Hierarchical Neural Reconstruction for Path Guiding Using Hybrid Path and Photon Samples

SHILIN ZHU, University of California San Diego, USA
ZEXIANG XU, Adobe Research, USA
TIANCHENG SUN, University of California San Diego, USA
ALEXANDR KUZNETSOV, University of California San Diego, USA
MARK MEYER, Pixar Animation Studios, USA
HENRIK WANN JENSEN, University of California San Diego and Luxion, USA
HAO SU, University of California San Diego, USA
RAVI RAMAMOORTHI, University of California San Diego, USA

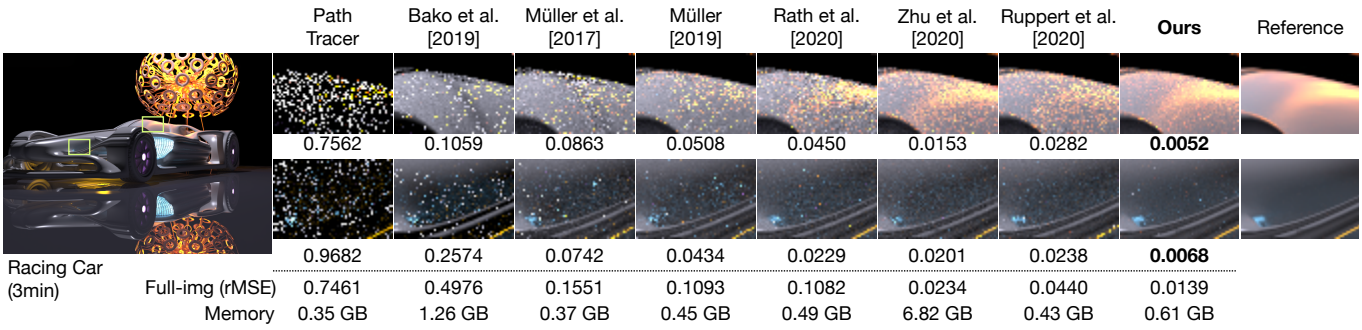

Fig. 1. We present a hierarchical neural path guiding framework which uses both path and photon samples to reconstruct high-quality sampling distributions. This Racing Car scene includes both complex direct and indirect illumination that are difficult for traditional path tracing to render. Traditional guiding methods [Müller et al. 2017; Müller 2019; Rath et al. 2020] can reconstruct hierarchical sampling distributions (quadtrees) via online learning for multi-bounce path guiding. However, the online learning process is relatively slow, which results in noisy sampling maps for a long time, restricting the guiding efficiency. Bako et al. [2019] leverages offline deep learning, but it can only guide the first bounce, which naturally cannot outperform traditional online methods for such a scene with strong global illumination. Ruppert et al. [2020] introduces parallax compensation and uses mixture models (VMMs) to represent sampling distributions. However, the use of analytical mixtures limits the capability to represent complex radiance field from sparse path samples, which requires careful strategies for merging and splitting of mixture components. The recent photon-driven work [Zhu et al. 2020b] can support multiple bounces using an offline-trained network, producing better renderings than many previous methods. However, this method uses standard regular 2D images (unlike quadtrees) for representing lighting distributions, requiring the largest memory consumption, which limits its scalability to large-scale scenes. Our approach enables neural reconstruction of the traditional hierarchical representation via an offline-trained novel network; we can effectively reconstruct accurate quadtree-based sampling distributions, consuming less system memory than [Zhu et al. 2020b]. Our approach also combines both path and photon samples, which is more robust against different light-transport scenarios. As a result, we can achieve better quantitative (reflected by lower rMSE– relative Mean Squared Error) and qualitative results, with moderate memory cost comparable to traditional online methods that do not use deep neural networks.

Path guiding is a promising technique to reduce the variance of path tracing. Although existing online path guiding algorithms can eventually learn good sampling distributions given a large amount of time and samples, the speed of learning becomes a major bottleneck. In this paper, we accelerate the learning of sampling distributions by training a light-weight neural network offline to reconstruct from sparse samples. Uniquely, we design our neural network to directly operate convolutions on a sparse quadtree, which regresses a high-quality hierarchical sampling distribution. Our approach can reconstruct reasonably accurate sampling distributions faster, allowing for efficient path guiding and rendering. In contrast to the recent offline neural path guiding techniques that reconstruct low-resolution 2D images for sampling, our novel hierarchical framework enables more fine-grained directional sampling with less memory usage, effectively advancing the practicality and efficiency of neural path guiding. In addition, we take advantage of hybrid bidirectional samples including both path samples and photons, as we have found this more robust to different light transport scenarios compared to using only one type of sample as in previous work. Experiments on diverse testing scenes demonstrate that our approach often improves rendering results with

Authors' addresses: Shilin Zhu, University of California San Diego, USA, shz338@eng.ucsd.edu; Zexiang Xu, Adobe Research, USA, zexu@adobe.com; Tiancheng Sun, University of California San Diego, USA, tis037@eng.ucsd.edu; Alexandr Kuznetsov, University of California San Diego, USA, a1kuznet@eng.ucsd.edu; Mark Meyer, Pixar Animation Studios, USA, mmeyer@pixar.com; Henrik Wann Jensen, University of California San Diego and Luxion, USA, henrik@cs.ucsd.edu; Hao Su, University of California San Diego, USA, haosu@eng.ucsd.edu; Ravi Ramamoorthi, University of California San Diego, USA, ravir@cs.ucsd.edu.

0730-0301/2021/8-ART35
https://doi.org/10.1145/3450626.3459810

better visual quality and lower errors. Our framework can also provide the proper balance of speed, memory cost, and robustness.

CCS Concepts: • **Computing methodologies** → **Ray tracing**.

Additional Key Words and Phrases: Global Illumination, Path Guiding, Ray Tracing, Sampling and Reconstruction, Neural Rendering

ACM Reference Format:
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 Trans. Graph.* 40, 4, Article 35 (August 2021), 16 pages. https://doi.org/10.1145/3450626.3459810

1 INTRODUCTION

The simple and flexible Monte-Carlo path tracing algorithm has become the gold standard for physically-based rendering. However, a major drawback is the slow convergence problem, leading to unpleasant Monte Carlo noise in the rendered image. In recent years, researchers have successfully tried many denoising and filtering techniques to reduce the noise level [Chaitanya et al. 2017; Bako et al. 2017; Vogels et al. 2018]. However, the denoised image is no longer unbiased, and sometimes has remaining low-frequency artifacts.

Path guiding is a promising direction to reduce path tracing variance while remaining unbiased. The key idea is to learn a better sampling distribution (approximating the incident light field or some variant of it) at arbitrary scene locations and guide camera rays towards the light source. Previous methods [Müller et al. 2017; Rath et al. 2020; Ruppert et al. 2020] often require a slow online learning process to obtain accurate sampling distributions for path guiding. While some recent works [Bako et al. 2019; Zhu et al. 2020b] use offline-trained neural networks, their methods require large system memory and can only reconstruct sampling distributions at a low resolution, restricting the accuracy and efficiency of path guiding.

In this work, we present a novel neural path guiding approach that can effectively reconstruct accurate hierarchical high-resolution sampling distributions, leading to efficient path guiding and rendering. Our approach uses an offline-trained neural network to accelerate the online learning in traditional path guiding. Unlike previous offline neural methods that represent a distribution using a uniform grid (as a 2D image), we consider the classical quad-tree based representation, allowing for efficient high-resolution distribution modeling. As shown in Fig. 1, our approach successfully advances the efficiency of neural path guiding, leading to better rendering quality with moderate memory costs.

We present a novel deep neural network for efficient hierarchical distribution reconstruction. Our technique is inspired by the octree networks [Wang et al. 2017] in 3D geometry processing. We propose to operate deep 2D convolutions directly on a sparse quadtree that represents a 2D angular sampling distribution, enabling an efficient hierarchical reconstruction. Our network can adaptively adjust the tree structure in reconstruction, which learns the proper angular resolution for each sampling solid angle bin. This results in high-quality distributions that accurately express the incident light fields. In contrast to the standard convolutional neural networks (CNNs) that can only regress low-resolution sampling maps [Zhu et al. 2020b], our network hierarchically regresses a compact quadtree that represents the same distribution at a much higher resolution using less memory. The adaptivity and compactness of our hierarchical reconstruction improves the scalability to large-scale complex scenes where a large number of sampling distributions need to be stored on numerous mesh surfaces.

Table 1. Comparison of different path guiding algorithms. Our proposed framework can achieve both fast and robust rendering by leveraging neural networks and hybrid samples with a small memory consumption thanks to the hierarchical representation of sampling distributions.

	Hybrid	Hierarchical	Neural
[Vorba et al. 2014]	✗(Photon)	✗(GMM)	✗
[Müller et al. 2017]	✗(Path)	✓(Quadtree)	✗
[Müller 2019]	✗(Path)	✓(Quadtree)	✗
[Bako et al. 2019]	✗(Path, 1st bounce)	✗(Image)	✓
[Rath et al. 2020]	✗(Path)	✓(Quadtree)	✗
[Zhu et al. 2020b]	✗(Photon)	✗(Image)	✓
Ours	✓(Path + Photon)	✓(Quadtree)	✓

Previous path guiding work uses either path samples [Müller et al. 2017; Müller 2019; Rath et al. 2020; Ruppert et al. 2020] or photons [Jensen 1995; Vorba et al. 2014, 2019] to reconstruct an incident radiance field which is then converted to a sampling distribution at any scene location. Our hierarchical neural reconstruction can potentially support either input samples independently. However, path samples and photons can perform differently depending on the actual light transport cases (see extreme examples in Fig. 3). When the scene contains caustics produced by transparent objects or tiny light sources, photons are more efficient since it is difficult for path samples to quickly find a valid direction towards the light. On the other hand, path samples are a better choice when some light sources do not illuminate the visible regions of the scene, since many photons can be invisible and useless in this case. In this work, we use both of them and let the neural network figure out how to effectively combine the hybrid samples into a single output sampling distribution. Therefore, our approach is more robust to general scenes with unknown light transport scenarios.

In summary, our main contributions are:

- We propose a novel learning-based framework that can reconstruct a *hierarchical* sampling distribution from sparse samples with a moderate memory cost;
- We consider *hybrid* input samples including both path samples and photons for path guiding, leading to higher robustness and generality on diverse light transport cases.

2 RELATED WORK

Path Guiding. Monte-Carlo path tracing [Kajiya 1986] has been the fundamental solution for solving the light transport in a complex scene. However, during path tracing, the incident light distribution is unknown at each 3D point. Thus, most of the path tracing variants sample the space only based on the geometry and reflectance properties. Instead, path guiding algorithms [Vorba et al. 2019] estimate sampling distributions based on the local incoming light field during path tracing, so that they can use the information to perform better importance sampling and accelerate the rendering process.

Several path guiding algorithms (Table 1) have been proposed to efficiently estimate the local light field information in order to better sample the space. Vorba et al. [2014] fitted a Gaussian-Mixture model (GMM) to represent the incoming radiance at each spatial cache point during ray tracing. With very few parameters, a GMM can efficiently model the light distribution, and is then applied to other rendering algorithms [Herholz et al. 2016; Ruppert et al. 2020]. However, GMMs fail to accurately represent high-frequency light distributions, which are common in scenes with complex lighting. Müller et al. [2017, 2019] proposed to use hierarchical quadtree structures to record the incoming light field in the space, which is more efficient and practical than a GMM [Vorba et al. 2014] or simple regular grid [Jensen 1995]. This hierarchical representation was also extended to primary space [Guo et al. 2018], product sampling [Diolatzis et al. 2020], and variance-aware importance sampling [Rath et al. 2020]. However, until now, such a hierarchical representation can only be reconstructed via traditional online learning without any neural network components, and requires relatively large number of samples. Our neural approach can directly reconstruct an accurate hierarchical quadtree representation from sparse input samples, using an offline-trained novel deep neural network.

Recently, deep learning techniques have been used to facilitate the learning of local light distributions and importance sampling of light paths (e.g., in primary sample space [Zheng and Zwicker 2019]). Müller et al. [2019] used an online-learnt neural network to perform the importance sampling. The network can estimate the distribution accurately, but can be potentially expensive in practice due to the repeated network inference and online optimization. Bako et al. [2019] trained an offline-learnt network to guide the first bounce, where regular images are used to represent the incoming light distribution. While images are convenient for neural networks, they consume more memory when detailed light distributions are needed. Huo et al. [2020] used a reinforcement learning technique to guide the samples, but their method is also limited to the first bounce. Zhu et al. [2020b] used photons as the primary source to estimate the local light distributions, and use them to guide all bounces. Again, standard images are used to represent the distributions, which is less memory efficient and limited to low resolutions compared to the quadtree. In this paper, we learn the light distribution on hierarchical structures, which are both detailed and memory-efficient. Our approach takes advantage of using both path and photon samples, leading to better generality on different scenes. We believe these are important steps to make neural path guiding practical.

Hierarchical Learning. Hierarchical structures can represent sparse data in an efficient way [Müller et al. 2017; Müller 2019]. However, learning on the hierarchical structures has been a particular challenge. Recently, there have emerged plenty of studies that focus on the learning and understanding on hierarchical structures, especially in the 3D geometry processing community. Wang et al. [2017, 2018] proposed O-CNN to analyze 3D shapes represented by octrees; Graham [2015] developed sparse convolution for 3D understanding, which is similar to sparse matrix representation. On the other hand, there are also works on generating hierarchical structures [Tatarchenko et al. 2017; Chitta et al. 2020; Riegler et al. 2017]. These algorithms were then extended to perform 3D shape completion [Wang et al. 2020], 3D segmentation [Graham and van der Maaten 2017; Graham et al. 2018], and sketch understanding [Kumar Jayaraman et al. 2018]. Besides convolutional operators, multi-layer perceptrons [Li et al. 2017, 2019] and graph networks [Mo et al. 2019] are also used for hierarchical learning. In this work, we extend these hierarchical 3D learning techniques to the problem of 2D sampling distribution reconstruction. We introduce a novel light-weight network that can effectively regress an accurate quadtree distribution for high-quality path-guiding.

Hybrid samples. Both paths and photons are efficient tools to explore the scene and compute the radiance in the 3D space. While path tracing [Kajiya 1986] algorithms are particularly good at exploring complex geometry setups, photon mapping algorithms [Shirley et al. 1995; Jensen 1996; Hachisuka et al. 2008; Knaus and Zwicker 2011; Zhu et al. 2020a] can be very effective when indirect lighting dominates the scene. Aiming at a rendering algorithm that can work on both cases, researchers proposed bidirectional approaches [Lafortune and Willems 1993; Veach and Guibas 1995a; Georgiev et al. 2012; Křivánek et al. 2014], which combine the benefits of both path tracing and photon mapping. Similarly, in our paper we use both path samples and photons as the sources to learn the local light distributions in the scene. Compared to the previous path guiding works that use only path samples [Müller et al. 2017; Bako et al. 2019; Rath et al. 2020] or photons [Vorba et al. 2014; Zhu et al. 2020b], our algorithm can render more efficiently and is more robust across a wide range of difficult scenes with complex light transports. In fact, Vorba et al. [2014] also use both path and photon samples. However, they train with two separate cache records, where path tracing is guided by local photons and path samples do not *directly* affect camera path guiding. Our neural system instead takes the *hybrid* of two types of samples as direct inputs; they directly contribute to the same forward sampling distribution.

As shown in Table 1, our path guiding algorithm uniquely utilizes the hybrid samples. Additionally, previous works either perform learning on image-based sampling distributions, or use hierarchical structures to represent the distributions (because of the difficulty of applying neural networks to irregular quad-tree structures), but not both. In contrast, our path guiding algorithm successfully applies an offline-learnt neural network on hierarchical structures.

3 BACKGROUND

Rendering equation. To render a scene using light transport simulation, our goal is to solve the rendering equation [Kajiya 1986]:

$$L(\boldsymbol{x}, \omega_o) = L_e(\boldsymbol{x}, \omega_o) + \int_\Omega L_i(\boldsymbol{x}, \omega_i) f_r(\boldsymbol{x}, \omega_i, \omega_o) \cos\theta_i d\omega_i, \quad (1)$$

where the outgoing radiance $L(\boldsymbol{x}, \omega_o)$ in direction ω_o at each surface point $\boldsymbol{x}$ equals the sum of the surface emission $L_e(\boldsymbol{x}, \omega_o)$ and the reflection from the incoming radiance $L_i(\boldsymbol{x}, \omega_i)$ of every direction ω_i that has angle θ_i to the surface normal over the hemisphere Ω. The Bidirectional Scattering Distribution Function (BSDF) $f_r(\boldsymbol{x}, \omega_i, \omega_o)$ describes how much radiance can be scattered to ω_o from ω_i.

The integration $L_r(\boldsymbol{x}, \omega_o) = \int_\Omega L_i(\boldsymbol{x}, \omega_i) f_r(\boldsymbol{x}, \omega_i, \omega_o) \cos\theta_i d\omega_i$ in Eqn. 1 is computed by Monte Carlo (MC) estimation [Veach 1997]

in the path tracing algorithm:

$$L_r(\boldsymbol{x}, \omega_o) = \frac{1}{N} \sum_{i=1}^{N} \frac{L_i(\boldsymbol{x}, \omega_i) f_r(\boldsymbol{x}, \omega_i, \omega_o) \cos\theta_i}{p(\omega_i)} \tag{2}$$

where N is number of samples and $p(\omega_i)$ is the probability density function (PDF) of sampling direction ω_i (i.e., importance sampling). When N is sufficiently large, the variance of $L_r(\boldsymbol{x}, \omega_o)$ reduces, and path tracing gradually converges to the noise-free result.

In many challenging light transport scenarios, the convergence is very slow, which is the major drawback of Monte Carlo path tracing. Fortunately, we can greatly speed up the variance reduction by sampling from a better PDF $p(\omega_i)$ that resembles the integrand $L_i(\boldsymbol{x}, \omega_i) f_r(\boldsymbol{x}, \omega_i, \omega_o) \cos\theta_i$. However, the incident radiance field $L_i(\boldsymbol{x}, \omega_i)$ is unknown in the beginning, so standard path tracing only leverages the BSDF for importance sampling:

$$p_{\text{BSDF}}(\omega_i) \propto f_r(\boldsymbol{x}, \omega_i, \omega_o) \tag{3}$$

Guiding using path samples. In contrast, path guiding is a method to evaluate the incident light $L_i(\boldsymbol{x}, \omega_i)$ and set the PDF to be proportional to some terms related to it. Many previous papers [Müller et al. 2017; Müller 2019] use early (or extra) Monte Carlo path samples to compute a sampling distribution as:

$$p_{\text{guide}}(\omega_i) \propto L_i(\boldsymbol{x}, \omega_i) \cos\theta_i, \tag{4}$$

which expresses the incident light field (the cosine term is sometimes associated to the BSDF sampling in Eqn. 3). In practice, this guided sampling is often combined with BSDF sampling, using one-sample Multiple Importance Sampling (MIS) [Veach and Guibas 1995b]:

$$p(\omega_i) = \alpha p_{\text{BSDF}}(\omega_i) + (1-\alpha) p_{\text{guide}}(\omega_i) \tag{5}$$

where the coefficient α determines the chance of selecting BSDF over guiding for importance sampling.

However, since $L_i(\boldsymbol{x}, \omega_i)$ is also from the noisy Monte Carlo samples [Müller et al. 2017], the estimates are also noisy and can have high variance, making the sampling inefficient. Recently, Rath et al.[2020] introduce a variance-aware guiding technique, leveraging a new target sampling function that considers the variance:

$$p_{\text{guide-var}}(\omega_i) \propto \sqrt{\mathbb{E}[L_i^2(\boldsymbol{x}, \omega_i)] \cos^2\theta_i} \tag{6}$$

where $\mathbb{E}[\cdot]$ represents the expectation. Additionally, they also take the surface material into account, resulting in the BSDF marginalized product sampling [Rath et al. 2020] used for path guiding:

$$p_{\text{guide-var-prod}}(\omega_i) \propto \sqrt{\mathbb{E}_{\omega_o}[f_r^2(\boldsymbol{x}, \omega_i, \omega_o)] \mathbb{E}[L_i^2(\boldsymbol{x}, \omega_i)] \cos^2\theta_i} \tag{7}$$

Our framework generally supports various sampling functions. We take advantage of the advanced variance-aware technique (Eqn. 7) to generate our results by default, leading to better quality than our results with the traditional distribution (Eqn. 4).

Guiding using photons. In some special light transport cases such as caustics from transparent objects or tiny light sources that are hard to find through Monte Carlo sampling, most path samples are terminated before reaching any light, leading to more noisy $L_i(\boldsymbol{x}, \omega_i)$ estimation. Compared to path samples, photons are often a better choice in these scenarios, which have been used for path guiding by previous work [Jensen 1995; Vorba et al. 2014; Zhu et al. 2020b]. Each photon p carries a small portion of the emitter power (radiant flux) $\Delta\Phi_p$ and its direction ω_p indicates where the light comes from. The power $\Phi(\boldsymbol{x}, \Delta\Omega)$ that flows through a solid angle footprint $\Delta\Omega$ in local surface area A is computed via integrating the incident radiance $L_i(\boldsymbol{x}, \omega_i)$ where $\omega_i \in \Delta\Omega$:

$$\Phi(\boldsymbol{x}, \Delta\Omega) = \int_A \int_{\Delta\Omega} L_i(\boldsymbol{x}, \omega_i) \cos\theta_i d\omega_i d\boldsymbol{x}. \tag{8}$$

The target distribution can be expressed as [Zhu et al. 2020b]:

$$p_{\text{guide-photon}}(\omega_i) \propto \Phi(\boldsymbol{x}, \Delta\Omega)/\Delta\Omega = \sum_{\omega_p \in \Delta\Omega, \boldsymbol{x} \in A} \Delta\Phi_p/\Delta\Omega \tag{9}$$

This sampling distribution similarly approximates Eqn 4, but it is evaluated by the summation of the surrounding photon power, instead of the Monte Carlo estimation of path samples.

In practice, it is hard to know whether a path sample or photon is better for an unknown scene; two extreme examples are shown in Fig. 3. Therefore, in this paper, we choose to use both of them (i.e., hybrid samples), although our framework also directly applies to a single type of sample. Combining path samples and photons is challenging since they distribute very differently and there is no obvious and cheap way to combine them through re-weighting (VCM [Georgiev et al. 2012] and its concurrent work [Hachisuka et al. 2012] design specific techniques to address a similar issue in radiance estimation, which however cannot be easily extended to distribution estimation). Therefore, we use a neural network that learns to combine their values and reconstructs a single sampling distribution (Sec. 6) that is then used for path guiding.

4 OVERVIEW

The entire framework is illustrated in Fig. 2. We trace both photon and path samples (i.e., hybrid samples), and deposit them into a local quadtree representation of the sampling distribution stored in a local spatial caching node as shown in Fig. 2(a) and (b). This step is similar to the online quadtree construction in [Müller et al. 2017]; it leads to noisy distributions unless a large number of samples are deposited. We instead propose to use a deep neural network (pre-trained) to hierarchically reconstruct accurate quadtree distributions from the noisy ones in Fig. 2(c). In the following sections, we first describe the steps of building an initial quadtree at arbitrary scene locations and depositing hybrid samples into it (Sec. 5). Next, Sec. 6 presents the key component of our framework: a novel neural network to reconstruct high-quality hierarchical sampling distributions using both the initial noisy path and photon distributions as input. Finally, we discuss the details of adaptively caching the reconstructed distributions at different locations in the scene and rendering of the final image (Sec. 7). Thereafter, Sec. 8 provides the implementation details of neural training, sample tracing, and rendering. Experiments on diverse testing scenes in Sec. 9 justify the effectiveness of our proposed framework.

5 HIERARCHICAL STRUCTURE FOR HYBRID SAMPLES

As we discussed in Sec. 3, we need to collect path samples and/or photons to learn a directional sampling distribution that resembles the incident radiance field at arbitrary scene locations. Compared to the previous neural path guiding work [Bako et al. 2019; Zhu et al.

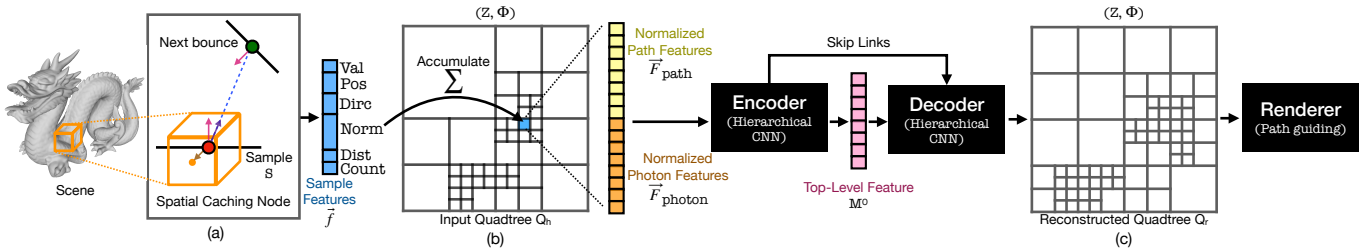

Fig. 2. High-level illustration of the proposed neural path guiding framework. The scene is partitioned into many spatial caching nodes (voxels). Each voxel collects all the samples that arrive at it (a) and uses the sample information to adatpively construct a quadtree Q_h (b), parameterized in the cylindrical coordinates ω_i to (z, ϕ). Each sample contains its value along with some auxiliary features including the relative position $\vec{p}$, direction ω_i, distance $\vec{d}$, normal $\vec{n}$, and sample count $c = 1$, which leads to a feature vector $\vec{f}$ that will be accumulated into a leaf l of the quadtree Q_h (Sec. 6.2). The accumulation is applied separately to path samples and photons, resulting in two independent feature vectors $\vec{\mathbb{F}}_{\text{path}}$ and $\vec{\mathbb{F}}_{\text{photon}}$ (Eqn. 11). We propose a novel neural network that can directly operate convolutions on quadtree distributions (Sec. 6.3), which has an architecture with hierarchical encoder and decoder (Sec. 6.4). We train our network offline that learns to hierarchically regress accurate sampling distributions from noisy inputs. The pre-trained network can reconstruct a high-quality quadtree Q_r (c) from the input Q_h; the reconstructed Q_r is stored in the spatial voxel and later used for path guiding (Sec. 7).

2020b], we hierarchically build a quadtree instead of a uniform 2D grid (image) to represent the distribution. Hybrid samples are traced and stored in the tree, which are later provided to our hierarchical neural network for sampling distribution reconstruction (Sec. 6).

Our quadtree-based distributions are stored in small spatial caching nodes distributed within the scene, as shown in Fig. 2(a). Later in Sec. 7, we discuss the details of adaptively partitioning the scene space into local regions of different sizes for efficient spatial caching. We keep two quadtrees in each spatial node: one records the online traced hybrid samples, representing a noisy distribution and used as network input; the other is the output of the network, representing an accurate sampling distribution for path guiding. The initial noisy quadtree collects local samples that arrive at the node, containing rich information of the local incident radiance field.

Quadtree representation. We use the 2D cylindrical coordinates to parameterize the angular space; each unit vector (x, y, z) is mapped to (z, ϕ), where $\phi = \arctan(y/x)$. A quadtree $\mathbb{Q}_h$ is built to hierarchically cover the space of $(z \in \mathbb{Z}, \phi \in \Phi)$ at each spatial node, recording the hybrid samples traced at rendering time (Fig. 2(b)).

Accumulating hybrid samples. Once a sample $\mathbb{S}$ (either path or photon), carrying a sample quantity $V_{\mathbb{S}}$, arrives at a particular spatial node, we convert its incident direction $\omega_i = (x, y, z)$ to the cylindrical space mentioned above, and deposit it to a corresponding leaf node l of the quadtree $\mathbb{Q}_h$. In particular, we leverage a stochastic box filter [Müller 2019], which deposits the sample value $V_{\mathbb{S}}$ into a single neighboring tree leaf l around its original direction ω_i; this is equivalent to splatting the sample with a box filter into the quadtree.

Since path samples and photons have different radiometric units (Sec. 3), we keep two separate accumulators $\mathbb{A}^l_{\text{path}}$ and $\mathbb{A}^l_{\text{photon}}$:

$$\begin{aligned} \mathbb{A}^l_{\text{path}} &= \sum V^l_{\mathbb{S}_{\text{path}}} \\ \mathbb{A}^l_{\text{photon}} &= \sum V^l_{\mathbb{S}_{\text{photon}}} \end{aligned} \tag{10}$$

where $V^l_{\mathbb{S}_{\text{path}}}$ and $V^l_{\mathbb{S}_{\text{photon}}}$ are splatted sample quantities in leaf l.

Quadtree subdivision. Initially, the tree $\mathbb{Q}_h$ has a single node. To effectively construct $\mathbb{Q}_h$ as a hierarchical structure, we iteratively trace samples (Sec. 7) and subdivide the tree accordingly. Specifically, $\mathbb{Q}_h$ is adaptively refined after the samples in the current iteration are deposited based on a criterion [Müller et al. 2017]: if a node value $\mathbb{A}^l_{\text{path}}$ or $\mathbb{A}^l_{\text{photon}}$ is greater than $k\%$ (we empirically find that $0.5\% \sim 1\%$ is a reasonable threshold) of its total value ($\sum_l \mathbb{A}^l_{\text{path}}$ or $\sum_l \mathbb{A}^l_{\text{photon}}$) in $\mathbb{Q}_h$, the node is split into four equal-sized child nodes where each of them is assigned $1/4$ of the parent value, otherwise it remains as a leaf node. This criterion is applied recursively to each node in the tree. After $\mathbb{Q}_h$ is updated, it is used to collect future samples in the next iteration, so that $\mathbb{Q}_h$ can be repeatedly refined to better versions. This strategy allows $\mathbb{Q}_h$ to have higher directional resolution when the radiance of an incident direction is large. Note that if only path samples are considered (as in previous work [Müller et al. 2017]), then only $\mathbb{A}^l_{\text{path}}$ is used to build and refine $\mathbb{Q}_h$.

This iteratively-refined quadtree $\mathbb{Q}_h$ can in fact model an accurate sampling distribution when the number of accumulated samples is large enough. However, this requires a large number of iterations and a long time for accumulation, which cannot promptly provide reliable sampling distributions. Especially at the beginning of rendering, the accumulated sampling quadtrees are highly noisy and inadequate for path guiding. In Sec. 6, we design a novel neural network to handle the hybrid input samples stored in each leaf l.

6 NEURAL RECONSTRUCTION OF SAMPLING DISTRIBUTIONS

In this section, we introduce our novel hierarchical neural network that can effectively convert the deposited hybrid samples (Sec. 5) to a high-quality sampling distribution for path guiding. We first discuss the motivation of applying neural networks in the context of sampling distributions (Sec. 6.1). Next, we present our network input (Sec. 6.2), the convolutional module applied on a quadtree (Sec. 6.3), and the detailed neural architecture (Sec. 6.4). Finally, we introduce our loss function to train the neural network (Sec. 6.5).

6.1 Motivation of neural reconstruction framework

As discussed in Sec. 5, directly reconstructing an accurate quadtree distribution $\mathbb{Q}_h$ via online accumulation usually requires a long time

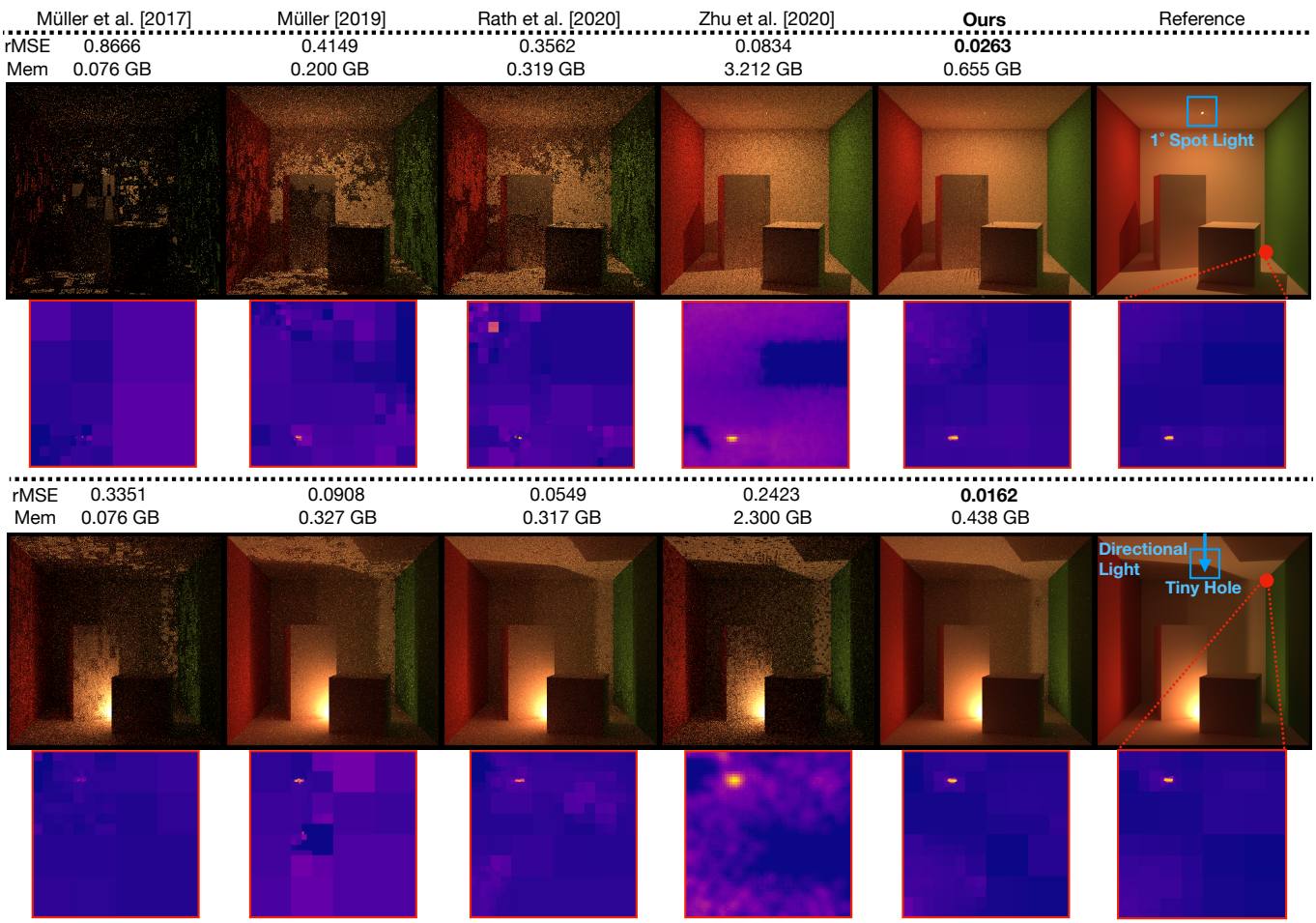

Fig. 3. Extreme conditions. We compare our method with previous path guiding methods, running with equal time, on two Cornell Box scenes that have two different extreme light transport settings. We turn on the next event estimation for all methods in this experiment. Previous methods utilize either path samples [Müller et al. 2017; Müller 2019; Rath et al. 2020] or photons [Zhu et al. 2020b] as input for path guiding, which cannot work well on both cases at the same time. In the first row, the scene is illuminated by a very small 1° *spotlight* (facing upwards) located very close to the roof. This setting is extremely hard for path-based methods [Müller et al. 2017; Müller 2019; Rath et al. 2020] since the light is hard to connect to; yet, the photon-based method [Zhu et al. 2020b] still works well. On the other hand, the second row shows a scene illuminated by a *directional light* coming from the top, while the roof only has a very tiny hole that can receive this light. While path-based methods can still be effective for this setting, the photon-based method [Zhu et al. 2020b] cannot work well (even empowered by deep learning) since most photons will be blocked by the roof and not useful at all. Our novel neural approach leverages both path and photon samples as input, and can successfully work on both challenging cases. We also show the corresponding sampling distributions reconstructed by all methods. Note that these methods may have different ground-truth target sampling distributions (see Sec. 3). We only show our ground truth (the target of [Rath et al. 2020]) as the reference. While the target sampling functions are different, we can still observe our neurally reconstructed quadtrees are of higher quality than the noisy quadtrees reconstructed traditionally by [Müller et al. 2017; Müller 2019; Rath et al. 2020]; ours also contain sharper details than the regular image representation of [Zhu et al. 2020b]. Our quadtrees are reasonably accurate compared to the reference.

to trace a large number of samples, leading to low quality of sampling at early rendering times (as appears in previous work [Jensen 1995; Müller et al. 2017]). We therefore seek to directly reconstruct an accurate quadtree distribution from the initial noisy quadtree; this can be seen as a traditional image reconstruction task (like denoising, inpainting, or restoration) in the $(\mathbb{Z}, \Phi)$ space, except that now the task is applied on hierarchical trees instead of regular 2D images. Therefore, the standard CNN on a 2D grid image (e.g., [Bako et al. 2019; Zhu et al. 2020b]) is no longer applicable, and we aim to design a new neural architecture that extends CNNs to hierarchical inputs and outputs. Meanwhile, prior works have been addressing a similar task in 3D geometry processing. They apply CNNs on octrees [Wang et al. 2017, 2018] and hierarchical MLPs on grammar trees [Li et al. 2017, 2019] to achieve highly efficient 3D learning. We extend these 3D learning techniques to reconstruction of sampling distributions and we propose to apply neural convolutional operations on the 2D sampling quadtrees. Note that our neural framework can not only denoise the values of the input $\mathbb{Q}_h$,

Table 2. List of notations used in Sec. 5, Sec. 6, and Sec. 7.

	Notation	Meaning
Hierarchical input structure (Sec. 5)	$\mathbb{S}$	Sample
	l	Quadtree leaf
	ω_i	Sample direction
	$(\mathbb{Z}, \Phi)$	Directional sampling space
	$V_{\mathbb{S}}$	Sample value
	$\mathbb{A}$	Accumulated sample value
	$\mathbb{Q}_h$	input quadtree from online accumulation
Neural network framework (Sec. 6)	$\mathbb{Q}_r$	Reconstructed quadtree from the neural network
	$\mathbb{Q}_{\text{gt}}$	Target (groundtruth) quadtree
	$\vec{f}$	Per-sample feature vector
	$\vec{\mathbb{F}}$	Per-leaf feature vector
	F_{conv}	Convolution result
	$\mathbb{M}$	Per-level feature map
	$\mathbb{V}$	Predicted relative value to the parent node
	p_{leaf}	Predicted probability of node being a leaf
	m, n	Encoding and decoding tree level
	q, q_c	Decoded tree node and one of its children
	$\mathbb{L}_{Q_r}$	Loss function
	$\mathcal{P}$	Pooling
	$\mathcal{S}$	Convolution
	$\mathcal{U}$	Upsampling
	$\mathcal{T}$	Node type classifier
	$\mathcal{R}$	Value regressor
Sampling and rendering (Sec. 7)	r_{init}	Initial grid resolution
	$\mathbb{G}$	Adaptive hierarchical hash grid
	$\mathbb{B}_{\text{spt}}$	KD-tree in each voxel
	k_{spt}	Spatial subdivision threshold
	$t, \mathbb{T}$	Current and total iteration(s)
	α	One-sample MIS coefficient

but also create a completely separate hierarchical structure $\mathbb{Q}_r$ that can be different from $\mathbb{Q}_h$, better representing the target distribution. Our hierarchical neural reconstruction leverages the sparsity of the sampling distribution, processing and modeling directly on quadtrees; this allows for high-resolution modeling using low memory, which is not achievable when using regular images with CNNs. In addition, we also design our network to be compact enough for high computational and memory efficiency; this is ideal for path guiding, since it needs to simultaneously reconstruct sampling distributions at many different scene locations without introducing too much overhead to the rendering algorithm.

6.2 Input hybrid samples

As described in Sec. 5, when a new path sample or photon arrives, we convert its ω_i into (z, ϕ) and search $\mathbb{Q}_h$ to find its corresponding leaf node. Two separate value accumulators ($\mathbb{A}_{\text{path}}$ and $\mathbb{A}_{\text{photon}}$ in Eqn. 10) are used for adaptively refining $\mathbb{Q}_h$. However, using only one-channel sample values is insufficient for reconstructing a better quadtree. In this work, we collect additional auxiliary per-sample information and form a hybrid multi-channel feature vector, as illustrated in Fig. 2(a). Specifically, each sample contains value $\mathbb{V}$ and additional features which include the local sample position $\vec{p}$, the sample direction ω_i, the distance d to the next bounce, the surface normal directions $\vec{n}$ of the current and next bounce, and the sample count $c = 1$. For path samples, we also append the BSDF value f_r to the vector. Finally, as shown in Fig. 2(b), sample features $\vec{f}_{\text{path}}$ and $\vec{f}_{\text{photon}}$ are accumulated on each leaf l at tree level m, and then concatenated into a single feature vector $\vec{\mathbb{F}}^{m,l}$:

$$\begin{aligned}
&\vec{f}_{\text{path}} = (\mathbb{V}, \omega_i, \vec{p}, d, \vec{n}, c, f_r)\ \vec{f}_{\text{photon}} = (\mathbb{V}, \omega_i, \vec{p}, d, \vec{n}, c) \\
&\vec{\mathbb{F}}^{\text{acc}} = (\sum \vec{f}_{\text{path}}, \sum \vec{f}_{\text{photon}}) \\
&\vec{\mathbb{F}}^{m,l} = (\frac{\vec{\mathbb{F}}^{\text{acc}}_{\text{path}}}{\max\limits_{Q_h} \vec{\mathbb{F}}^{\text{acc}}_{\text{path}}}, \frac{\vec{\mathbb{F}}^{\text{acc}}_{\text{photon}}}{\max\limits_{Q_h} \vec{\mathbb{F}}^{\text{acc}}_{\text{photon}}})
\end{aligned} \tag{11}$$

where $(,)$ means vector concatenation, and summations are computed for each leaf. $\max\limits_{Q_h} \vec{\mathbb{F}}^{\text{acc}}$ is the feature-wise maximum value within the entire quadtree $\mathbb{Q}_h$ after the summation, which is used for separately normalizing the input of path samples and photons. This normalization effectively removes the radiometric unit difference between path samples and photons. The sample direction ω_i is also implicitly included in the (z, ϕ) coordinates of $\mathbb{S}$.

6.3 Convolution on a quadtree

We propose to directly apply convolutions on the quadtree to process and regress the hierarchical feature data. In general, given a leaf l on level m in $\mathbb{Q}_h$, a convolutional layer outputs a new feature $F^{m,l}_{\text{conv}}$ via a linear operation that is applied on its neighbors (empty neighbor nodes are regarded as zeros) on the same tree level m:

$$\begin{aligned}
F^{m,l}_{\text{conv}}[g] &= \sum_c \sum_i \sum_j W_{i,j,c}[g] \cdot \vec{\mathbb{F}}^{m,l}_{i,j,c} \\
\mathbb{M}^m[g][l] &= F^{m,l}_{\text{conv}}[g]
\end{aligned} \tag{12}$$

where i and j are 2D indices of the neighbors inside the convolutional kernel W, c represents the channel index of input features, and g is the index of kernels (also the channel index of the output feature). Here $\mathbb{M}^m$ is a sparse 2D feature map, containing the output features of all valid leaves. Note that, this convolution on a quadtree (Eqn 12) is not much different from the standard convolutional layer on a 2D image. However, each $\vec{\mathbb{F}}^{m,l}_{i,j,c}$ represents a feature in a quadtree leaf node instead of a standard pixel; unlike an image, leaf nodes on a single tree level m can distribute very sparsely, where only a few leaves contain actual features that require convolutions.

Moreover, accessing a neighbor within the convolutional kernel requires searching in the quadtree $\mathbb{Q}_h$ to get its stored features $\vec{\mathbb{F}}^{m,l}$ (Eqn. 11). This is non-trivial and can be much slower than the standard CNN on a regular image where any element in an array is immediately accessible. Fortunately, this neighboring search problem has been addressed by the 3D shape processing community using a faster hash table implementation [Graham et al. 2018; Wang et al. 2017, 2018] with an optimization on reducing hash table lookup times. In this work, we apply the same technique to speed up our CNN on quadtrees, enabling efficient quadtree convolutional operations. The same neighboring search is also naturally applied to pooling layers in our network. Note that, because of the sparsity of a quadtree, the network layers are applied only to the sparse nodes in each tree level m, which actually reduces the amount of computation compared to the standard dense CNNs.

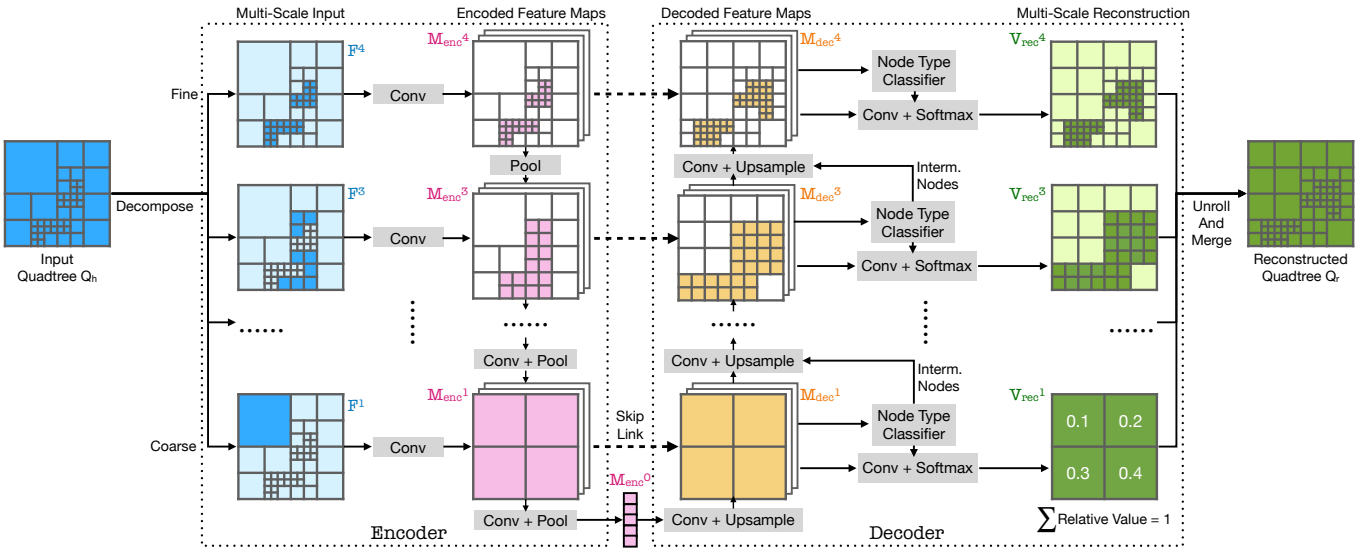

Fig. 4. Our proposed hierarchical encoder-decoder architecture for reconstructing an accurate quadtree representation of sampling distributions. Here, we show an example containing only 4 levels. In practice, the input and output tree can have different levels ranging from 1 to 20. First, on each level m of the noisy input quadtree Q_h, we apply a series of convolutional and pooling layers to encode the sample features $\mathbb{F}^m$ to a neural feature map $\mathbb{M}^m_{\text{enc}}$. By repeatedly applying these operations hierarchically from the bottom level to the root node, we eventually encode and compress the whole Q_h into a single feature vector $\mathbb{M}^0_{\text{enc}}$. The decoder can be seen as the reverse of the encoder, which includes a series of convolutional and upsampling layers to extract new features $\mathbb{M}^n_{\text{dec}}$ and reconstruct a new quadtree Q_r that has new tree structure and values. On each decoding level n, we use convolutions followed by a SoftMax operation to regress a relative value $\mathbb{V}^{n,q}_{\text{rec}}$ for each node q with respect to its parent node value (therefore the summation of every four child nodes satisfy $\sum_q \mathbb{V}^{n,q}_{\text{rec}} = 1$). Meanwhile, a MLP classifier $\mathcal{T}_n$ predicts the type of each decoded node on that level, and sends all the intermediate nodes into the $(n+1)$-th level for further processing. Finally, the predicted values $\mathbb{V}^{n,q}_{\text{rec}}$ are converted and merged into the output quadtree Q_r. Note that encoding and decoding operations are applied only to the sparse nodes on each level, which is more computationally efficient compared to the standard CNNs that operate on dense image grids.

6.4 Hierarchical architecture

Our proposed neural architecture (Fig. 4) contains a hierarchical encoder and decoder, with the skip links in between (Figure 2(c)). Each hierarchical processing layer represents a corresponding tree level in the input $\mathbb{Q}_h$ or the output $\mathbb{Q}_r$.

Neural hierarchical quadtree encoder. We take $\mathbb{Q}_h$ as the input and process the leaves from the bottom (finest) level to the top level. On each level m, we apply a series $\mathcal{S}_m$ of convolutions (Eqn. 12) and nonlinear ReLU activation functions on the accumulated feature vectors $\vec{\mathbb{F}}^{m,l}$. The output feature map $\mathbb{M}^m$ (Eqn. 12) is downsampled to the $(m-1)$-th quadtree level after the 2×2 average pooling $\mathcal{P}_m$, and is then fused with the feature map $\mathbb{M}^{m-1}$ at the $(m-1)$-th level. This iterative encoding can be expressed as:

$$\mathbb{M}^{m-1}_{\text{enc}} = (\mathcal{P}_m(\mathbb{M}^m), \mathbb{M}^{m-1}) \tag{13}$$

where $\mathbb{M}^m_{\text{enc}}$ is the fused feature at level m. In summary, we start with the bottom tree level $m_{\max}$ in $\mathbb{Q}_h$ and combine the features from every coarser level until reaching the 0-th level (tree root).

Neural hierarchical quadtree decoder. Our goal is to reconstruct a tree $\mathbb{Q}_r$, which can better represent the target sampling distribution from hybrid sample inputs. To do so, we design our decoder to not only regress the output distribution values at each tree level but also determine if every node needs to be a leaf node or requires further subdivision. This allows the decoder to simultaneously build a new tree structure and reconstruct (denoise) leaf values.

The entire decoder can be seen as an inverse process of the encoder. After the multi-scale features $\mathbb{M}^m_{\text{enc}}$ are hierarchically extracted from $\mathbb{Q}_h$ via the encoder (Eqn. 13), we apply a series $\mathcal{S}_n$ of convolutions and ReLU activations to compute the feature $\mathbb{M}^n_{\text{dec}}$ at each decoding level n ($n = m = 0$ is the tree root). A 2×2 upsampling layer $\mathcal{U}_n$ on level n is also applied, subdividing a node into four equal-sized children, which reverses the operation of average pooling $\mathcal{P}_m$ in the encoder when $m = n$.

In order to obtain the final outputs, we apply final layers $\mathcal{S}_n$ to regress the distribution values and $\mathcal{T}_n$ to classify node types. In particular, $\mathcal{T}_n$ on level n predicts the type of each decoded node q, outputting the probability $p^{n,q}_{\text{leaf}}$ of the node being a leaf node. During inference, when $p^{n,q}_{\text{leaf}} > 0.5$ then the node is decoded as a leaf node, otherwise ($p^{n,q}_{\text{leaf}} < 0.5$) the current node is split into four children nodes in the next tree level. $\mathcal{R}_n$ is applied to regress a *relative* distribution value $\mathbb{V}^{n,q}_{\text{rec}}$ for each node q to its parent node at each level n; we apply the SoftMax in $\mathcal{R}_n$ to output the final relative values, ensuring $\sum_q \mathbb{V}^{n,q}_{\text{rec}} = 1$ for the four child nodes. This whole iterative decoding process is written as:

$$\begin{aligned} \mathbb{M}^{n+1}_{\text{dec}} &= \mathcal{S}_n(\mathcal{U}_n(\mathbb{M}^n_{\text{dec}}), \mathbb{M}^{n+1}_{\text{enc}}); \\ \mathbb{V}^{n,q}_{\text{rec}} &= \mathcal{R}_n(\mathbb{M}^{n,q}_{\text{dec}}) \\ p^{n,q}_{\text{leaf}} &= \mathcal{T}_n(\mathbb{M}^{n,q}_{\text{dec}}) \end{aligned} \tag{14}$$

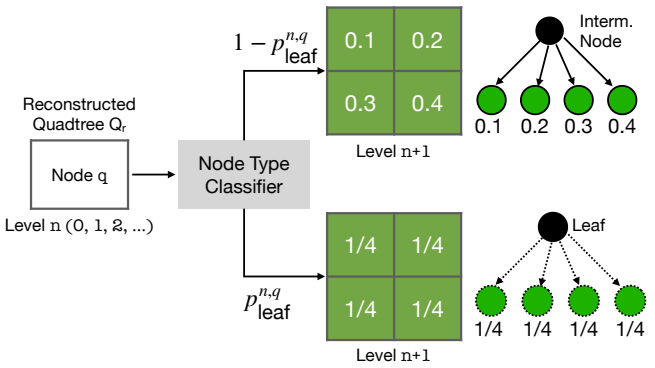

Fig. 5. Illustration of the loss computation. After the tree Q_r is hierarchically reconstructed by the network, we compute a loss value for every node q at every level n from the top to the bottom. We compute the expected distribution value (Eqn. 16) depending on how probable p is a leaf (i.e., p_{leaf}) predicted by the node classifier $\mathcal{T}_n$. Note that, when p is a leaf, the corresponding distribution values for the next level are just $\frac{1}{4}$.

Here, $\mathcal{S}_n$ takes both the upsampled feature $\mathcal{U}_n(\mathbb{M}^n_{\text{dec}})$ at the $(n+1)$-th level (upsampling increases n by one) and the skip-link feature $\mathbb{M}^{n+1}_{\text{enc}}$ from the same level of the encoder. This skip-link is inspired by the traditional U-Net [Ronneberger et al. 2015] architecture and it makes the neural network more robust to spatial size variations through pooling and upsampling. Without skip links, we have to decode an entire $\mathbb{Q}_r$ from only the last layer feature $\mathbb{M}^0$, which is much more difficult and can end up having a shallow tree.

In summary, the decoding process starts from the coarsest 0-th level and gradually builds $\mathbb{Q}_r$ until reaching $n_{\max} = 20$. If all the nodes are leaves when reaching a level, the decoding process terminates early. In practice, the input and output tree can have different numbers of levels. Note that our neural network only predicts the relative value $\mathbb{V}^{n,q}_{\text{rec}}$ (= 1 if q is the root node) for every node q on every level n with respect to its parent node value, and actual absolute values in the trees are reconstructed by unrolling the relative values using a series of multiplications. This hierarchical encoder and decoder architecture efficiently extends U-Net style CNNs to quadtrees that are naturally more sparse than image grids.

6.5 Loss function

We train the network to output accurate quadtree distributions as close to the ground-truth quadtrees as possible. The ground-truth trees are generated in the same way as our input trees $\mathbb{Q}_h$ (Sec. 5), by tracing and accumulating a large number of samples until converged (more details in Sec. 8). As a result, for each spatial location, we have its ground-truth quadtree $\mathbb{Q}_{\text{gt}}$ with node type label $\gamma^{n,q}_{\text{leaf}}$ and distribution value $\mathbb{V}^{n,q}_{\text{gt}}$ for each node. Here, $\gamma^{n,q}_{\text{leaf}}$ represents the node type in the ground-truth tree, which is deterministic and binary.

Therefore, we can supervise our network output $p^{n,q}_{\text{leaf}}$ and $\mathbb{V}^{n,q}_{\text{rec}}$ with the ground-truth $\gamma^{n,q}_{\text{leaf}}$ and $\mathbb{V}^{n,q}_{\text{gt}}$ respectively. However, since the ground-truth tree $\mathbb{Q}_{\text{gt}}$ is generated using a lot of samples, its structure can be very deep and fine-grained corresponding to a high-resolution distribution; enforcing the network to reconstruct such a deep quadtree structure from sparse input samples is highly challenging and even unrealistic, especially at the beginning of rendering. Therefore, we let the network put more emphasis on regressing accurate distribution values; we seek to allow a different tree structure as long as its final distribution is close to the ground truth. To this end, we focus on the expected distribution value for each node, without directly supervising the tree structure.

Given a parent node q and its four potential child nodes q_c, we compute the expectation of the distribution value $\mathbb{V}^{n+1,q_c}_{\text{rec}}$ for each q_c utilizing the node type probability $p^{n,q}_{\text{leaf}}$ of the parent node:

$$\mathbb{E}[\mathbb{V}^{n+1,q_c}_{\text{rec}}] = p^{n,q}_{\text{leaf}} \cdot \frac{1}{4} + (1 - p^{n,q}_{\text{leaf}}) \cdot \mathbb{V}^{n+1,q_c}_{\text{rec}}. \tag{15}$$

Note that, our regressed distribution value $\mathbb{V}^{n+1,q_c}_{\text{rec}}$ is a relative value, i.e. a ratio of its actual value to its parent node value. If the parent node q is a leaf node, the distribution is assumed uniform inside the node and thus the corresponding relative value for the same region of each q_c is just exactly $\frac{1}{4}$, multiplying $p^{n,q}_{\text{leaf}}$, which is the relative distribution value if q is a leaf and q_c does not exist. We propose to supervise the expected value $\mathbb{E}[\mathbb{V}^{n+1,q_c}_{\text{rec}}]$ with the ground-truth value $\mathbb{V}^{n+1,q_c}_{\text{gt}}$ for all children nodes q_c of q. This loss is given by:

$$\mathbb{L}^{n,q}_{\text{value}} = \sum_{q_c} \|\mathbb{E}[\mathbb{V}^{n+1,q_c}_{\text{rec}}] - \mathbb{V}^{n+1,q_c}_{\text{gt}}\| \tag{16}$$

Similar to the above discussion for Eqn. 15, if the ground-truth node q is a leaf ($\gamma^{n,q}_{\text{leaf}} = 1$) and q_c does not exist, we just use $\mathbb{V}^{n+1,q_c}_{\text{gt}} = \frac{1}{4}$. This loss (Eqn. 16 with Eqn. 15) jointly supervises the predicted node type probabilities and the distribution values. However, we find in our experiments that using this loss only can be unstable in the early training time. We therefore provide direct supervision for the tree structure at the beginning of the training, using a binary cross entropy loss $\mathbb{L}^{n,q}_{\text{class}}$ that supervises $p^{n,q}_{\text{leaf}}$ with $\gamma^{n,q}_{\text{leaf}}$. We apply deep supervisions to every generated tree node output on all the levels and sum their losses up. Our full loss function is expressed by

$$\mathbb{L}_{Q_r} = \sum_{n=0}^{n_{Q_r}} \sum_{q \in q_n} (\beta \mathbb{L}^{n,q}_{\text{class}} + \mathbb{L}^{n,q}_{\text{value}}) \tag{17}$$

where $\mathbb{L}_{Q_r}$ denotes the loss of the whole reconstructed tree $\mathbb{Q}_r$ summed over every node q on every decoding level n. Here, q_n is the set of nodes on the n-th level, n_{Q_r} is the actual maximum decoding level, and β is a weight factor. During training, we start with $\beta = 1$ in Eqn. 17 to stabilize the early optimization by supervising both the structure (with $\mathbb{L}_{\text{class}}$) and values (with $\mathbb{L}_{\text{value}}$), and then gradually reduce β to zero. Therefore, eventually, we supervise the sampling map implicitly using $\mathbb{L}_{\text{value}}$ without forcing the network to output the same target quadtree structure $\mathbb{Q}_{\text{gt}}$ (an over-strong regularization and often impossible to achieve). Our neural network can generalize well to new scenes since it mainly operates on the local sample input without any strong global scene-level dependency.

7 PATH GUIDING AND RENDERING

We use an iterative algorithm to trace and deposit samples, accumulate the initial quadtrees $\mathbb{Q}_h$, reconstruct the accurate quadtrees $\mathbb{Q}_r$ as sampling distributions, and use the learned distributions for rendering the final image. Here, we share a similar design with

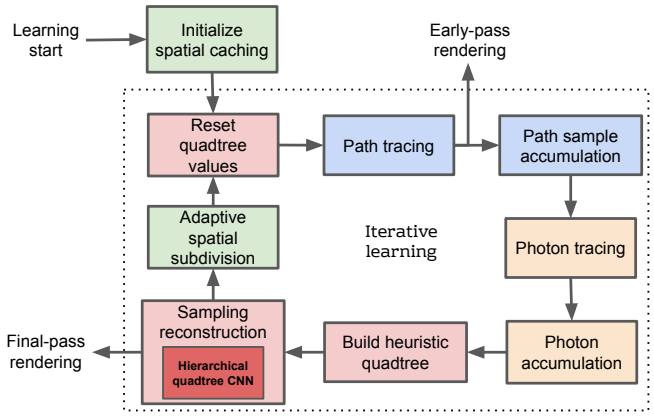

Fig. 6. Illustration of iterative learning and rendering. We show the pipeline of our path guiding and rendering process (Sec. 7). It starts by building a coarse grid $\mathbb{G}$, which is later iteratively refined online. We trace a set of path samples to detect valid spatial voxels in $\mathbb{G}$ for storing sampling distributions, as well as accumulating their input features into per-voxel quadtrees Q_h (Eqn. 10); these path samples also contribute radiance to the rendering result. We then trace photons from the light and deposit them to the corresponding quadtrees Q_h in their arriving spatial voxels. These accumulated quadtrees Q_h are adaptively subdivided (Sec. 5) based on the sample information they accumulate in this iteration. We then send Q_h to our neural network and reconstruct accurate quadtrees Q_r, which will be used as sampling distributions to guide the path tracing in the next iteration. Afterwards, we refine the voxels of the spatial grid $\mathbb{G}$ as needed. Before moving to the next-iteration path tracing, we reset the values in Q_h to zero, while retaining their tree structure to continue to accumulate samples and possibly obtain further refined quadtrees in the next iteration.

many state-of-the-art path guiding works [Müller et al. 2017; Müller 2019; Rath et al. 2020; Zhu et al. 2020b], as presented in Fig. 6.

Spatial caching of the sampling distributions. We use a hierarchical hash grid $\mathbb{G}$ [Zhu et al. 2020b] in the scene to receive the hybrid samples, store the input $\mathbb{Q}_h$ and output $\mathbb{Q}_r$ in individual voxels. In the first iteration, the traced path samples are collected to determine the bounding box of our spatial grid, which covers the *visible* part of the scene. Next, we start from a discrete 3D volume that uniformly partitions the visible scene space where each voxel is a cube with a side length $\mathbb{R}_\mathrm{B}/r_\mathrm{init}$ where $\mathbb{R}_\mathrm{B}$ is the diagonal length of the initial estimated bounding box; each voxel receives hybrid samples and builds sampling quadtrees, which can be further sub-partitioned to a KD-tree as needed. This leads to a hierarchical spatial grid with per-voxel sampling distributions. Here, each voxel is iteratively subdivided to a KD-tree based on a simple but effective criterion: if the total number of samples N_spt within a spatial voxel is larger than a pre-defined threshold k_spt (i.e., $N_\mathrm{spt} > k_\mathrm{spt}$), then we split the voxel into two sub-voxels through the middle plane along an alternating dimension. This subdivision is applied recursively to sub-voxels until all voxels do not satisfy the subdivision criteria, similar to the strategy proposed by Muller et al. [2017]. Therefore, when a new sample arrives, we first search within the hierarchical spatial grid $\mathbb{G}$ to find the voxel that covers the sample, then deposit the sample to its stored quadtree $\mathbb{Q}_h$ (Sec. 5). In practice, we also jitter the sample across neighboring spatial voxels (similar to depositing a sample into the angular quadtree in Sec. 5), which creates a spatial stochastic box filtering as is done in Müller [2019].

Importance sampling from the quadtree. When guiding paths, the importance sampling of ω_i is done by traversing the $\mathbb{Q}_r$ from the top to the bottom similar to [Müller et al. 2017; Rath et al. 2020]. From the root node of the quadtree, we iteratively sample one from the four child nodes based on their relative value $\mathbb{V}_\mathrm{rec}^{n,q}$, until reaching a leaf node. We then uniformly sample the leaf node. Suppose the leaf has a solid angle bin $\Delta z \cdot \Delta\phi$ in the $(\mathbb{Z}, \Phi)$ space, then the final sampling PDF corresponds to $p_\mathrm{leaf} = 1/(\Delta z \cdot \Delta\phi)$.

Iterative sample tracing and rendering. Similar to most of the previous work [Müller et al. 2017; Rath et al. 2020; Zhu et al. 2020b], we iteratively trace samples (though our samples are uniquely hybrid) to refine our quadtrees over time. Specifically, in the t-th iteration ($t = 0, 1, 2, ..., \mathbb{T}$), we trace 2^t sample-per-pixel (SPP) camera and light rays; each bounce point of the ray yields a path sample or a photon, which is deposited into the quadtree in a corresponding spatial voxel as discussed above. Each spatial voxel stores an input online accumulated quadtree $\mathbb{Q}_h$ and a neurally reconstructed quadtree $\mathbb{Q}_r$ for path guiding. After each iteration, we reconstruct a new $\mathbb{Q}_r$ from the current $\mathbb{Q}_h$; the values of the input quadtree $\mathbb{Q}_h$ are then cleared to zero, and $\mathbb{Q}_h$ continues to accumulate new samples in the next iteration, while inheriting the same tree structure.

After $\mathbb{T}$ iterations, we discontinue learning distributions and initiate a final pass where we use the most recent reconstructed quadtrees $\mathbb{Q}_r$ from the $(\mathbb{T})$-th iteration for guiding the rest of the path samples. Since our approach allows the learning to stop earlier because of high-quality reconstructed distributions, we can save more samples for the final-pass rendering. The final rendered image combines the radiance of all samples from $t \geq 2$ iterations weighted by the inverse of their estimated per-pixel variances [Müller 2019].

Combining sampling strategies. The learned guiding sampling is combined with BSDF sampling via the one-sample MIS (Eqn. 5). In the iterative process ($t < \mathbb{T}$), we use $\alpha = 0.5$ for the one-sample MIS. For the final rendering pass, we follow Zhu et al. [2020b] to compute the blending coefficient α adaptively: $\alpha = \mathbb{E}_{\omega_o,\omega_i}[L^\mathrm{BSDF}_{\omega_o,\omega_i}]/(\mathbb{E}_{\omega_o,\omega_i}[L^\mathrm{BSDF}_{\omega_o,\omega_i}]+\mathbb{E}_{\omega_o,\omega_i}[L^\mathrm{guide}_{\omega_o,\omega_i}])$. Here, $\mathbb{E}_{\omega_o,\omega_i}[\cdot]$ is the expected radiance sent back to the viewing direction using one of the two sampling strategies, which is statistically estimated from the previously traced path samples in past iterations.

8 IMPLEMENTATION

In this section, we discuss some details in dataset generation, neural network training, and rendering.

Dataset generation. We create a large-scale dataset to train our neural network. We collect 50 complex indoor and outdoor scenes either used by researchers in previous papers [Vorba et al. 2014; Müller et al. 2017; Rath et al. 2020; Bako et al. 2019; Zhu et al. 2020b] or designed by artists from online resources [Bitterli 2016; Jakob 2010; Evermotion 2012; Trader 2020; Squid 2020; Blend Swap 2016]. Following Zhu et al. [2020a,b], we also add additional procedurally

Fig. 7. Multiple sets of training scenes, including diverse procedural random scenes from the previous work [Zhu et al. 2020b] and complex indoor and outdoor scenes designed by researchers and modeling artists.

generated training scenes, created by combining multiple randomized geometry primitives under area lights and environment maps. We hold out 12 (from the 50) complex scenes as testing scenes to evaluate our method. The remaining scenes are used for training.

We use the same method (described in Sec. 5) to create the input (noisy) and output (ground-truth) quadtrees, from the training dataset. In particular, we iteratively emit 2^t SPP camera and light rays in iteration $t \in [0, \mathbb{T}]$ to create path and photon samples and accumulate them in the spatial grid $\mathbb{G}$ in the scene, similar to the rendering process in Sec. 7. We obtain the input quadtrees $\mathbb{Q}_h$ by accumulating hybrid samples in the spatial voxels at every $t_{\text{in}} \in [0, 12]$ iteration. For every input $\mathbb{Q}_h$ at iteration t_{in}, we freeze the spatial cache $\mathbb{G}$ for the following iterations $t > t_{\text{in}}$, continue collecting more samples and repeatedly refine $\mathbb{Q}_h$ to create the $\mathbb{Q}_{\text{gt}}$ when reaching $t = t_{\text{gt}} = 20$. When accumulating hybrid samples, we use the BSDF marginalized variance-aware sampling function (Eqn. 7) for the path samples, unless otherwise stated in ablation studies (Sec. 9). As for photons, we simply use their power (Eqn. 8) as the input values.

We also apply additional data augmentation designs to increase the generalization ability of the neural networks. Since the hierarchical hash grid $\mathbb{G}$ has KD-trees $\mathbb{B}_{\text{spt}}$ containing spatial voxels of different sizes (Sec. 7), we augment the training data by selecting 10 different initial resolutions r_{init} equally spaced between $r_{\text{init}}^{\text{min}} = 10$ and $r_{\text{init}}^{\text{max}} = 200$, which can cover diverse voxel sizes. We also further augment the input by randomly rotating the global frames.

Neural network training. Our network architecture is designed to be compact for fast inference in rendering. The maximum number of feature channels in our neural network is set to be 128. While this leads to efficient sampling reconstruction, it is still challenging for such a single network to handle diverse inputs with various numbers of input samples or very different sparsity levels. Therefore, we train five separate versions of the same network as is done in [Zhu et al. 2020b], where each one only needs to handle the input $\mathbb{Q}_h$ that contains a certain range of sample numbers (i.e., $[0, 100)$, $[100, 500)$, $[500, 1000)$, $[1000, 5000)$, $[5000, \infty)$). During both training and testing, we split the set of $\mathbb{Q}_h$ into these smaller groups, and these networks are executed on GPUs in parallel to reconstruct the set of $\mathbb{Q}_r$. We train these networks using the ADAM optimizer [Kingma and Ba 2014] with a learning rate of 1.0×10^{-4} until convergence.

Rendering. When rendering, we stop learning distributions after $5 \sim 10$ iterations depending on the actual light transport complexity of each scene, and guide the remaining path samples in the final-pass rendering. Experiments are rendered on a workstation with an Intel Core i9-7960X CPU and two Nvidia Titan RTX GPUs required to run our neural networks. For some simple testing scenes, one GPU is sufficient. Sample tracing and rendering are performed on the Mitsuba engine [Jakob 2010]. The neural network is integrated into the rendering engine using the TensorFlow C++ API with acceleration libraries, and other standard C++/CUDA libraries for efficient data streaming. To utilize the potential parallelization between the CPU and GPUs, the CPU keeps ray tracing and rendering the current-pass result using the previously reconstructed sampling distributions until the GPU finishes computing a new set of $\mathbb{Q}_r$ and updating those distributions. This effectively keeps the CPU and GPU running busy and staying at high utilization. Our quadtree-based neural networks are efficient to evaluate. The GPU processing time is about $6\% \sim 15\%$ (varying across scenes) of the CPU processing time in our experiments. In the future, implementing our proposed neural path guiding framework into a GPU-based rendering engine leveraging hardware ray-tracing (e.g., [Parker et al. 2010]) can possibly result in higher efficiency in practice.

9 RESULTS

We present extensive evaluation in this section. Additional experiments can be found in the supplementary material.

Configuration. We evaluate our method on 12 complex testing scenes, each containing complex global illumination and diverse geometric variations. When rendering each scene, we limit the maximum number of bounces to 20; Next Event Estimation (NEE) is turned off (except for Fig. 3) to clearly show the effectiveness of path guiding for ours and all comparison methods. We compare our methods with several traditional online path guiding methods [Müller et al. 2017; Müller 2019; Rath et al. 2020; Ruppert et al. 2020] which do not leverage deep learning techniques (CPU-only) but either use hierarchical quadtrees (similar to ours) or mixture models as their sampling distribution representation. We also compare with neural guiding methods, including one [Bako et al. 2019] that can only guide the first bounce and a recent photon-driven approach [Zhu et al. 2020b] that can guide multiple bounces; these previous neural methods represent sampling distributions as regular images. For quantitative results, we use the standard relative Mean Squared Error (rMSE) widely used in previous work [Rath et al. 2020; Zhu et al. 2020b]. All the numbers are computed on tone-mapped LDR images. In addition, we also show the memory cost of each method.

Qualitative and quantitative comparisons. Figure 8, 9 and 10 show equal-time comparisons between our method and previous path guiding methods on various complex (indoor, outdoor, and object) scenes. Note that, our approach often achieves better qualitative and

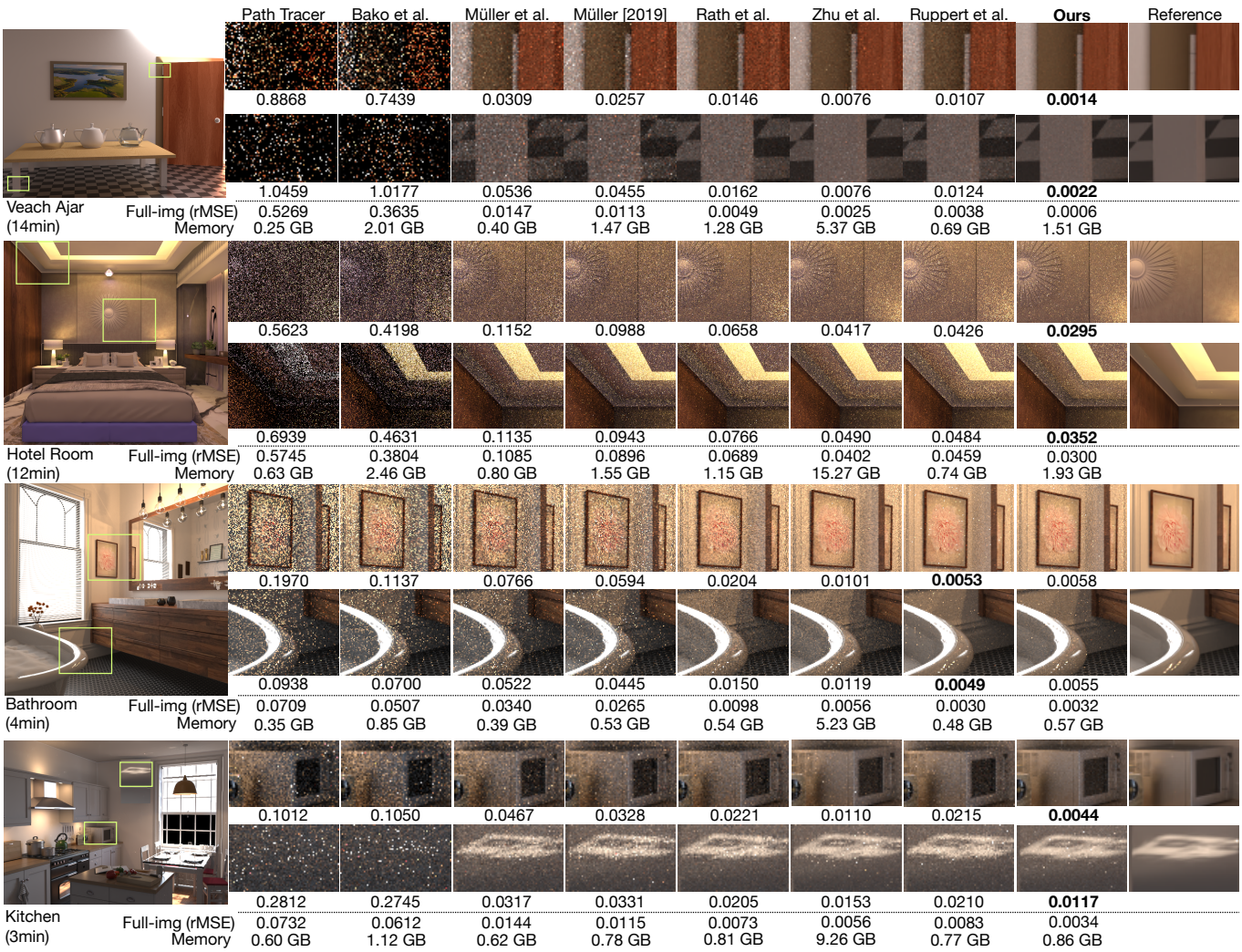

Fig. 8. Equal-time comparisons. We compare our method with previous path guiding methods [Müller et al. 2017; Müller 2019; Bako et al. 2019; Rath et al. 2020; Zhu et al. 2020b; Ruppert et al. 2020] on complex indoor scenes. For each scene, we show visual comparisons on two crops with corresponding rMSE numbers. We also show the rMSE of the full image and the memory usage for all the methods. Our approach often achieves better visual quality and lower rMSE (on both crops and full images). Our method achieves this with memory cost that is comparable to traditional methods [Müller 2019; Rath et al. 2020] and much less than the previous neural technique [Zhu et al. 2020b].

quantitative results. Our results of zoomed-in rendering crops are smoother, showing less noticeable noise than other results, and are visually closer to the reference. In contrast, the previous first-bounce guiding method [Bako et al. 2019] cannot handle these challenging cases very well, although it also leverages deep learning techniques; it can only improve the primary bounce sampling thus performs worse than the other guiding methods including the traditional online ones on our testing scenes with strong indirect illumination. The three traditional methods [Müller et al. 2017; Müller 2019; Rath et al. 2020] use pure path samples as input and reconstruct hierarchical quadtree distributions online for multi-bounce path guiding. They achieve effective path guiding and improve over the standard path tracing; in particular, Rath et al. [2020] shows clear advantages over the other two because of its more efficient variance-aware sampling distribution. Other than the quadtree, Ruppert et al. [2020] leverages mixture models (VMMs) to fit path samples by an online adaptive optimization framework, which outperforms many other techniques due to the careful positioning of mixture components and a novel parallax compensation module. However, these methods still leverage a slow online learning process, requiring a large number of path samples and many iterations to achieve accurate distributions for path guiding. The recent photon-driven neural method [Zhu et al. 2020b] uses a pre-trained network to relieve this slow online learning, leading to better results. However, this technique [Zhu et al. 2020b] (same to [Bako et al. 2019]) can only reconstruct sampling distributions as regular 2D images (unlike quadtress) that have a

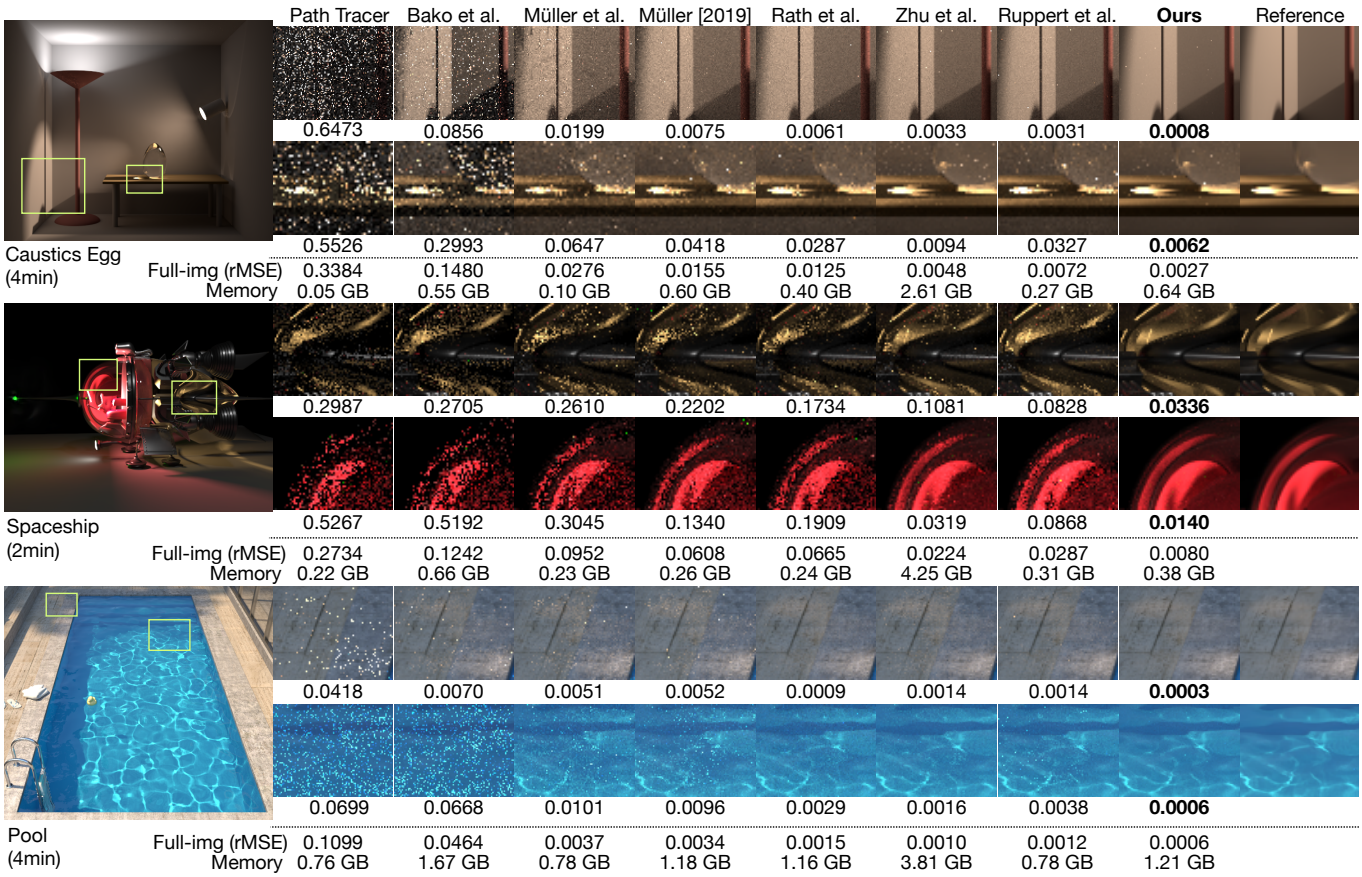

Fig. 9. Equal-time comparisons. Similar to Fig. 8, we show more equal-time comparisons between our method and previous path guiding methods [Müller et al. 2017; Müller 2019; Bako et al. 2019; Rath et al. 2020; Zhu et al. 2020b; Ruppert et al. 2020]. Our method can also achieve better qualitative and quantitative results using moderate memory costs.

fixed low resolution, hence restricting the accuracy and efficiency of sampling. Our approach instead directly regresses hierarchical quadtrees from hybrid samples for sampling and can represent more fine-grained distributions under different light transport conditions. As a result, our approach further outperforms [Zhu et al. 2020b].

We achieve better rendering quality without a large memory overhead; the sparseness of our representation and the effectiveness of our neural reconstruction lead to high memory-efficiency. The recent neural technique [Zhu et al. 2020b] requires much larger memory due to the use of grid representation (image). For most scenes, our memory consumption is comparable to the traditional methods [Müller 2019; Rath et al. 2020] without deep learning.

Hybrid samples. To further demonstrate the effectiveness of using hybrid samples, comparisons on two extreme light transport settings are shown in Fig. 3 earlier in the paper. These two Cornell Box scenes are specifically designed to make only one type of the input samples (either paths or photons) useful. Previous methods that use either path samples or photon samples cannot work effectively on both challenging cases. In contrast, our approach uses a hybrid of both path samples and photons with a novel hierarchical neural reconstruction, leading to more robust rendering on both cases. Our neural network learns to correlate the information and convert it into a single high-quality hierarchical sampling distribution. As demonstrated in other results of complex scenes (Fig. 1, 8 and 9), our proposed framework with hybrid input can robustly work well across various challenging light transport cases.

Convergence. We also evaluate how our method performs with an increasing number of samples. In particular, we run our method on two testing scenes (Racing Car and Kitchen, shown in Fig. 1 and 8) with different total numbers of traced rays (including both camera and light rays) per pixel and compare the rMSEs with other methods using the same budgets of sampling rays. The results are shown in Fig. 11. We can see that our novel neural path guiding approach consistently achieves lower errors with more samples; ours also has smaller errors compared to previous methods. Note that, while the recent neural method [Zhu et al. 2020b] can often achieve better results than the other traditional methods with a moderate sampling budget, its gain gets reduced with very large sampling budgets due to the fixed resolution sampling map which intrinsically cannot express the high-frequency lighting perfectly. On the other hand,

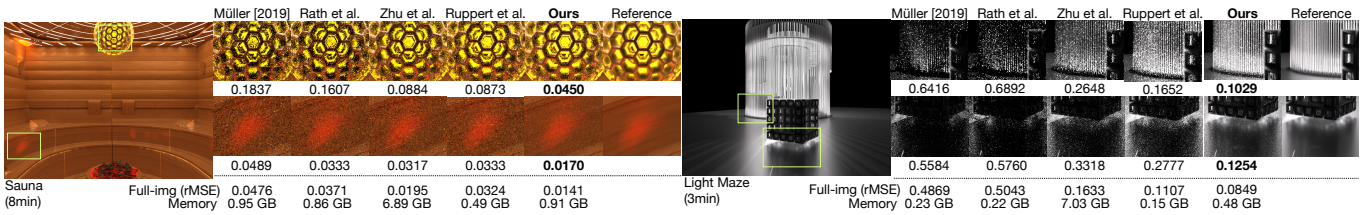

Fig. 10. Equal-time comparisons with some best performing baseline methods [Müller 2019; Rath et al. 2020; Ruppert et al. 2020] on two complex-visibility scenes. The incident radiance fields of these scenes contain high-frequency details and repeated patterns. We can still achieve better results in such light transport scenarios.

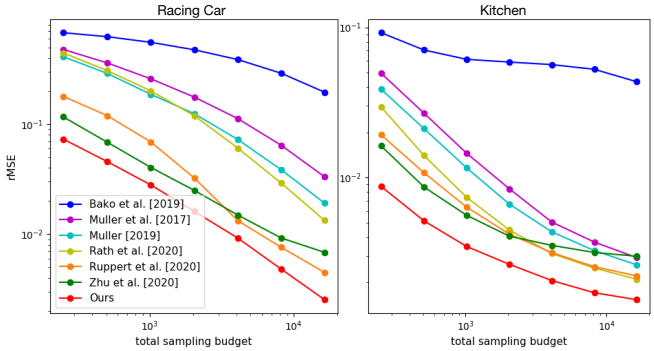

Fig. 11. Convergence curves of two testing scenes (from 256 SPP to 16,384 SPP). We compare our approach with previous methods using different numbers of samples. The sampling budget represents the total number of rays (including both camera and light) per pixel through the entire guiding and rendering process. Both the X (number of samples) and Y (rMSE) axes are on a logarithmic scale. Our hierarchical neural path guiding performs consistently better with the increasing samples on these two scenes. Because some rays are used for guiding and the convergence is influenced by the quality of the guiding distribution, these curves are not straight lines, as expected for standard path tracing in a log-log plot. Also note that, the recent previous neural method [Zhu et al. 2020b] may not be more effective than the traditional methods when using a very large number of samples.

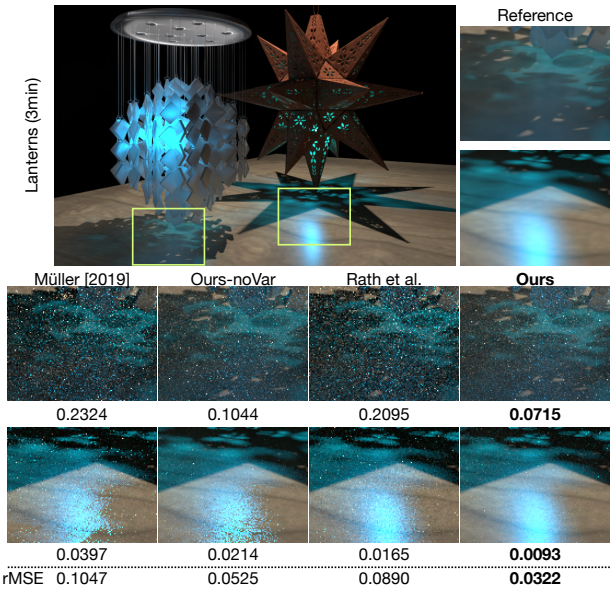

Fig. 12. Target sampling distribution. Our approach by default uses the variance-aware sampling function (Eqn. 7) as the target to train the network. We can also use a simpler target sampling distribution (Eqn. 4) without variance-awareness. We compare this with our default model and also traditional methods using the two different sampling functions. Our full model performs better, which justifies the variance-aware technique and necessity of using an advanced target distribution for training.

traditional quadtree-based methods [Müller et al. 2017; Müller 2019; Rath et al. 2020] can be more fine-grained with a large number of samples, leading to better results eventually. This example illustrates the benefits of having a hierarchical representation. Our approach successfully applies hierarchical quadtree-based sampling in neural path guiding, leading to efficient rendering.

Hierarchical reconstruction. We show some examples of the reconstructed sampling distributions in Fig. 3 and 13. Our regressed quadtree distributions are accurate and fine-grained, and are close to the reference. In contrast, [Rath et al. 2020] is reconstructing the same target distribution as ours, but it leverages traditional online accumulation, which often obtains more noisy quadtrees. Essentially, our neural network is trained to denoise such noisy online-accumulated quadtrees into the smooth and accurate quadtrees. On the other hand, the neural techniques [Zhu et al. 2020b] that use uniform grids (images) as the sampling representation can also reconstruct smooth sampling distributions. However, because of the limited image resolution, their sampling is less sharp and detailed compared to our reconstructed quadtrees.

We further investigate the benefits of using the hierarchical network, by training and comparing with a network that regresses 2D images from hybrid samples without any hierarchical structure. In particular, we use the recent network architecture of [Zhu et al. 2020b] and train it using the same hybrid samples as input and the same variance-aware target distribution for path guiding. The corresponding results compared with the results of our full model and other methods are shown in Fig. 13, with corresponding sampling distributions. This non-hierarchical network with the image representation performs worse than our full model. Meanwhile, our reconstructed hierarchical distribution contains more details and is faster to compute than the uniform image representation, which leads to more efficient path guiding and better rendering quality.

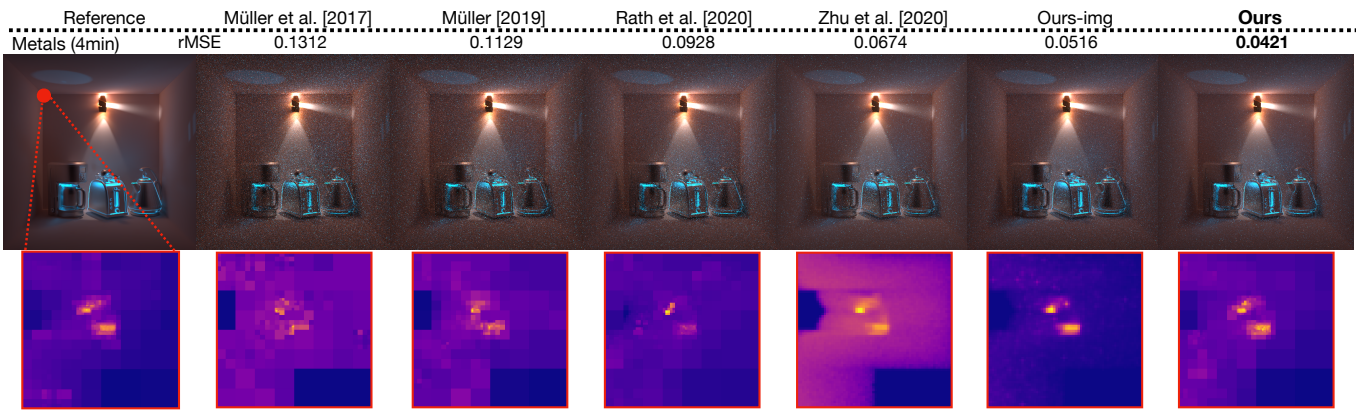

Fig. 13. Hierarchical reconstruction. We compare with previous methods and show the corresponding sampling distributions of all methods for a scene point (marked by the red point in the reference). Similar to Fig. 3, the ground-truth sample distribution is with respect to our method and [Rath et al. 2020] (and also the "Ours-img" variant). We also compare with a non-hierarchical variant (labeled with Ours-img) that takes hybrid samples as input but regresses image grid distributions using the same network architecture as [Zhu et al. 2020b]. Our full model leverages hierarchical reconstruction to regress accurate sampling distributions and achieve better results compared to its non-hierarchical counterpart.

Target sampling distribution. Our framework has good flexibility; it can support various target sampling distributions. By default, we use the recent variance-aware function [Rath et al. 2020] (Eqn. 7) for the better performance. In Fig. 12, we show results from a variant of our model trained with the traditional target sampling function without variance-awareness (Eqn. 7 as is used in [Müller et al. 2017; Müller 2019]). Note that, our approach still works well even without the variance-aware technique, and can still outperform many previous methods. Our full model can achieve better results, taking advantage of the advanced target sampling function.

Limitations. Our approach leverages path and photon samples and treats them equally, tracing the same number of rays for each type of sample in guiding and rendering. However, the two types of samples often do not contribute equally to the final distribution (as in Fig. 3) and one of them can be less useful, which is a waste of sampling budget. Addressing this may require future research to support distributing the samples non-equally, adaptive to the actual light transport cases. Besides, we believe guiding the photon emission and tracing properly (e.g., [Vorba et al. 2014]) can be very useful to our framework, which reduces the well-known photon visibility issue and further increases the robustness of our approach.

Our framework utilizes discrete voxels to partition the scene and cache the sampling distributions. Similar to previous methods that use similar caching techniques [Müller et al. 2017; Zhu et al. 2020b], this spatial structure can have discontinuous sampling distributions across neighboring voxels, leading to some aliasing artifacts that are usually gone after a number of iterations. While our neural framework accelerates the convergence of sampling reconstruction, which alleviates this issue to some extent, some minor artifacts can still appear in early iterations. Exploring an idea similar to the parallax compensation [Ruppert et al. 2020] is left for future work.

Currently, we implement our approach in a hybrid CPU and GPU fashion where tracing/shading and sampling reconstruction are executed separately. The extra data copying overhead is still non-negligible even if we carefully manage the data flow and parallelization. In practice, it can be beneficial to put more modules on GPUs directly and make use of the specialized processor cores.

10 CONCLUSION AND FUTURE WORK

In this paper, we present a novel path guiding framework that is learning-based, hierarchical and hybrid. We present an unique neural network that extends traditional CNNs to hierarchical representations, and produces accurate sampling distributions faster than traditional online accumulation methods. Our approach further uses a hybrid of path samples and photons as input, allowing for increased robustness and generality across different complex light transport scenarios. We demonstrate extensive experiments on diverse testing scenes. Our proposed neural path guiding framework can achieve the state-of-the-art rendering quality with a reasonably small memory cost compared to other existing approaches.

Our approach also inspires interesting future research. In this work, we focus on making the local directional distribution reconstruction neural and hierarchical. Future work can explore if the spatial caching grid can also be hierarchically reconstructed via a neural network, potentially making local distribution reconstruction aware of the global context. Another interesting direction is to combine our offline neural framework with the online neural techniques [Müller et al. 2019, 2020] that regress a global and continuous sampling function. Meanwhile, combining with the adjoint Russian roulette and splitting technique [Vorba and Křivánek 2016] and extending our framework to product sampling can be the direct next steps. Our approach leverages quadtree-based neural modeling for local light field approximation; this technique can also inspire other related research areas in computer graphics, such as lighting estimation and light transport acquisition.

ACKNOWLEDGMENTS

This work was supported in part by NSF grants 1703957 and 1764078, Google Ph.D. Fellowships, the Ronald L. Graham Chair and the UC San Diego Center for Visual Computing.

REFERENCES

Steve Bako, Mark Meyer, Tony DeRose, and Pradeep Sen. 2019. Offline Deep Importance Sampling for Monte Carlo Path Tracing. In *Computer Graphics Forum*, Vol. 38. Wiley Online Library, 527–542.

Steve Bako, Thijs Vogels, Brian McWilliams, Mark Meyer, Jan Novák, Alex Harvill, Pradeep Sen, Tony Derose, and Fabrice Rousselle. 2017. Kernel-predicting convolutional networks for denoising Monte Carlo renderings. *ACM Trans. Graph.* 36, 4 (2017), 97–1.

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

LLC Blend Swap. 2016. Blend swap.

Chakravarty R Alla Chaitanya, Anton S Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. 2017. Interactive reconstruction of Monte Carlo image sequences using a recurrent denoising autoencoder. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–12.

Kashyap Chitta, Jose M Alvarez, and Martial Hebert. 2020. Quadtree Generating Networks: Efficient Hierarchical Scene Parsing with Sparse Convolutions. In *The IEEE Winter Conference on Applications of Computer Vision*.

Stavros Diolatzis, Adrien Gruson, Wenzel Jakob, Derek Nowrouzezahrai, and George Drettakis. 2020. Practical Product Path Guiding Using Linearly Transformed Cosines. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 23–33.

TM Evermotion. 2012. Evermotion 3d models.

Iliyan Georgiev, Jaroslav Křivánek, Tomas Davidovic, and Philipp Slusallek. 2012. Light transport simulation with vertex connection and merging. *ACM Trans. Graph.* 31, 6 (2012), 192–1.

Ben Graham. 2015. Sparse 3D convolutional neural networks. *arXiv preprint arXiv:1505.02890* (2015).

Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 2018. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 9224–9232.

Benjamin Graham and Laurens van der Maaten. 2017. Submanifold sparse convolutional networks. *arXiv preprint arXiv:1706.01307* (2017).

Jerry Guo, Pablo Bauszat, Jacco Bikker, and Elmar Eisemann. 2018. Primary sample space path guiding. In *Eurographics Symposium on Rendering*, Vol. 2018. The Eurographics Association, 73–82.

Toshiya Hachisuka, Shinji Ogaki, and Henrik Wann Jensen. 2008. Progressive photon mapping. In *ACM SIGGRAPH Asia 2008 papers*. 1–8.

Toshiya Hachisuka, Jacopo Pantaleoni, and Henrik Wann Jensen. 2012. A path space extension for robust light transport simulation. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 1–10.

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. 2010. Mitsuba renderer. http://www.mitsuba-renderer.org.

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

Henrik Wann Jensen. 1996. Global illumination using photon maps. In *Rendering Techniques' 96*. Springer, 21–30.

James T Kajiya. 1986. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*. 143–150.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Claude Knaus and Matthias Zwicker. 2011. Progressive photon mapping: A probabilistic approach. *ACM Transactions on Graphics (TOG)* 30, 3 (2011), 25.

Jaroslav Křivánek, Iliyan Georgiev, Toshiya Hachisuka, Petr Vévoda, Martin Šik, Derek Nowrouzezahrai, and Wojciech Jarosz. 2014. Unifying points, beams, and paths in volumetric light transport simulation. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–13.

Pradeep Kumar Jayaraman, Jianhan Mei, Jianfei Cai, and Jianmin Zheng. 2018. Quadtree convolutional neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 546–561.

Eric P Lafortune and Yves D Willems. 1993. Bi-directional path tracing. (1993).

Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao Zhang, and Leonidas Guibas. 2017. Grass: Generative recursive autoencoders for shape structures. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–14.

Manyi Li, Akshay Gadi Patil, Kai Xu, Siddhartha Chaudhuri, Owais Khan, Ariel Shamir, Changhe Tu, Baoquan Chen, Daniel Cohen-Or, and Hao Zhang. 2019. Grains: Generative recursive autoencoders for indoor scenes. *ACM Transactions on Graphics (TOG)* 38, 2 (2019), 1–16.

Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy J Mitra, and Leonidas J Guibas. 2019. StructureNet: hierarchical graph networks for 3D shape generation. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 242.

Thomas Müller. 2019. "Practical Path Guiding" in Production. In *ACM SIGGRAPH Courses: Path Guiding in Production, Chapter 10*. ACM, New York, NY, USA, 18:35–18:48. https://doi.org/10.1145/3305366.3328091

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.

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 (Proceedings of SIGGRAPH 2020)* 39, 4 (July 2020), 151:1–151:12. https://doi.org/10.1145/3386569.3392441

Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. 2017. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3577–3586.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 234–241.

Lukas Ruppert, Sebastian Herholz, and Hendrik P. A. Lensch. 2020. Robust Fitting of Parallax-Aware Mixtures for Path Guiding. *ACM Transactions on Graphics (TOG)* (2020).

Peter Shirley, Bretton Wade, Philip M Hubbard, David Zareski, Bruce Walter, and Donald P Greenberg. 1995. Global illumination via density-estimation. In *Rendering Techniques' 95*. Springer, 219–230.

Turbo Squid. 2020. 3D Models, Plugins, Textures, and more at Turbo Squid.

Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. 2017. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *Proceedings of the IEEE International Conference on Computer Vision*. 2088–2096.

CG Trader. 2020. Cg trader. *URL http://www. cgtrader. com* 4 (2020).

Eric Veach. 1997. *Robust Monte Carlo methods for light transport simulation*. Vol. 1610. Stanford University PhD thesis.

Eric Veach and Leonidas Guibas. 1995a. Bidirectional estimators for light transport. In *Photorealistic Rendering Techniques*. Springer, 145–167.

Eric Veach and Leonidas J Guibas. 1995b. Optimally combining sampling techniques for Monte Carlo rendering. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. 419–428.

Thijs Vogels, Fabrice Rousselle, Brian McWilliams, Gerhard Röthlin, Alex Harvill, David Adler, Mark Meyer, and Jan Novák. 2018. Denoising with kernel prediction and asymmetric loss functions. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–15.

Jiří Vorba, Johannes Hanika, Sebastian Herholz, Thomas Müller, Jaroslav Křivánek, and Alexander Keller. 2019. Path Guiding in Production. In *ACM SIGGRAPH Courses*. ACM, New York, NY, USA, 18:1–18:77. https://doi.org/10.1145/3305366.3328091

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.

Jiří Vorba and Jaroslav Křivánek. 2016. Adjoint-driven Russian roulette and splitting in light transport simulation. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–11.

Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. 2017. O-CNN: Octree-based Convolutional Neural Networks for 3D Shape Analysis. *ACM Transactions on Graphics (SIGGRAPH)* 36, 4 (2017).

Peng-Shuai Wang, Yang Liu, and Xin Tong. 2020. Deep Octree-based CNNs with Output-Guided Skip Connections for 3D Shape and Scene Completion. *Computer Vision and Pattern Recognition (CVPR) Workshops*.

Peng-Shuai Wang, Chun-Yu Sun, Yang Liu, and Xin Tong. 2018. Adaptive O-CNN: A Patch-based Deep Representation of 3D Shapes. *ACM Transactions on Graphics (SIGGRAPH Asia)* 37, 6 (2018).

Quan Zheng and Matthias Zwicker. 2019. Learning to importance sample in primary sample space. In *Computer Graphics Forum*, Vol. 38. Wiley Online Library, 169–179.

Shilin Zhu, Zexiang Xu, Henrik Wann Jensen, Hao Su, and Ravi Ramamoorthi. 2020a. Deep Kernel Density Estimation for Photon Mapping. In *Computer Graphics Forum*, Vol. 39. Wiley-Blackwell.

Shilin Zhu, Zexiang Xu, Tiancheng Sun, Alexandr Kuznetsov, Mark Meyer, Henrik Wann Jensen, Hao Su, and Ravi Ramamoorthi. 2020b. Photon-Driven Neural Path Guiding. *arXiv preprint arXiv:2010.01775* (2020).