 2 (2019), 169–179. https://doi.org/10.1111/cgf.13628

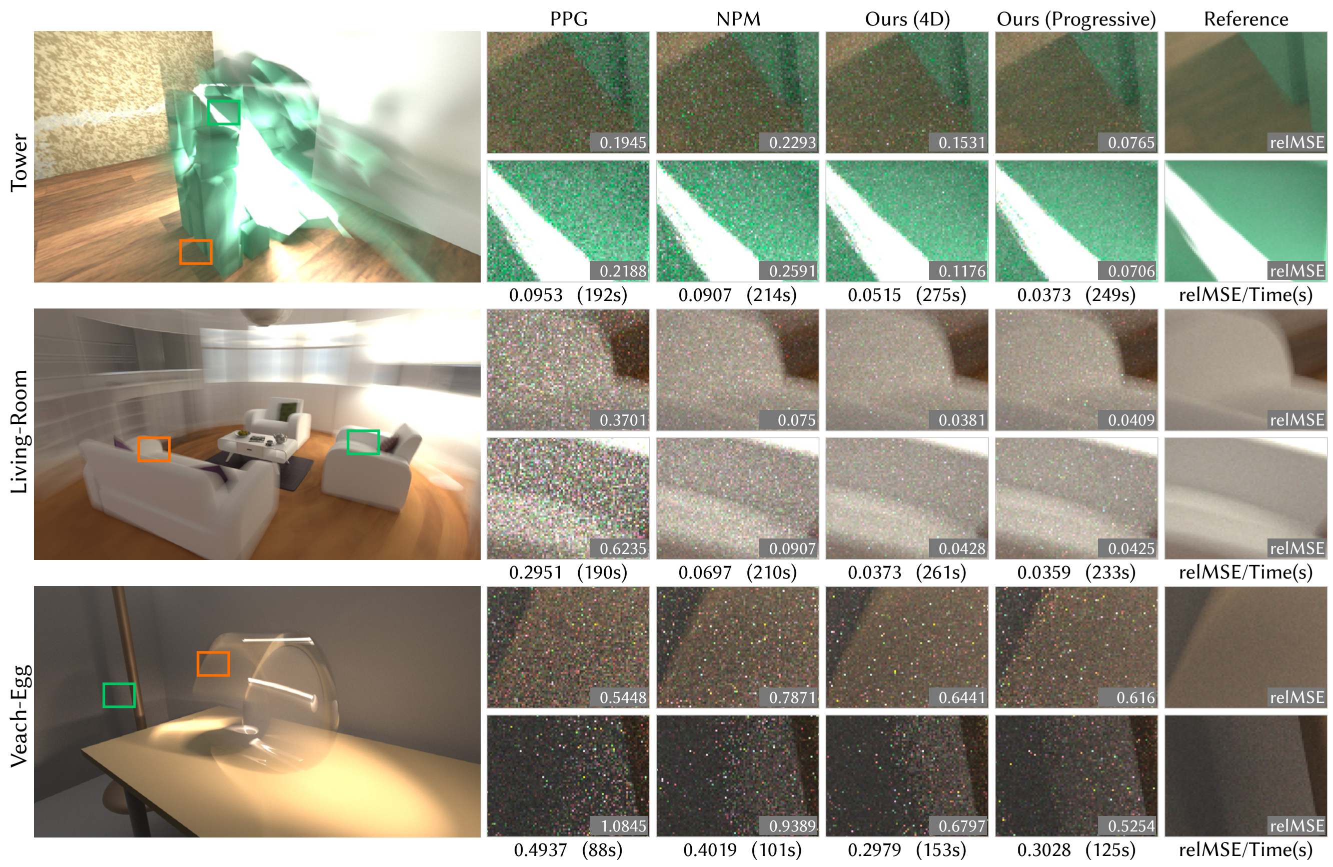

Fig. 5. Visual comparisons of rendering motion blur effects on dynamic test scenes. All images are rendered at 1280 × 720 using 1500 spp, while we also provide the equal-time comparisons in Tab. 1. Details about the scene animation and the rendering process can be inspected in our supplementary video.

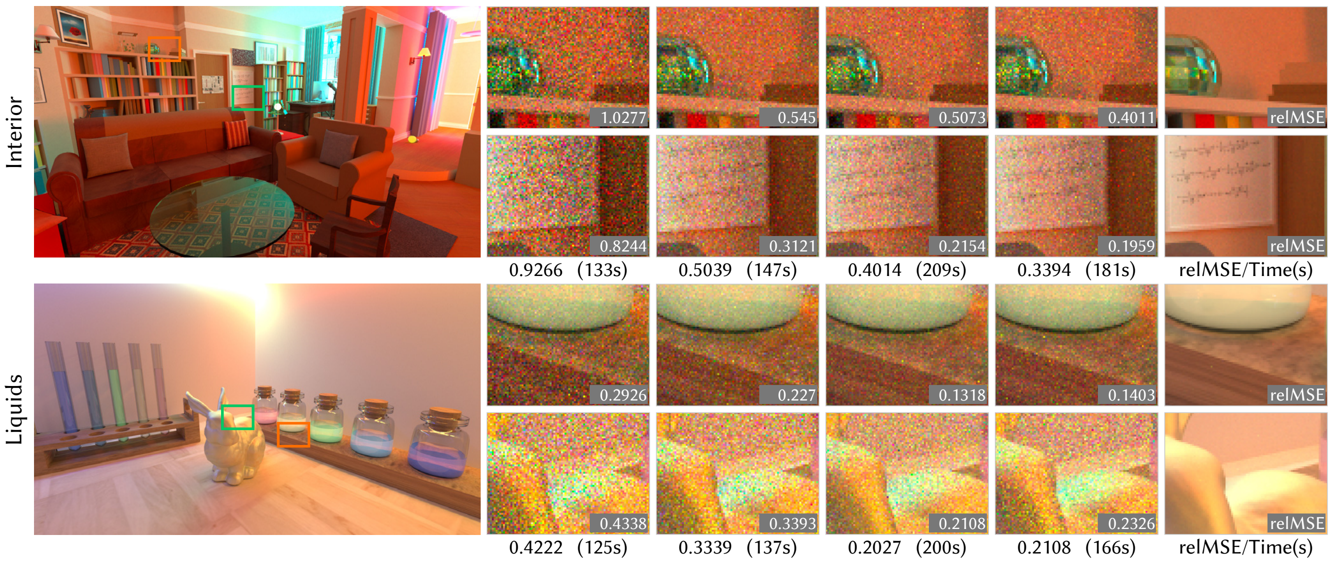

Fig. 6. Visualization of spectral rendering results on two test scenes using the same experimental setup as Fig. 5. Both test scenes contain colored lights with spatially varying SPDs and dispersive dielectric materials. Our method reduces the color noise compared to the alternative guiding methods.

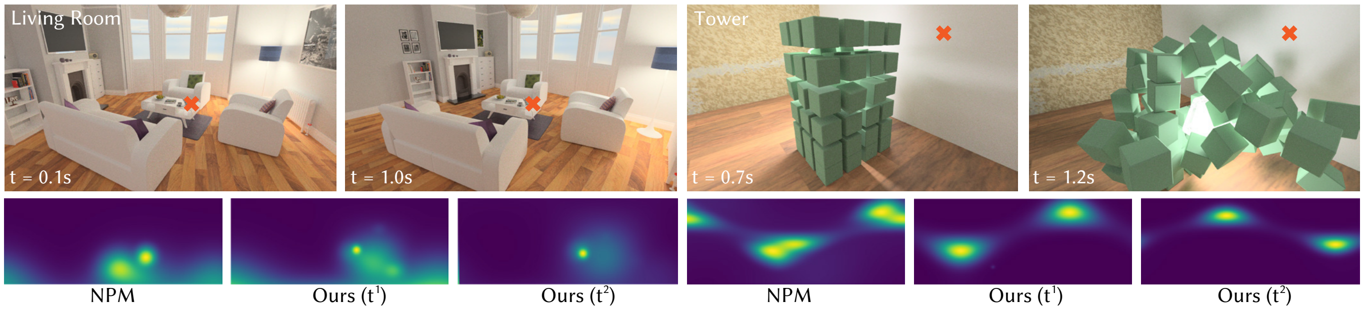

Fig. 7. We visualize the animation of two dynamic scenes, while comparing our time-dependent guiding distribution with previous methods. We find NPM tends to learn a marginalized distribution, which is "blurred" over the footprint of all transient contributing areas, while our method learns a fine-grained distribution that is time-specific. We refer readers to our supplemental video for inspecting detailed scene dynamics and the rendering process.

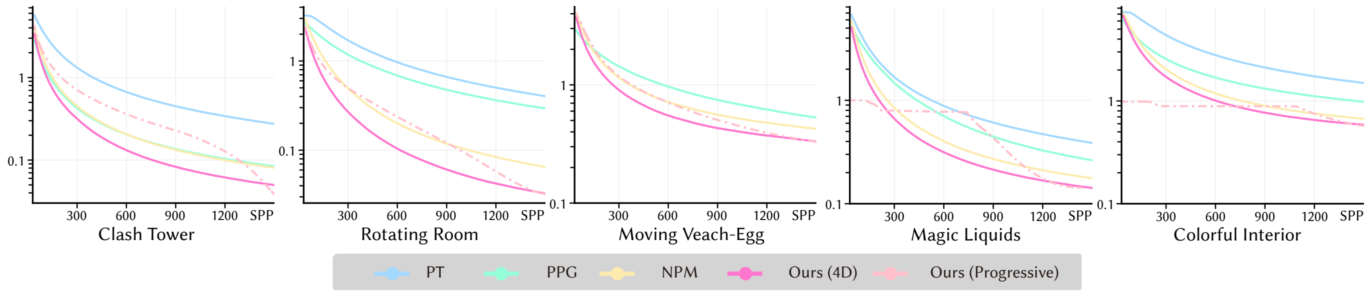

Fig. 8. Convergence plots of path tracing with BSDF importance sampling (PT), practical path guiding (PPG) [Müller et al. 2017], neural parametric mixtures (NPM) [Dong et al. 2023], and the two variants (Ours-4D and Ours-`Progressive`, respectively). Relative MSE is used to visualize the convergence regarding samples per pixel (spp). We use dashed lines to denote the convergence of Ours-`Progressive`, as it uses stratified sampling on the extra dimension, which inherently leads to different convergence plots compared to other methods. Our method consistently outperforms the compared methods on the test scenes.

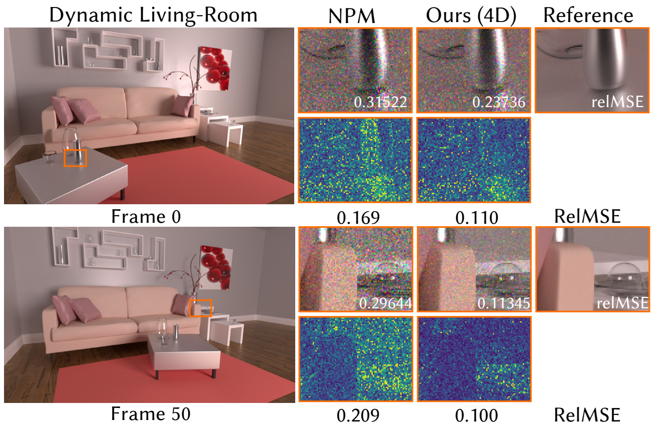

Fig. 9. We pre-train the guiding methods on an animated Pink Room scene with dynamic geometry and lighting, then compare equal-sample-count (300spp) performance of both methods. The results on two random frames are shown. Our method significantly reduces the noise, while using roughly 30% additional frame time on average.