Photon-Driven Neural Path Guiding

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 novel photon-driven neural path guiding approach that can effectively reduce the variance of path tracing results. This complex scene is lit by several decorative ceiling lights which are extremely difficult to discover in path tracing. We compare the equal-time (~20 minutes) rendering results with standard path tracing and state-of-the-art path guiding methods (including Müller et al. [2017], Bako et al. [2019], and Rath et al. [2020]), showing the crops of the rendered results with corresponding relative MSRs (rMSRs). Recently, Bako et al. [2019] use an offline trained neural network for path guiding; however, it only supports guiding the first bounce, which is not effective since this scene is dominated by indirect lighting. On the other hand, while traditional methods allow for multi-bounce path guiding, they are purely online learning methods and it is highly expensive for them to learn the complex sampling functions for this challenging scene. Our method utilizes an offline trained deep neural network and enables neural path guiding at any path bounces. Ours achieves the best rendering results qualitatively and quantitatively.

Although Monte Carlo path tracing is a simple and effective algorithm to synthesize photo-realistic images, it is often very slow to converge to noise-free results when involving complex global illumination. One of the most successful variance-reduction techniques is path guiding, which can learn better distributions for importance sampling to reduce pixel noise. However, previous methods require a large number of path samples to achieve reliable path guiding. We present a novel neural path guiding approach that can reconstruct high-quality sampling distributions for path guiding from a sparse set of samples, using an offline trained neural network. We leverage photons traced from light sources as the input for sampling density reconstruction, which is highly effective for challenging scenes with strong global illumination. To fully make use of our deep neural network, we partition the scene space into an adaptive hierarchical grid, in which we apply our network to reconstruct high-quality sampling distributions for any local region in the scene. This allows for highly efficient path guiding for any path bounce at any location in path tracing. We demonstrate that our photon-driven neural path guiding method can generalize well on diverse challenging testing scenes that are not seen in training. Our approach achieves significantly better rendering results of testing scenes than previous state-of-the-art path guiding methods.

1 INTRODUCTION

Monte Carlo path tracing has been widely used in photo-realistic image synthesis. However, while simple and flexible, path tracing can take a significant amount of time to generate noise-free images for complex scenes (e.g., Fig. 1). One critical challenge for Monte Carlo based methods is to effectively construct light transport paths connecting the light and the camera.

Many path guiding methods [Müller et al. 2017; Jensen 1995] have been presented to construct advanced distributions (usually approximating incident light fields or some variants of those) for importance sampling at local shading points, guiding the local path sampling for high-energy path construction. The recent successful ones are unidirectional guiding methods [Müller et al. 2017; Rath et al. 2020]; they rely on early path samples to discover high-energy sampling directions. However, this unidirectional path discovery process can be slow for a challenging scene that is dominated by indirect illumination. While using light paths is known to be efficient in exploring the path space, previous photon-driven or bidirectional path guiding methods [Jensen 1995; Vorba et al. 2014] are not yet efficient, requiring sampling a large number of light paths.
We present a novel path guiding approach that can achieve highly efficient path sampling using only a sparse set of light paths as input, thus significantly advancing the overall rendering speed. Inspired by the original path guiding work [Jensen 1995], we leverage photons to compute local sampling distributions for importance sampling in path tracing. As is done by Jensen [1995], a sampling distribution at any 3D local region can be easily obtained by binning local photons according to their directions (i.e., a 2D histogram map). However, such distributions are only reliable with locally dense enough photons, and, on the other hand, are usually low-quality and appear highly noisy with sparse photons (see Figs. 2 and 3).

We propose to use a deep neural network to reconstruct high-quality sampling maps for path guiding from low-quality noisy sampling maps that are acquired by binning sparse photons (see Fig. 2). Our approach is the first deep learning based photon-driven path guiding approach. In essence, we break down the complex path guiding problem, mainly focusing on reconstructing local sampling distributions represented as 2D maps (i.e., images), and thus pose this problem as one of the image-to-image reconstructions that can now be addressed by deep learning techniques. Our sampling map reconstruction network is effectively trained offline in a scene-independent way. The trained network is able to recover the accurate shapes of a diverse set of complex sampling distributions on challenging novel scenes, which enables highly efficient guided path tracing with complex global illumination effects.

Our network is designed to reconstruct high-quality sampling maps at local spatial regions. To make these sampling maps well distributed and locally representative in the scene space, we adaptively partition the entire scene space into a hierarchical grid, according to the complexity of local geometry and incident light. The sampling map of every leaf voxel in the grid is reconstructed by our network, enabling path guiding at any location in a scene. Note that, our approach allows for efficient guided path sampling at any bounce points; this is the first offline-learning neural path guiding approach that can guide arbitrary bounces. We demonstrate that our novel deep path guiding achieves significantly better rendering quality on various challenging scenes than previous state-of-the-art path guiding methods given equal rendering time (see Fig. 1).

In summary, our main contributions are:
- We present the first deep learning based photon-driven path guiding approach;
- To our knowledge, this is the first offline-learning neural path guiding approach that can guide arbitrary bounces;
- Our proposed framework generalizes well to unseen new scenes and produces significantly better rendering results.

2 RELATED WORK

Monte Carlo rendering. One central problem of computer graphics is to efficiently evaluate the rendering equation [Kajiya 1986], which describes how light transports globally inside a scene. Monte Carlo methods are among the most effective methods to compute the light transport, which require effectively sampling high-energy paths that connect the camera and light for efficient rendering. Since Monte Carlo path tracing was introduced in the seminal work by Kajiya [1986], numerous papers have developed more efficient methods to explore path space, including bidirectional path tracing [Laforetude and Willems 1993; Veach and Guibas 1995a] and metropolis light transport [Veach and Guibas 1997; Pauly et al. 2000]. These methods typically leverage importance sampling to sample sub-path directions at any bounces for each traced path traversing the scene. Since the incident illumination is unknown, the importance sampling usually only considers the reflectance term (with a cosine term) in the rendering equation (please refer to Sec. 3 for more details); this however is not efficient for challenging scenes with complex indirect lighting. Path guiding [Jensen 1995; Vorba et al. 2019] can instead provide more efficient importance sampling; our novel photon-driven path guiding approach can reconstruct high-quality sampling distributions that well approximate the complex incident light fields, thus leading to highly efficient rendering.

Photon-based rendering. Particle density estimation has also been applied in computer graphics to evaluate the rendering equation, which introduces photon mapping and many other particle-based methods [Shirley et al. 1995; Jensen 1996; Hachisuka et al. 2008; Knaus and Zwicker 2011]. These methods focus on photon density estimation at any given shading point, which avoids the high-frequency noise in MC rendering and is very effective for computing complex global illumination. Photon density estimation can only provide biased radiance or irradiance estimates, since it blurs the photon contributions within a certain kernel bandwidth (though this bias can be consistently reduced to zero by progressively reducing the bandwidth and tracing infinite photons [Hachisuka et al. 2008; Hachisuka and Jensen 2009; Knaus and Zwicker 2011]). Our goal is not to compute photon density for a single point but to approximate incident light fields for a local area (in a voxel) as sampling distributions. Therefore, we consider the integral of irradiance over an area (i.e., the incident flux), which can be effectively evaluated using photons in an unbiased way.

Recently, Zhu et al. [2020] introduce a deep learning based method for photon density estimation in photon mapping. They leverage a PointNet [Qi et al. 2017] style neural network to process individual photons. However, the complexity of running such a network grows linearly with the number of photons. We instead leverage a UNet [Ronneberger et al. 2015] style network and consider a raw photon histogram map, composed by binning photons [Jensen 1995], as input; therefore, the complexity of our network is independent to the photon count and runs in constant time. We show that our method consistently reconstructs better sampling distributions with more photons.

Path guiding. In general, path guiding aims to estimate the incoming light fields and draw samples accordingly to accelerate the convergence of Monte-Carlo rendering. The first path guiding technique is based on photons [Jensen 1995]: it traces light paths from the light sources, distributes photons in the scene, and constructs local photon histograms as sampling distributions for the importance sampling in path tracing. Though very efficient to compute, such histogram-based sampling maps are only of high quality when accumulating dense enough photons. We extend this simple classical histogram-based technique to a novel learning-based method in a new path guiding framework; our method regresses high-quality
sampling maps from sparse photons, avoiding expensively tracing a large number of photons.

Vorba et al. [2014] present a bidirectional guiding method, where both camera paths and light paths are guided using online fitted Gaussian-Mixture (GM) distributions at spatial cache points. This technique was further extended to product sampling [Herholz et al. 2016], and to account for parallax [Ruppert et al. 2020]. However, the online fitting process in these methods is usually slow and the GM model also makes it difficult to express high-frequency sampling distributions. Our approach leverages histograms as input (that can be easily computed at very low cost online) and an offline trained compact neural network that can rapidly reconstruct high-quality sampling maps with high-frequency details from the input.

Recently, unidirectional guiding methods have become more effective and practical, thanks to the efficient adaptive guiding framework introduced by Müller et al. [2017]. Many works extend this framework to achieve sampling in primary space [Guo et al. 2018], product sampling [Diolatzis et al. 2020], and variance-aware sampling [Rath et al. 2020]. These methods iteratively trace camera paths to adaptively reconstruct the incident light fields; this relies on early iteration paths to discover the light sources, in order to reconstruct reliable sampling distributions to guide the following iteration paths. However, the light discovery can be slow and unsuccessful for a scene with dominant indirect lighting, and the errors in the early-iteration sampling distributions can bias the path sampling in later iterations and never get fixed. In contrast, we leverage photons that are efficient in exploring indirect light transport; our learning based approach can also recover high-quality sampling distributions from sparse photons at an early stage, effectively avoiding a slow start in the guiding and rendering. Moreover, our photons are traced independently in each iteration, which avoids accumulating the sampling errors through multiple iterations.

**Neural path guiding.** Recently, deep learning techniques have been applied in path guiding. Müller et al. [2019] train an online neural network to perform importance sampling in global path space. This method can reproduce accurate ground-truth sampling functions, but the online training process is extremely slow. Some recent works leverage offline trained networks [Bako et al. 2019; Huo et al. 2020]; however, they only guide the path sampling at the first bounce. While we also leverage an offline trained neural network, our method instead leverages photons and supports guiding at any bounces, enabling significantly better rendering results than the first-bounce guiding approach [Bako et al. 2019] (see Fig. 1).

### 3 BACKGROUND

Physically-based rendering can be expressed by the Rendering Equation [Kajiya 1986] that describes the radiance leaving an intersection point $x$ in direction $\omega_o$:

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

where $L_e(x, \omega_o)$ denotes the emitted radiance, $L_i(x, \omega_i)$ is the incident radiance from direction $\omega_i$, $f_r$ is the bidirectional scattering distribution function (BSDF), and $\Omega$ is the visible hemisphere. The key component in the equation is the integral that computes the reflected radiance $L_r(x, \omega_o) = \int_{\Omega} L_r(x, \omega_i) f_r(x, \omega_i, \omega_o) \cos \theta_i d\omega_i$ over all directions in the hemisphere.

The integral can be numerically evaluated using Monte Carlo estimation [Veach 1997]:

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

(2)

where $N$ Monte Carlo path samples in various directions $\omega_i$ are drawn from the probability density function (PDF) $p(\omega_i)$. Considering global illumination with multiple bounces, $L_i(x, \omega_i)$ is in fact computed by recursively evaluating integrals using Eqn. 1. Therefore in Monte Carlo path tracing, rays are sampled from each intersection point to compute the radiance that contributes to the pixel color at multiple bounces.

The variance of the Monte Carlo estimate $L_r(x, \omega_o)$ can be reduced by sampling $\omega_i$ from a density function $p(\omega_i)$ that resembles the numerator $L_i(x, \omega_i) f_r(x, \omega_i, \omega_o) \cos \theta_i$. Ideally, if $p(\omega_i)$ and the numerator only differ by a constant scale, the variance is reduced to zero. However, this numerator is unknown and is as difficult as the integral to compute, due to complex visibility and indirect lighting in $L_i$; therefore, in practice, path tracing often proceeds with BSDF importance sampling.

Path guiding aims to reconstruct a density function that matches the shape of the numerator as closely as possible. In particular, since the standard BSDF importance sampling satisfies

$$p_{BSDF}(\omega_i) \propto f_r(x, \omega_i, \omega_o),$$

recent path guiding methods often set the target probability density to be proportional to the incident light [Vorba et al. 2014; Müller et al. 2017]

$$p_{guide}(\omega_i) \propto L_i(x, \omega_i) \cos \theta_i.$$  

(3)

The final sampling strategy is achieved by combining the guiding and BSDF sampling using one-sample Multiple Importance Sampling (MIS) [Veach and Guibas 1995b]:

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

(4)

where $\alpha$ is the mixture coefficient that determines the probability of choosing BSDF sampling or guided sampling.

Many recent works rely on early path samples in the path tracing to approximate the incident light field (Eqn. 3), which is not sufficient for challenging scenes with strong indirect illumination as shown in Fig. 1. We instead leverage photons traced from the light sources to compute the sampling density functions, which effectively explores the challenging light transport. Our novel approach advances the traditional path guiding with powerful deep learning techniques and an efficient spatial structure, thus enabling highly efficient path guiding from sparse photons.

### 4 OVERVIEW

Our path guiding approach uses a deep neural network to regress high-quality sampling maps that can be used to guide path sampling. Correspondingly, we introduce a novel practical path guiding framework that utilizes our neural network to reconstruct sampling maps in an adaptive spatial hierarchical grid, enabling effective path guiding at multiple bounces. In the following sections, we first introduce our sampling map parameterization, target sampling...
density, and how to use photons to compute the sampling maps in Sec. 5. We then introduce our deep neural network that can regress high-quality sampling maps given noisy low-quality sampling maps in Sec. 6. We present our full neural path guiding framework in Sec. 7, which describes our iterative guiding and rendering process, adaptive spatial structure, and how paths, photons, and the neural network are incorporated in the system. The implementation details are discussed in Sec. 8. We present an extensive evaluation of our method in Sec. 9. In the end, we conclude our paper and discuss future work in Sec. 10.

5 COMPUTING SAMPLING MAPS FROM PHOTONS

Previous methods [Jensen 1995; Vorba et al. 2014] usually compute hemispherical distributions at sampled surface points to approximate incident light fields. However, such hemispherical functions only approximate light fields at very local flat 2D surface regions, and are hard to interpolate on surfaces with complex normal variation. Inspired by the recent unidirectional path guiding methods [Müller et al. 2017; Rath et al. 2020; Bako et al. 2019], we utilize a full spherical sampling distribution (instead of a hemispherical one) that models the incident light distribution in a local 3D region. In particular, we build a hierarchical grid (see Sec. 7.1) in the scene space, and compute a spherical sampling distribution for each local 3D voxel of the grid. In this section, we discuss the representation of our sampling function and the computation of it from photons.

Spherical function representation. We use a regular directional grid that represents the sampling density function as a 2D sampling map (similar to [Bako et al. 2019]). We leverage the cylindrical mapping to parameterize the spherical domain for better area preservation (similar to [Müller et al. 2017]). In particular, a unit vector \( r = (x, y, z) \) (corresponding to a point on a unit sphere) is mapped to a 2D location \((u, v) = (z, \phi)\) on the sampling map, where \( \phi = \arctan(y/x) \). Our sampling map is like a standard environment map or radiance map in lighting representation, but ours is monochromatic and uses cylindrical mapping.

Target sampling density. As discussed in Sec. 3 (Eqn. 3), in general, the goal of path-guiding is to compute the sampling density at any position, making it proportional to the incident light \( L_i (x, \omega_i) \cos \theta_i \). For our discrete case we consider a 3D voxel region and a certain pixel range (representing a solid angle bin) of a sampling map, it is in fact the expected incident light that is of our interest. In particular, given a voxel \( j \) and a solid angle footprint of pixel \( k \) in the sampling map, the expected \( L_i (x, \omega_i) \cos \theta_i \) coming from the solid angle over the 2D surface area (that is of the scene geometry located in the voxel) inside the voxel is expressed by:

\[
\Xi(L_i (x, \omega_i) \cos \theta_i) = \frac{\int_{\Delta A_j} \int_{\Delta \Omega_k} L_i (x, \omega_i) \cos \theta_i d\omega dA}{\Delta \Omega_p \Delta A_j},
\]

(5)

where \( \Delta A_j \) represents the entire surface area of the scene geometry covered by the voxel \( j \), \( \Delta \Omega_k \) represents the solid angle footprint covered by the pixel \( k \) in the sampling map, and \( \Phi_{j,k} \) represents the total incident energy in the spatial and directional range. Therefore, it is the total energy (radiant flux)

\[
\Phi_{j,k} = \int_{\Delta A_j} \int_{\Delta \Omega_k} L_i (x, \omega_i) \cos \theta_i d\omega dA,
\]

(7)

that governs our sampling map distribution. Essentially, \( \Phi_{j,k} \) models the integrated incident radiance and is proportional to the sampling probability of a pixel \( k \) in a sampling map at a voxel \( j \). Note that, the irradiance \( E(x, \omega_i) = \int_{\Delta \Omega_k} L_i (x, \omega_i) \cos \theta_i d\omega \) at surface point \( x \) is a standard radiometry term and widely discussed in previous works [Jensen 1995; Rath et al. 2020], when divided by the total area, \( \Phi_{j,k} \) also describes the expected irradiance \( \Phi_{j,k} / \Delta A_j \) in the voxel. Therefore, we seek to obtain sampling densities that are proportional to the expected incident light:

\[
p_{\text{Guide}} (\omega_i) \propto \Phi_{j,k_i} / \Delta A_{k_i},
\]

(8)

where \( k_i \) is the pixel covering direction \( \omega_i \) in the sampling map, and we ignore the \( \Delta A_j \) in Eqn. 6 since it is a constant value for all solid angles in a voxel. This sampling density corresponds to a sampling map, each pixel value of which is proportional to \( \Phi_{j,k_i} \). We thus reconstruct a sampling map by normalizing an energy map that records the energy \( \Phi_{j,k_i} \) in each pixel.

Computing incident energy. In this work, we leverage particle tracing to effectively evaluate the integral of \( \Phi_{j,k} \) (Eqn. 7). We trace light paths from the light sources to distribute photons in the scene, where each photon carries a portion of flux; \( \Phi_{j,k} \) can then be
evaluated by simply binning the photons similar to [Jensen 1995]. In particular, $\Phi_{j,k}$ is estimated by:

$$
\Phi_{j,k} = \sum_{\omega_p \in \Delta \Omega_k, x_p \in \Delta A_j} \Delta \Phi_p, \tag{9}
$$

where $p$ denotes a photon, the photon arrives at the surface point $x_p$ from direction $\omega_p$, and $\Delta \Phi_p$ is the energy carried by the photon. Equation 9 essentially accumulates all the photon energies inside the voxel and directional bin.

Note that, Müller et al. [2017] leverages path tracing to accumulate the radiance samples inside a 3D region; this can also be seen as an integral (a Monte Carlo one) of the radiance over an area and a solid angle, similar to the energy integral of Eqn. 7. We leverage photon tracing to evaluate the integral and our particle-based approach provides an unbiased estimate for the energy $\Phi_{j,k}$ when the photon count goes to infinity.

Since the evaluation is governed by a single summation, we can progressively trace as many photons as needed, and accumulate the photons to compute an energy map without any memory bottleneck. Once a photon is accumulated in a voxel, the photon data is immediately deleted, except when the grid needs to be refined at the beginning (Sec. 7.1). Note that, an accurate energy map requires tracing a large number of photons, but in practice, we can only allow for tracing a small number of photons at rendering time, which by themselves cannot directly lead to high-quality sampling.

We propose to compute high-quality sampling maps, using a large number of photons, and take them as ground truth to train a deep neural network offline that can regress high-quality sampling maps online efficiently.

6 LEARNING TO REGRESS HIGH-QUALITY SAMPLING MAPS

While using a large number of photons can result in an accurate estimate of $\Phi_{j,k}$, it requires a significant amount of tracing time. On the other hand, computing a sampling map from sparse photons is fast, but the map is usually low-quality and appears noisy with many empty bins. As a result, neither using dense photons (too slow) nor sparse photons (too low-quality) is suitable for efficient path guiding. To overcome this, our central idea is to obtain accurate sampling maps offline as ground truth using dense enough photons, and then leverage supervised learning to regress such maps from low-quality maps that can be computed efficiently from sparse photons for path guiding. Specifically, we propose to train a deep convolutional neural network that learns to reconstruct a high-quality sampling map from sparse photons.

Our sampling maps are reconstructed iteratively through multiple iterations in our path guiding framework. Specifically, we consider a raw sampling map $S_{e,t}$ (1 channel) as input, acquired by accumulating a sparse set of traced photons from iteration 1 to $t$ using Eqn. 9, where $t$ denotes the iteration number and $e$ means accumulated photon energy. To give the neural network a hint on how the raw sampling map evolves over previous iterations with more photons, we keep the raw sampling map $S_{e,t-1}$ from the previous iteration also as an input channel. In addition, we record the number of photons per solid angle bin in $S_{e,t}$ and $S_{e,t-1}$, resulting in two additional maps $P_{e,t}$ and $P_{e,t-1}$, and use them as auxiliary buffers in the input, which provides two additional input channels. Inspired by the image inpainting techniques [Liu et al. 2018; Yu et al. 2019; Yi et al. 2020], we also concatenate a binary mask $B_{e,t}$ (1 channel) indicating whether a solid angle bin contains photon data or not, and use light-weight masked convolutions to process the input maps. As a result, our full input is a 2D image map with 5 channels in total and our network $F$ can be expressed by:

$$
\tilde{S}_d = F(S_{e,t}, S_{e,t-1}, P_{e,t}, P_{e,t-1}, B_{e,t}). \tag{10}
$$

Our network learns to regress a one-channel sampling map $\tilde{S}_d$, supervised by the ground-truth map $S_d$ computed from a large number of photons.

6.1 Network architecture

Note that, our network is essentially designed to solve an image-to-image reconstruction task. Many existing 2D neural networks for image-to-image denoising, translation, and inpainting ([Chaitanya et al. 2017; Bako et al. 2017; Vogels et al. 2018; Liu et al. 2018]) can thus be potentially applied to address our problem. However, our
network is applied on a large number (thousands) of voxels, while our end goal is to speed up the total rendering process. Therefore, we balance the inference speed and reconstruction quality in our network design.

We propose to use a compact U-Net [Ronneberger et al. 2015] style neural network with residual links and skip connections to achieve the sampling map reconstruction as illustrated in Fig. 2. Our network uses multiple downsampling and upsampling convolutional layers to extract meaningful neural features from the input sampling map $S_d$ and regress a high-quality sampling map $\hat{S}_d$. Our input raw sampling maps are computed from sparse photons, which contain many holes or empty bins as shown in Fig. 3. Therefore, we use the light-weight masked convolutions as the convolutional layers in our network, inspired by the recent image inpainting works [Liu et al. 2018; Yi et al. 2020]. This ensures that valid (non-empty) and invalid (empty) solid angle bins are treated differently in the network and only valid bins can contribute in a convolutional operation. Note that, our network is relatively compact, compared to the previous U-Net-like networks ([Chaitanya et al. 2017; Bako et al. 2017; Vogels et al. 2018]) used in other tasks; the maximum number of feature channels in our network is only 64. This compactness allows for fast sampling map reconstruction during path tracing on high-end GPUs, keeping our network from becoming the bottleneck in the entire rendering process. In fact, a large network is not very necessary for our task, since a sampling map has only a single channel (no color variations) and we only need to reconstruct low-resolution maps (just $32 \times 64$ or $64 \times 128$ that are much lower than other reconstruction tasks), which are already adequate for high-quality rendering. While compact, our network can regress high-quality maps that enable efficient path guiding in path tracing with quickly reduced variances. We believe our network size can be further reduced by advanced network compression techniques [Cheng et al. 2018; Deng et al. 2020] that can enable even more efficient path guiding, and we leave this as future work.

We also find that the usage of photon counts and the previous sampling map as buffers is effective; these auxiliary features are simple to obtain but useful indicators for the quality and evolution of the original per-solid-angle probabilities. These buffers are also compact and improve the reconstruction quality with marginal extra cost. On the other hand, we find that other geometric features, such as position, normal, and depth — that are used in previous screen-space guiding methods [Bako et al. 2019; Huo et al. 2020] — are not very helpful in most of our scenes since our reconstruction operates on each local 3D voxel. To justify our neural network design, we compare the performance with a standard U-Net [Ronneberger et al. 2015] in the supplementary material.

6.2 Loss function

We utilize an $L_1$ loss to supervise the output sampling map:

$$L_S = |\hat{S}_d - S_d| \quad (11)$$

where $\hat{S}_d$ is the ground-truth sampling map computed by tracing a large number of photons. Inspired by the deep supervision in machine learning [Xie and Tu 2015; Lee et al. 2015], we also provide the ground-truth signal to every decoding level in order to ease the loss backpropagation. To avoid potential over-blurring, we leverage an asymmetric function inspired by [Vogels et al. 2018]; this leads to our full loss

$$L_{\text{rec}} = L_S \cdot (1 + (\lambda - 1) \cdot E) \quad (12)$$

**Algorithm 1:** Our neural path guiding framework in Sec. 7. Through multiple iterations of path tracing and photon tracing, we construct a hierarchical grid (Sec. 7.3), reconstruct and update the sampling map in each valid grid voxel (Sec. 7.2), and guide the path tracing using the sampling maps (Sec. 7.3). We also apply a final path tracing pass guided by the reconstructed sampling maps (Sec. 7.4). We use different colors to mark different subsections, with green for Sec. 7.1, blue for Sec. 7.2, red for Sec. 7.3 and purple for Sec. 7.4.

**Input:** Target scene, pre-trained neural network $\mathcal{F}$

**Output:** A rendered image

```plaintext
Initialize a regular spatial grid; set all $Q_j = 0$
for each iteration $t < T$
    Initialize $z^t$ SPP path samples:
        for each path do
            for each bounce $b$
                Locate voxel $j (x_p \in \Delta A_j)$
                if not isValid$(j) \text{ (no sampling map)}$ then
                    Sample$(p_{\text{BSDF}}) \rightarrow \omega_b$
                else
                    Sample$(p_{\text{MIS}}) \rightarrow \omega_b$ (Eqn. 14);
                end
                markValid$(j)$
            Compute path throughput and $L(x_p, \omega_b)$:
                for each bounce at $x_p, \omega_b$
                    if isValid$(j)$ then
                        $\nu_b = L(x_p, \omega_b) \cos \theta_{b} f(x, \omega_p, \omega_b)$
                        if $\omega_b \rightarrow p_{\text{guide}}$ then
                            $\nu_{j, b} \leftarrow \nu_b$, else $\nu_{j, b} \leftarrow 0$
                        if $\omega_b \rightarrow p_{\text{MIS}}$ then
                            $Q_{j, b} \leftarrow 1$ else $Q_{j, b} \leftarrow 0$
                            $Q_{j, b} \leftarrow 1$
                            if $Q_{j, b} \geq 50$ & $Q_{j, b} \geq 50$ then Update $\sigma_f$ (Eqn. 13);
                        end
                    end
                    Update the output image
                end
            Trace $z^t N_p$ light paths for photons:
                for each photon $p$
                    Locate voxel $j$, solid angle $k (x_p \in \Delta A_j, \omega_p \in \Delta \Omega_k)$
                    if isValid$(j)$ then
                        Update energy map: $\Phi_{j, k} \leftarrow \Delta \Phi_p$ (Eqn. 9);
                        $M_p \leftarrow 1$; Update $V_p$
                        if $M_j > \Phi_p$ or $V_n > \Phi_{\text{thr}}$ then
                            Subdivide voxel $j$ into two sub-voxels (Sec. 7.1);
                        end
                    end
                    for each valid voxel $j$
                        if not isValid$(j)$ then
                            $\nu_{j, b} \leftarrow 0$
                        else
                            $\nu_{j, b} \leftarrow \nu_{j, b}$
                        end
                    end
                end
            Trace $N_f$ paths for final output (Sec. 7.4):
        end
    end
end
```
where $\mathbb{1}_0$ if the output and the input values are both larger or smaller than the ground-truth value and $\mathbb{1}_1$ if they are not on the same side. More specifically, when there are two equally-good output values, this function prefers the one that is closer to the input and penalizes the other that diverges too much from the input. We find that $\lambda = 1.5 \sim 2.5$ leads to reasonable output sampling maps with sufficient details.

We also find that an adversarial loss used in previous work [Yu et al. 2019; Yi et al. 2020; Bako et al. 2019] offers only slight improvements on recovering details in the sampling map and is not very helpful for final rendering in most cases, so we avoid using adversarial losses for simplicity.

6.3 Discussion

Our network focuses on reconstructing high-quality sampling functions for local path sampling. This is a central sub-problem in many path guiding frameworks. Note that, this problem of sampling map regression is independent of other sub-modules in path guiding. We train our network independently without relying on any works (like Jensen 1995; Vorba et al. 2014; Müller et al. 2017; Rath et al. 2019) where $H$ (Line 30) with adaptive spatial partitioning which iteratively builds various sizes in training scenes, and compute sampling maps with both sparse and dense photons to obtain many training pairs (please refer to Sec. 8 for details of data generation and training).

Note that, our learning-based sampling map reconstruction module can potentially be applied in many existing path-guiding frameworks (like Jensen 1995; Vorba et al. 2014; Müller et al. 2017; Rath et al. 2020), and improves the traditional sampling distribution reconstruction modules. In this work, we present a new framework (Sec. 7) with adaptive spatial partitioning which iteratively builds sampling maps using our neural network in a hierarchical grid for path guiding.

7 NEURAL PATH GUIDING USING A HIERARCHICAL GRID

In this section, we introduce our novel path guiding framework that leverages our presented deep network to reconstruct high-quality sampling maps in a hierarchical grid. Our full framework is illustrated in Algorithm 1. As shown in Algorithm 1, we first initialize a grid (Line 1) and then utilize an iterative process (Line 2−39) to build a hierarchical grid with per-voxel sampling maps for path guiding and rendering. In each iteration, we trace camera paths (Line 3−24); these paths can be guided (Line 7−11) when tracing, and they are used to detect valid voxels (Line 12) and compute the mixture weight of one-sample MIS (Line 17−20). We also trace photons (Line 25−35) per iteration; in each valid voxel, we accumulate photon energy (Line 29) that is required by our network and also collect other photon statistics for subdividing the hierarchical grid (Line 30−33). We then reconstruct the sampling map in each valid voxel using our pre-trained deep neural network at the end of each iteration (Line 36−38); these sampling maps are used to guide the following path tracing. After the iterative process, we apply a final path tracing to compute the final output image (Line 40).

Essentially, we iteratively trace camera paths and photons for adaptively partitioning the scene space to a hierarchical grid (see Sec. 7.1 and green blocks in Algorithm 1). Meanwhile, the photon samples are also used for computing the sampling maps in each voxel (see Sec. 7.2 and blue blocks in Algorithm 1) to guide the tracing of the paths in the following iterations; the path samples are also used for rendering and computing the weight $\alpha$ for one-sample MIS (see Sec. 7.3 and red blocks in Algorithm 1). After a total number of $T$ iterations, we do a final path tracing pass (see Sec. 7.4 and the purple block in Algorithm 1) with $N_f$ spp to render the image. The final rendering result is computed from all path samples in the iterations (except for the first iteration that is not guided) and the final pass. Note that, we double the number of paths and photons after each iteration, so that both the quality of the input raw sampling maps and final rendering can be progressively improved; this leads to $2^t N_p$ spp paths and $2^t N_p$ photon rays for iteration $t$, where $N_c$ is the initial spp and $N_p$ is the initial number of photon rays in the first iteration.

7.1 An adaptive hierarchical grid for path guiding

Since a pure uniform spatial structure (often achieved by spatial cache points in early works [Jensen 1995; Vorba et al. 2014]) is very expensive and impractical for large-scale scenes, recent works often utilize a KD-Tree [Müller et al. 2017] to adaptively partition the space, starting from a single root node that covers the entire scene. This coarse-to-fine spatial structure is effective, and, in fact, also necessary for many pure online-learning approaches [Guo et al. 2018; Rath et al. 2020], since they need to use a large number of (path) samples that can be only acquired in a large spatial region at an early stage. In contrast, our deep learning based approach can reconstruct a high-quality sampling map from a sparse set of photons; consequently, starting from a highly coarse spatial partitioning is unnecessary and also even inefficient for our approach. Therefore, we propose to use a hierarchical grid for spatial partitioning, which combines uniform and adaptive spatial partitioning (Fig. 4).
An initial regular grid. We start from a regular grid, uniformly dividing the entire scene at a relatively coarse level (see the three coarse voxels in Fig. 4), as the initial spatial structure (Line 1 in Algorithm 1); the initial grid is still coarse but relatively much denser than a shallow KD-Tree used in early stages in previous work [Müller et al. 2017]. This regular grid enables reconstructing more locally representative sampling maps, leading to good path guiding quality even at the first iteration in our framework, which fully utilizes the benefits of our offline trained deep neural network. Starting from this regular grid, we iteratively sub-partition each grid voxel into a local KD-tree (Line 30–33 in Algorithm 1), leveraging the statistical information of per-iteration paths and photons; this results in a hierarchical grid that adaptively covers the scene and we reconstruct the sampling maps per voxel in each iteration accordingly.

Detecting valid voxels using paths. While we can compute a sampling map for every voxel in the grid for path guiding, this is usually costly and in fact unnecessary, since many voxels may not be reached by any paths from the viewpoint when rendering a large scene. Therefore, we leverage the per-iteration camera paths to detect which voxels are necessary for rendering this viewpoint (Line 12 in Algorithm 1). Specifically, when tracing the $2^\mathcal{N}_c$ spp path samples in each iteration, we mark a voxel (that hasn’t been marked before) as a new valid voxel, if there is at least one bounce point of the paths located in the voxel (see the two valid voxels in Fig. 4). In other words, we only consider a voxel for sampling map reconstruction and further spatial partitioning when it is known to be necessary (at least likely necessary) in rendering in the following iterations. This avoids the waste of reconstructing many unnecessary sampling maps and local sub-KD-trees. Once a voxel is marked as valid, we start accumulating photons in the voxel for sampling map reconstruction and further subdivision of the voxel.

Voxel subdivision. It is not efficient to use a regular grid for spatial partitioning, since various local spatial regions may involve highly diverse geometry, appearance, and lighting distributions. Therefore, we iteratively subdivide the initial regular grid into a hierarchical grid, where a voxel is divided into a binary tree similar to a local KD-tree if necessary (Line 30–33 in Algorithm 1). Our hierarchical grid is built to adapt to the complexity of local geometry and incident light fields. In the very beginning iterations, we trace small numbers of light paths and photon data is temporarily stored in each voxel. We leverage the statistics of accumulated photons in the current iteration in each valid voxel for the voxel’s possible subdivision. In particular, for each valid voxel $j$, we consider $M_j$ – the total number of photons hitting the voxel through iterations – and $V_{nj}$ – the variance of the surface normals at the photon hitpoints. A voxel is split into two sub-voxels by the middle of photon positions along an axis (just like KD-tree construction), if $M_j > M_{thr}$ or $V_{nj} > V_{nthr}$, where $M_{thr}$ and $V_{nthr}$ are two predefined thresholds and we recursively apply our subdivision criterion to sub-voxels (see the right voxel in Fig. 4). Once a voxel is subdivided, its two sub-voxels are kept as valid, accumulating photons from the current iteration and waiting for photons in following iterations to reconstruct sampling maps. These simple photon statistics are easy to compute, enabling efficient subdivision. This photon-based subdivision process subdivides voxels that either have complex light fields (dense photons) or complex geometry (large normal variation). Our method allows these complex voxels to utilize more local and accurate sampling maps in the following iterations, thus leading to more accurate renderings.

7.2 Sampling map reconstruction
Apart from determining the subdivision in the hierarchical grid, the main goal of tracing the per-iteration photons is to reconstruct the per voxel sampling maps for path guiding. For any valid voxel (marked by camera paths), we accumulate photon energies to compute the energy map of the voxel (Line 29 in Algorithm 1), as is expressed by Eqn. 9. The energy map records the sum of the energies of all hitting photons $\Phi_P$ in the voxel through the current and all previous iterations. The per-pixel accumulated energy $\Phi_{ij}$ in an energy map will be normalized, which leads to a raw sampling map $S_j$, that is sent as input to the network to reconstruct the sampling map in iteration $t$. As discussed in Sec. 6, we also provide additional input buffers (photon count, previous raw sampling map, and binary mask) for the network. Specifically, we record the number of accumulated photons and also keep the raw sampling map and number of photons in the previous iteration to construct the network input. After tracing all photons in an iteration, we collect all valid voxels that have new photons arrived and reconstruct the sampling maps $S_j$ using our deep neural network for path guiding (Line 37 in Algorithm 1). As mentioned, we exponentially increase the photon count per iteration with a base of 2, similar to the growth of path samples by Müller et al. [2017], so that the number of photons consumed by the input sampling map is roughly doubled after each iteration. Once a sampling map is reconstructed at a voxel in one iteration, the map is used in the following iterations and the final path tracing pass to guide the path sampling in the voxel.

7.3 Path guiding and one-sample MIS
In any iteration, if a path hits a voxel that doesn’t have a sampling map, we just use standard BSDF sampling at the bounce point (Line 8 in Algorithm 1); such a voxel is usually still an invalid voxel, which will be marked as valid and start accumulating photons immediately in the same iteration, allowing for path guiding in the following iterations. On the other hand, once a path ray hits a valid voxel that has a reconstructed sampling map, path guiding can be achieved by doing importance sampling on the sampling map (where a CDF is built via a fast cumulative sum over pixels on GPUs, just like sampling an environment map). Since our sampling map only considers the incident radiance (and a cosine term), we apply a one-sample MIS similar to previous works to combine guided sampling and BSDF sampling (Line 10 in Algorithm 1), as discussed in Eqn. 4. The combined sampling strategy however requires a parameter $\alpha$ that determines how often either sample strategy is selected. Usually, $\alpha = 0.5$ is a simple choice and performs reasonably well in previous work [Müller et al. 2017]. An $\alpha$ that is learned via online optimization ([Müller 2019]) is also presented for better performance but requires expensive online training.
We present a heuristic $\alpha$ computation technique, based on path statistics (Line 17–20 in Algorithm 1), though simple, it results in effective per-voxel $\alpha_j$ in practice for high-quality path guiding. In particular, we initially use $\alpha_j = 0.5$ in each voxel. While tracing paths in each iteration, we first construct all paths using one-simple MIS according to the current per-voxel mixture weights $\alpha_j$. And once a full path is constructed, we compute the actual sub-path contribution (often known as throughput, Line 17 in Algorithm 1) for every bounce point $b$ on the path as $v_b = L(x_b, \omega_b) \cos \theta_{b, f}(x, \omega_b, \omega_b)$, where $x_b$ is the position of the bounce point, $\omega_b$ is the sampled direction (that can come from either BSDF or guided sampling), $L(x_b, \omega_b)$ is computed by consecutively multiplying the light radiance, BSDFs, and invers sampling PDFs through all following bounce points as in a standard Monte Carlo path sample. Meanwhile, for each voxel $j$, we accumulate all bounce contributions $v_b$ (of the bounces that are in the voxel, i.e., $x_b \in \Delta A_j$) in $v_{j,B}$ and $v_{j,G}$, according to from which distribution $\omega_b$ is sampled (Line 18 in Algorithm 1). Specifically, $v_{j,B}$ records the sum of all path contributions $v_b$ if its direction $\omega_b$ is sampled by BSDF sampling, and $v_{j,G}$ records the sum of $v_b$ if $\omega_b$ is sampled by guided sampling. We also record the numbers of bounces sampled by the two sampling strategies as $Q_{j,B}$ and $Q_{j,G}$ (Line 19 in Algorithm 1) in each valid voxel. Once $Q_{j,B} \geq 50$ and $Q_{j,G} \geq 50$ sub-paths are sampled in a valid voxel $j$ (Line 20 in Algorithm 1), we use the ratio of the averaged $v_{j,B}$ and $v_{j,G}$ to determine the mixing weight $\alpha_j$ for following path guiding:

$$\alpha_j = \frac{v_{j,B}}{v_{j,B} + v_{j,G}}.$$  

where $v_{j,B} = v_{j,B}/Q_{j,B}$ and $v_{j,G} = v_{j,G}/Q_{j,G}$. Correspondingly, our one-sample MIS is expressed by:

$$p_{\text{MIS}}(\omega_i) = \frac{v_{j,B}}{v_{j,B} + v_{j,G}} p_{\text{BSDF}}(\omega_i) + \frac{v_{j,G}}{v_{j,B} + v_{j,G}} p_{\text{guide}}(\omega_i).$$  

We set $\alpha_j = 1.0$ if BSDF is a delta function and clamp $\alpha_j$ between 0.2 and 0.8 otherwise. This heuristic mixing weight considers the data that reflects the actual performance of BSDF sampling and guiding sampling, leading to effective one-sample MIS sampling in our path guiding.

7.4 Rendering and final path tracing

Our learning based approach is able to reconstruct high-quality sampling maps from very sparse photons, leading to efficient guided path tracing even in early iterations. The first iteration paths are not guided at all since there are no sampling maps reconstructed yet. However, thanks to our deep neural network, our path guiding is often of very good quality starting from the second iteration. We therefore leverage all path sampling starting from the second iteration for rendering the final image.

While we can keep iteratively tracing more photons and refining our sampling maps, we find that our reconstructed sampling maps are often of very high quality after $T = 4$–9 iterations. As a result, continuing tracing more photons afterwards merely leads to marginal sampling improvement. Therefore, we choose to stop tracing photons after $T = 4$–9 iterations, fix the per-voxel sampling maps, and switch to do pure path tracing guided by the fixed sampling maps using a number of $N_f$ spp as needed. This is called the final path tracing pass in our framework (Line 40 in Algorithm 1). Our final rendered image is computed from all path samples traced in all $T$ iterations and the final path tracing pass.

8 IMPLEMENTATION

Dataset generation and neural network training. We create a large scale dataset to train our sampling map reconstruction network. Our dataset consists of both designed scenes and auto-generated scenes as shown in Fig. 5. We first collect available online scenes designed by researchers or artists, by collecting several released scenes from previous work and purchasing scenes from several websites [Bitterli 2016; Jakob 2010; Evermotion 2012; Trader 2020; Squid 2020; Blend Swap 2016]. This leads to 32 designed scenes in total, including multiple realistic indoor and outdoor scenes; we use 20 of them in our training set and the rest for testing our algorithm. To enhance the generalizability of our network, we further enlarge our training set by auto-generating many more scenes. In particular, we procedurally generate 500 scenes using randomized shape primitives, materials, and area lights, similar to [Zhu et al. 2020; Xu et al. 2018]. We also leverage a complex lighting dataset [Gardner et al. 2017] and randomly select an environment map for each generated scene as its additional illumination. This auto-generation process largely increases the diversity and complexity of our training scenes, leading to better generalization on novel testing scenes.

We reconstruct sampling maps with the same resolution of $128 \times 64$. As expected, if memory allows, a higher resolution of sampling maps often leads to better rendering quality. While our rendering quality degrades with a lower resolution, we find that, even using $64 \times 32$ sampling maps, our method can still outperform previous state-of-the-art methods (see Fig. 10). Our network aims to reconstruct a sampling map of a local 3D voxel. We partition the space of each training scene uniformly using a regular grid with a random resolution ranging from $50^3$ to $200^3$. This makes our network generalize to various voxel sizes, naturally enabling high-quality sampling map reconstruction for any voxel at any depth in a hierarchical grid. To further augment the data, we also randomly rotate...
the world coordinate frame when partitioning. We trace photons in each scene and compute sampling maps based on Eqn. 9 using both sparse and dense photons, which constructs the input and ground-truth training pairs. The total number of training pairs in our dataset is about 10.5 million. To make the network generalize well on different iterations in our path guiding, for each training scene, we randomly select an iteration number \( t \) from 1 to 12, and compute the corresponding input sampling map using the photons generated by the \( 2^t N_p \) light paths. On the other hand, the ground-truth sampling map of each voxel is computed by accumulating photons generated through 20 iterations for each scene.

During rendering, the number of photons in different voxels can be highly different (from several to several thousand), leading to highly diverse input distributions; we therefore train multiple networks separately and the correspondingly trace the same number of light paths (\( N_p \)) for each training scene. For voxel subdivision, we use an iteration-dependent threshold for the photon count, given by \( M_{\text{thr}} = c \cdot \sqrt{2^t} \) similar to [Müller et al. 2017], where \( t \) is the iteration number and \( c \) is a scalar parameter. We find \( c \approx 400 \) to 800 performs similarly in practice, and we use \( c = 500 \) for all our testing experiments. The normal variance threshold is set to \( V_{\text{thr}} = 0.5 \). We also set the maximum depth of a local KD-tree to 8, which already corresponds to a very fine grid and avoids unnecessarily detailed subdivision. In practice, we also only proceed to voxel subdivision in the first two iterations, which results in a reasonable hierarchical grid.

We use a high-end machine with Intel Core i9-7960X CPU and Nvidia Titan RTX GPUs for rendering our testing scenes. Our framework is implemented in the standard rendering engine Mitsuba [Jakob 2010], and we use the PyTorch C++ API [Paszke et al. 2019] at rendering time for sampling map reconstruction on GPUs. This Mitsuba based implementation ensures a fair comparison with previous methods, most of which are also implemented with Mitsuba. In particular, we only use GPUs to do network inference for sampling map reconstruction, while all other parts of the algorithm (including path tracing, photon tracing, ray sampling, radiance computation, spatial grid construction, etc.) are done on the CPU as in the standard Mitsuba renderer. The CPU and the GPU parts are run in a sequence in our implementation. We believe this is a fair enough setting when comparing with traditional pure CPU-implemented path guiding methods that do not use neural networks. In fact, our GPU computation time is only about 10% of the total running time; please refer to Tab. 1 for detailed running times for each of our testing scenes. In the future, a more efficient implementation in practice can be done by making the GPU part run in parallel with the CPU part or even implementing a pure GPU-based framework leveraging hardware ray tracing in modern GPUs [Parker et al. 2010].

### 9 Evaluation

We now present extensive experiments to evaluate our path guiding approach. We first evaluate the rendering quality of our method by comparing against various state-of-the-art path guiding methods quantitatively and qualitatively. We then investigate sub-components in our system to justify their effectiveness. Some additional evaluation results can be found in the supplementary material.

#### Configuration

We evaluate our method comprehensively on 12 realistic testing scenes; the corresponding images of these scenes can be found in Fig. 6, 7, 8, 10 and 11. These testing scenes include challenging indoor and outdoor cases with complex global illumination, covering a wide range of scene complexity and diversity. Each scene contains both direct and indirect illumination. For indoor scenes with outside environment map illumination, we provide the window geometry for sampling light paths from the environment.

| Algorithm 1 Path (%) | Photon (%) | Neural rec (%) | Path (%) | Time (min) |
|---------------------|------------|----------------|----------|------------|
| Algorithm 1         | LN3-24     | LN25-35        | LN36-38  | LN40       | /           |
| Device              | CPU        | CPU            | GPU      | CPU        | /           |
| Phase               | iterative  | process when \( t < T \) | final    | /           |
| CAUSTICS EGG        | 13.91      | 21.83          | 8.36     | 55.88      | 4.0         |
| VÆCH AJAR           | 14.58      | 21.08          | 5.76     | 58.56      | 18.0        |
| BATHROOM            | 15.42      | 11.86          | 9.77     | 62.93      | 5.0         |
| HOTEL               | 15.05      | 18.58          | 5.89     | 60.46      | 20.0        |
| STAIRCASE           | 15.79      | 15.77          | 5.01     | 63.41      | 11.0        |
| LIVING ROOM         | 16.42      | 11.73          | 5.89     | 65.94      | 11.0        |
| SPACESHIP           | 16.49      | 9.20           | 8.06     | 66.23      | 3.0         |
| CLASSROOM           | 15.33      | 16.68          | 6.39     | 61.57      | 13.0        |
| WILD CREEK          | 17.05      | 8.41           | 6.02     | 68.50      | 10.0        |
| TORUS               | 15.72      | 12.78          | 8.32     | 63.16      | 4.0         |
| KITCHEN             | 14.35      | 19.06          | 8.94     | 57.63      | 4.0         |
| POOL                | 16.78      | 8.22           | 7.57     | 67.40      | 4.0         |

Table 1. Running time. Percentages of running time of different components in the proposed system are shown in the table for different testing scenes. The total rendering time for each scene is also shown in the rightmost column. The time distribution varies depending on the scene complexity and light setup.

Path guiding details. We use \( N_c = 1 \) (spp) for all our experiments, leading to \( 2^t N_c = 2^t \) spp paths for iteration \( t \). We also correspondingly trace the same number of light paths (\( N_p \)) thus equals to the number of pixels) per iteration for distributing photons. The initial regular grid is implemented as a hash grid that can be accessed in \( O(1) \) time. Each sub binary tree is like a local KD-tree that can be accessed in \( O((\log(n)) \) time. Our final hierarchical grid is a hybrid spatial structure and can thus be quickly accessed at rendering time, enabling highly efficient path guiding. Since our spatial structure is adaptively constructed, our method is not very sensitive to the resolution of the initial grid, and we use a resolution of \( 100^3 \) for all the testing scenes. For voxel subdivision, we use an iteration-dependent threshold for the photon count, given by \( M_{\text{thr}} = c \cdot \sqrt{2^t} \) similar to [Müller et al. 2017], where \( t \) is the iteration number and \( c \) is a scalar parameter. We find \( c \approx 400 \) to 800 performs similarly in practice, and we use \( c = 500 \) for all our testing experiments. The normal variance threshold is set to \( V_{\text{thr}} = 0.5 \). We also set the maximum depth of a local KD-tree to 8, which already corresponds to a very fine grid and avoids unnecessarily detailed subdivision. In practice, we also only proceed to voxel subdivision in the first two iterations, which results in a reasonable hierarchical grid.

ACM Trans. Graph., Vol. 0, No. 0, Article 0. Publication date: January 2021.
To better illustrate the effectiveness of path guiding, we turn off the Nearest-Event Estimation (NEE) for our and all comparison methods as map, facilitating the photon tracing process in these scenes. For our method, the required time to achieve good rendering quality ranges from 3 to 20 minutes (depending on scene complexity) on these testing scenes. We demonstrate equal-time comparisons by comparing with four state-of-the-art path guiding methods [Bako et al. 2019; Müller et al. 2017; Vorba et al. 2014; Rath et al. 2020] on all testing scenes; we also show the corresponding equal-quality rendering time on a few scenes. In the comparisons, we directly use the released source code of [Müller et al. 2017], [Rath et al. 2020], and [Vorba et al. 2014], which are all implemented with Mitsuba [Jakob 2010] that runs on CPU. Since there’s no publicly available source code of [Bako et al. 2019], we use our own implementation of it with Mitsuba for all experiments. As discussed in Sec. 8, we implement our method also in Mitsuba, mostly running on CPU for fair comparisons, while only the network inference for sampling map reconstruction runs on GPU, which only takes about 10% of the total running time (see Tab. 1 for detailed timing). Our implementation of [Bako et al. 2019] follows similar CPU and GPU separation, where we run their sampling map reconstruction network on GPUs and run other parts of the algorithm on CPU. All comparisons are run on the same machine with the same CPU and GPUs (if needed).

To better illustrate the effectiveness of path guiding, we turn off the Next Event Estimation (NEE) for our and all comparison methods as done in previous work [Vorba et al. 2014; Müller et al. 2017]. Comparison results with NEE turned on are shown in the supplementary material. The ground-truth images are rendered using path tracing with NEE for 2 to 6 days per scene.

Quantitative and qualitative evaluation. We now demonstrate the quantitative and qualitative results of our method and compare against other methods with equal rendering time. For quantitative evaluation, we use the relative Mean Squared Error (rMSE, as used in [Rath et al. 2020]) and the perceptually-based Structural Similarity Index (SSIM, as used in [Bako et al. 2019]) as metrics. Table 2 shows the quantitative results of rMSEs and SSIMs of the full images of all 12 testing scenes. The corresponding percentages of running time of sub-components are shown in Tab. 1. In most cases, the path and photon tracing on CPU take more than 90% of the entire system running time, and we only spend a small amount of time (10%) on requesting GPU resources for neural sampling map reconstruction. Our method achieves the best quantitative results with the lowest rMSEs and highest SSIMs on all testing images. Note that ours is able to lower the rMSEs of the best comparison methods by more than 50% in many challenging scenes like CAUSTICS EGG, VEACH AJAR, BATHROOM, HOTEL, STAIRCASE, LIVING ROOM, and TORUS. These results demonstrate the high effectiveness and efficiency of our method, which is significantly better than all comparison methods.

To illustrate the details of our results, we also show quantitative and qualitative comparisons on multiple crops of the rendered images in Fig. 1, 6, 7 and 8. Our results are of the highest visual quality in these figures, which can also be reflected by the lowest rMSEs of all the comparison image crops.

Note that the two unidirectional guiding methods [Müller et al. 2017; Rath et al. 2020] are usually the best two of all four comparison methods on these testing cases. They utilize an adaptive tree as their spatial partitioning, which is more efficient than the uniform cache points used in [Vorba et al. 2014], leading to much better rendering quality in most testing scenes despite the fact that [Vorba et al. 2014] is bidirectional. However, it can be highly challenging for unidirectional methods to discover high-energy paths, when a scene involves complex specular-diffuse interactions (like those in Fig. 6 that contain many reflective and refractive objects) or other strong global illumination effects (like in Fig. 8). Therefore, [Vorba et al. 2014] sometimes achieves better results than the unidirectional ones, like the results of SPACESHIP and LIVING ROOM, since it leverages photons from light paths that ease the process of light discovery.

In contrast, our approach also leverages an adaptive spatial structure and our novel hierarchical grid enables finer spatial partitioning.

### Table 2. Quantitative comparison

| Scene/Method          | PT [Bako et al. 2019] | [Vorba et al. 2014] | [Müller et al. 2017] | [Rath et al. 2020] | Ours   | PT [Bako et al. 2019] | [Vorba et al. 2014] | [Müller et al. 2017] | [Rath et al. 2020] | Ours   |
|-----------------------|-----------------------|---------------------|---------------------|-------------------|--------|-----------------------|---------------------|---------------------|-------------------|--------|
| Metric                | rMSE  | SSIM   | rMSE  | SSIM   | rMSE  | SSIM   | rMSE  | SSIM   | rMSE  | SSIM   |
| CAUSTICS EGG          | 0.3187 | 0.1353 | 0.0462 | 0.0311  | 0.0121 | 0.0052 | 0.1017 | 0.1824 | 0.3472 | 0.4581 | 0.7006 | 0.8242 |
| VEACH AJAR            | 0.3684 | 0.2585 | 0.0154 | 0.0073  | 0.0047 | 0.0011 | 0.0474 | 0.0898 | 0.4579 | 0.5455 | 0.6325 | 0.8572 |
| BATHROOM              | 0.0610 | 0.0403 | 0.0204 | 0.0249  | 0.0142 | 0.0050 | 0.4481 | 0.4725 | 0.5472 | 0.5260 | 0.5924 | 0.7427 |
| HOTEL                 | 0.4176 | 0.2607 | 0.2838 | 0.0812  | 0.0792 | 0.0276 | 0.0695 | 0.1155 | 0.0914 | 0.2665 | 0.2801 | 0.4378 |
| STAIRCASE             | 0.0176 | 0.0183 | 0.0110 | 0.0045  | 0.0038 | 0.0013 | 0.4810 | 0.4957 | 0.6513 | 0.7337 | 0.8626 | 0.8951 |
| LIVING ROOM           | 0.1928 | 0.1553 | 0.0255 | 0.0468  | 0.0416 | 0.0060 | 0.1360 | 0.1719 | 0.4734 | 0.2960 | 0.3327 | 0.6576 |
| SPACESHIP             | 0.2212 | 0.0914 | 0.0198 | 0.0716  | 0.0389 | 0.0137 | 0.5610 | 0.7476 | 0.8611 | 0.7452 | 0.8124 | 0.8793 |
| CLASSROOM             | 0.0733 | 0.0514 | 0.0124 | 0.0085  | 0.0038 | 0.0021 | 0.2789 | 0.3037 | 0.5756 | 0.6352 | 0.7681 | 0.8234 |
| WILD CREEK            | 0.1425 | 0.1100 | 0.0560 | 0.0618  | 0.0549 | 0.0382 | 0.3023 | 0.3734 | 0.4890 | 0.4852 | 0.5386 | 0.6222 |
| TORUS                 | 0.0511 | 0.0425 | 0.0150 | 0.0015  | 0.0015 | 0.0005 | 0.2610 | 0.6660 | 0.7864 | 0.9150 | 0.9300 | 0.9529 |
| KITCHEN               | 0.0644 | 0.0578 | 0.0249 | 0.0063  | 0.0035 | 0.0030 | 0.3898 | 0.4173 | 0.4655 | 0.6753 | 0.7873 | 0.8168 |
| POOL                  | 0.1175 | 0.0528 | 0.0026 | 0.0025  | 0.0016 | 0.0011 | 0.2264 | 0.4595 | 0.8551 | 0.8598 | 0.9364 | 0.9510 |
Fig. 6. Qualitative and quantitative comparison with equal rendering time. These scenes contain many transparent surfaces and involve complex specular-diffuse interactions; Photon-based methods have a natural advantage over path samples in this case. We show zoomed-in crops with rMSEs in the figure and compare with the results of [Vorba et al. 2014], [Müller et al. 2017], [Bako et al. 2019] and [Rath et al. 2020]. Corresponding equal rendering time for each scene is also listed. Our method achieves the best visual quality and the lowest rMSEs in these challenging cases.

Mean-while, our deep learning based method can reconstruct high-quality sampling maps from sparse photons; this enables high-quality path guiding in our finer spatial partitioning from the first through all iterations, avoiding the slow starting of those online learning methods and leading to highly efficient rendering. Our approach purely relies on photons to reconstruct sampling maps, which is effective in general and also highly efficient for challenging scenes that are dominated by indirect lighting. Thanks to our deep neural networks and our efficient spatial partitioning, our approach utilizes photons in a way that is much more efficient than previous work [Vorba et al. 2014]. Our photon-driven neural path guiding approach enables high-quality rendering results that are significantly better than all previous unidirectional and bidirectional guiding methods.

[Bako et al. 2019] is a recent deep learning approach that first leverages an offline trained network for unidirectional path guiding; yet their method can only guide the first bounces and leads to the worst results in most testing cases. As shown in their paper, this technique can be effective for lowering the initial severe MC noise with sparse path samples, especially on scenes with strong direct illumination. However, such a first-bounce technique is not very effective for scenes with strong indirect illumination; the benefits of its offline learning also become more limited through longer rendering, once other traditional multi-bounce techniques get enough path samples online. In contrast, our method is the first offline deep learning method that enables multi-bounce path guiding. Our approach takes full advantage of an offline trained network and successfully models the incident light field at any local regions in a scene, enabling significantly better rendering quality than [Bako et al. 2019] and all other traditional multi-bounce guiding techniques.

**Equal-quality comparison.** In addition to the equal-time comparison, we also compare the time spent to achieve the results of similar quality on some highly challenging scenes shown in Fig. 8; the corresponding rendering time (compared to our time) of each method is listed, for achieving the same rMSE (with a threshold of $10^{-4}$ in difference) of the full image as our method (corresponding to the rMSEs shown in Tab. 2). We can see that our method can significantly speed up the naive path tracing with the rendering speed that is several tens of times faster. Moreover, the fastest comparison
methods for these scenes still require at least two times the rendering time as our method does. Our approach significantly reduces the required amount of time to achieve realistic rendering.

**Sampling map reconstruction.** The core of our path guiding approach is our deep learning based sampling map reconstruction. We show examples of our reconstructed sampling maps, corresponding inputs and the ground-truth in Fig. 3; more examples are provided in the supplementary material. Note that our method can consistently improve the reconstruction quality through iterations. Even at the second iteration, when the input is extremely noisy, our network can still denoise the input and recover a full sampling map that has many details and is very close to the ground-truth. We also show additional comparison with using a simple U-Net for sampling map reconstruction in the supplementary material. This high-quality sampling map reconstruction allows for highly efficient path sampling when rendering.

To further justify the effectiveness of our network, we compare with only using the raw input sampling map (without the network reconstruction) for path guiding in Fig. 10. We also compare with a version that reconstructs sampling maps at a lower resolution of 64 × 32 (we use 128 × 64 by default as mentioned in Sec. 8). The results of [Müller et al. 2017] and [Rath et al. 2020] (which generally performs the best among all comparison methods as stated) are also shown in the figure to better understand the position of these versions of our method with reduced or degraded components. Note that, our method without the network can already achieve comparable rendering quality compared to previous methods in some cases. And for Pool, our method without network reconstruction can even perform better than [Müller et al. 2017]; this is because using photons is highly effective for such a scene, involving complex specular-diffuse interactions. This example clearly demonstrates the benefit of leveraging photons. The neural network in our framework can significantly improve the rendering quality achieved without the network. Our full model achieves the best visual quality and the lowest rMSE in these testing scenes. Also note that, while worse than our final model, our method with a lower resolution of sampling maps can already outperform the comparison methods and the one without the network. This demonstrates the high reconstruction quality of our network. This also shows that our method
Fig. 8. Equal-time and equal-quality comparison. Similar to Fig. 6 and 7, we do qualitative and quantitative equal-time comparisons on crops of the final renderings on these challenging scenes. Our method achieves better qualitative and quantitative results given equal rendering time. In addition, we also show equal-quality rendering time comparison. In particular, we list the corresponding rendering time (expressed by the scale to our time) of each method for achieving the same rMSE (that our method achieves in the equal-time comparison, shown in Tab. 2) of the full image. Note that, our method takes significantly less time; the fastest comparison method still requires more than two times the rendering time as our method for each scene.

Fig. 9. The effect of initial grid resolution and our hierarchical spatial partitioning framework. Ideally, the voxel size should be small enough to reflect the locality of the incident radiance, and large enough to contain enough photons for sampling map reconstruction. In the extreme case, under-partitioning and over-partitioning will both hurt the performance.

Hierarchical grid. We now investigate our presented spatial structure - the hierarchical grid. We show rMSEs of images rendered with different resolutions for the initial regular grid in Fig. 9. We also show corresponding results using only a regular grid without the adaptive partitioning inside voxels. Note that, without the adaptive partitioning, rendering quality varies drastically across different resolutions, since a low-resolution grid lacks expressibility of complex light fields in the scene and a high-resolution grid does not have enough photons in each voxel. On the contrary, our hierarchical grid is more stable with different resolutions, since it is able to adaptively subdivide the initial grid to a desired resolution locally. Our hierarchical grid also consistently enables better rendering quality than a regular grid at the same initial resolution.

Temporal stability. We also evaluate the temporal stability of our method. In particular, we use the DSSIM (i.e., dissimilarity as used in [Vogels et al. 2018]) between consecutive frames with a moving camera to express the temporal stability. Figure 11 shows the DSSIMs of our method and other comparison methods. Since
Photon-Driven Neural Path Guiding

10 CONCLUSION AND FUTURE WORK

In this paper, we present the first deep learning-based photon-driven path guiding approach. Our approach leverages photons to reconstruct sampling distributions, which is more effective than pure unidirectional (path-driven) methods for challenging scenes that are dominated by indirect lighting; we propose to use a deep neural network to process and reconstruct. However, it consumes more memory than the directional quad-tree used in [Müller et al. 2017]; the memory also limits the resolution of sampling maps we can reconstruct. Nonetheless, we show that our resolution of $128 \times 64$ can already achieve high-quality sampling, and even a lower resolution (like $64 \times 32$ as shown in Fig. 10) can also provide better results than previous methods. We leave extensions with a sparse directional representation in a learning framework as future work. Our approach leverages photons to reconstruct sampling maps. However, tracing photons can sometimes be highly inefficient; for example, if a camera is looking at only a small region of a large scene, there can be a large number of photons that are traced but never reach any valid voxels, leading to very expensive photon tracing. Leveraging bidirectional guiding techniques like [Vorba et al. 2014] to also guide the photon tracing process can potentially resolve this. Please refer to our supplementary material for an initial extension of our method with photon guiding. Finally, we currently use CPU for rendering and GPUs for neural reconstruction. Although we overlap data transfers with computation to reduce the synchronization latency, integrating our proposed framework to a GPU-based renderer like Nvidia OptiX [Parker et al. 2010] may be a better choice to accelerate the whole system.

Fig. 10. We study the effectiveness of the neural reconstruction module. We compare our full model with a version without neural sampling map reconstruction and a version that uses a lower resolution ($64 \times 32$) of sampling maps. We also compare with [Müller et al. 2017] and [Rath et al. 2020] on these results. We show crops with rMSEs of the rendered images for each method given equal rendering time. The corresponding equal-quality rendering time to achieve our full-image rMSE is also listed. Our final model achieves the best results among all comparison methods.

Fig. 11. Average DSSIM (lower is better) computed from 30 consecutive frames when the camera is moving fast along a direction. The dissimilarity is affected by both the content change as well as the noise level.

Our renderings are consistently better than those of other methods, our method also achieves the best temporal stability.

Limitations. We use a regular grid to represent a sampling distribution as a standard 2D map (image). This is easy for a deep neural network to process and reconstruct. However, it consumes more memory than the directional quad-tree used in [Müller et al. 2017]; the memory also limits the resolution of sampling maps we can reconstruct. Nonetheless, we show that our resolution of $128 \times 64$ can already achieve high-quality sampling, and even a lower resolution (like $64 \times 32$ as shown in Fig. 10) can also provide better results than previous methods. We leave extensions with a sparse directional representation in a learning framework as future work. Our approach leverages photons to reconstruct sampling maps. However, tracing photons can sometimes be highly inefficient; for example, if a camera is looking at only a small region of a large scene, there can be a large number of photons that are traced but never reach any valid voxels, leading to very expensive photon tracing. Leveraging bidirectional guiding techniques like [Vorba et al. 2014] to also guide the photon tracing process can potentially resolve this. Please refer to our supplementary material for an initial extension of our method with photon guiding. Finally, we currently use CPU for rendering and GPUs for neural reconstruction. Although we overlap data transfers with computation to reduce the synchronization latency, integrating our proposed framework to a GPU-based renderer like Nvidia OptiX [Parker et al. 2010] may be a better choice to accelerate the whole system.
We believe our method takes an important step towards making
whereas previous related techniques either train an online network
heuristic criteria to achieve voxel subdivision in the hierarchical grid;
proper reward function could provide more benefits. We leverage
learning techniques [Huo et al. 2020] to achieve sampling with a
learning based local sampling reconstruction with reinforcement
one-sample MIS) for better sampling efficiency. Combining our deep
extended to some advanced distributions like variance-aware [Rath
In addition, our target sampling density function can be potentially
is to also consider some global context and even achieve global
statistics for sampling map reconstruction; an interesting extension
appealing future directions. Our approach leverages local photon
the neural path guiding more practical, thus also opening up many
direction to explore.

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

11 SUPPLEMENTARY MATERIAL
In this supplementary material, we provide additional experimental
results and sampling map visualizations, as well as some discussions
about potential extensions of our proposed framework. Although
not being emphasized in the main paper, these additional studies and
evaluations are also very important in the design of a full-fledged
path guiding system in practice.

11.1 Monte-Carlo Denoising
Monte-Carlo (MC) rendering algorithms like path tracing are known
to suffer from the slow convergence problem when producing noise-
free images [Kajiya 1986; Lafortune 1996]. In recent years MC de-
noising has become a very successful approach to reduce pixel
variance, especially those based on neural networks [Bako et al.

Fig. 12. Monte-Carlo denoising on the path guiding rendering results. We use the default deep learning based denoiser in Nvidia OptiX 6.5. In general, the
denoiser fills the holes in between pixels and filters out the high-frequency MC noise. The denoised images rendered with our method are more acceptable
without severe blur or distortion.

| Scene       | Path tracer | Müller et al. | Rath et al. | Ours | Reference |
|-------------|-------------|---------------|-------------|------|-----------|
| Egg         | 0.3850      | 0.0674        | 0.0432      | 0.0168 |
| Space-ship  | 0.3109      | 0.0308        | 0.0098      | 0.0003 |
| Hotel       | 0.0023      | 0.0076        | 0.0014      | 0.0002 |
| Full-img    | 0.2212      | 0.0176        | 0.0039      | 0.0001 |
| Full-img    | 0.4040      | 0.0861        | 0.0198      | 0.0032 |
| Full-img    | 0.0141      | 0.0093        | 0.0087      | 0.0031 |
| Full-img    | 0.4176      | 0.0812        | 0.0702      | 0.0256 |
| Full-img    | 0.0104      | 0.0073        | 0.0042      | 0.0012 |

only sparse photons. To fully utilize the benefits of our network, we
introduce an adaptive hierarchical grid to cache our reconstructed
sampling maps across the scene, allowing for efficient path guiding
at any spatial location. We demonstrate that our method achieves
significantly better quantitative and qualitative results than various
previous state-of-the-art path guiding methods on diverse challen-
ging scenes.

Our method is the first neural path guiding method that uses
an offline trained network and supports guiding at any bounces,
whereas previous related techniques either train an online network
[Müller et al. 2019] or only guide the first bounce [Bako et al. 2019].
We believe our method takes an important step towards making
the neural path guiding more practical, thus also opening up many
appealing future directions. Our approach leverages local photon
statistics for sampling map reconstruction; an interesting extension
is to also consider some global context and even achieve global
guiding in primary space (like [Müller et al. 2019; Guo et al. 2018]).
In addition, our target sampling density function can be potentially
extended to some advanced distributions like variance-aware [Rath
et al. 2020] or product sampling [Herholz et al. 2016] (avoiding the
one-sample MIS) for better sampling efficiency. Combining our deep
learning based local sampling reconstruction with reinforcement
learning techniques [Huo et al. 2020] to achieve sampling with a
proper reward function could provide more benefits. We leverage
heuristic criteria to achieve voxel subdivision in the hierarchical grid;
this spatial partitioning process could also be potentially learned
via a deep neural network in the future. While we purely leverage
photons in our method, combining photons and path samples in
a holistic neural path guiding framework is an interesting future
direction to explore.
Fig. 13. The effect of next event estimation (NEE) on the final rendering results. The comparison is equal-time for each row. When turning on the NEE, the rendering time increases since we keep a similar total sample count. Results show that NEE greatly improves the results in some cases while it is not very useful in some other cases, depending on the sampling map quality in path guiding as well as the levels of light visibility at different scene locations.

11.2 Next Event Estimation

In the default experimental setting, we turn off the next event estimation (NEE) to clearly compare the effects from path guiding (similar to [Müller et al. 2017] and [Vorba et al. 2014]), though in practice NEE can be effective on some cases for all comparison methods. In particular, NEE can help reduce the variance by easing the search of a light source and improving the sampling map quality. To study how NEE affects the results, we turn on the NEE and keep the sample count unchanged on multiple test scenes. Results in Fig. 13 show that whether NEE is useful or not depends mostly on the light setup. For the BATHROOM scene, the glass bulb fixture and staggered window blinds make the direct connection very hard to

...
succeed; for the Veach Ajar and Living Room scenes, NEE fails and succeeds from time to time depending on the local light visibility. As a consequence, the rendering time greatly increases for all methods when NEE is turned on. In fact, our method achieves the best performance whether NEE is enabled or not, thanks to the high-quality reconstructed sampling maps that can capture both direct and indirect illumination. We believe the decision to request NEE or not is highly related to the total timing budget in specific applications.

### 11.3 Neural Sampling Map Reconstruction

As we mentioned in Sec. 6 of the main paper, we design a neural network that effectively reconstructs high-quality sampling maps. To demonstrate the effectiveness of our proposed network architecture, loss function, and our multi-expert inference scheme (Sec. 8 of the main paper), we train a single standard U-Net [Ronneberger et al. 2015] with $l_1$ loss function and without the auxiliary features, and use this simplest network to reconstruct all the sampling maps without training multiple versions. In addition, we try a simple Gaussian filter denoising and choose the best result from a range of variances from 0.01 to 10. Figure 14 shows the error curve during neural network training, as well as a visual comparison on the Classroom scene. Although deep learning based results are both better than the one without applying neural reconstruction, our proposed neural networks can produce a more smooth and lower-noise image. As shown in the loss curve, the average error of our reconstructed sampling maps is also smaller. The traditional Gaussian filter gives

---

Fig. 15. Additional reconstructed sampling map visualization through learning iterations. The reconstructed sampling maps lead to better path space exploration at the beginning and more accurate representations of the incident radiance in the subsequent iterations.

Fig. 16. Our proposed heuristic one-sample MIS scheme performs better than the default mixture coefficient $\alpha = 0.5$ especially when BSDF importance sampling and guiding have very disparate contributions to the final image.
We leverage photons in our neural path guiding, which is very effective when photons are too sparse since there is not enough information for rebuilding the incident radiance distribution. In contrast, it is sometimes beneficial to guide traced photons into visually important regions.

much worse performance since it only adds the same level of blur to the entire sampling map. We believe our proposed neural network can be further compressed by the state-of-the-art network compression methods [Cheng et al. 2018; Deng et al. 2020] and improved by more advanced architectures in the future.

11.4 One-Sample MIS
In Sec. 7.3 of the main paper, we demonstrate a new heuristic pipeline for estimating mixture coefficient $\alpha$ in one-sample MIS of BSDF sampling and guiding. This is quite useful in some cases, as shown in Fig. 16. For example in the Poo! scene, the BSDF sampled directions from the floor often fail to find the light source and leave the scene permanently, leading to a small contribution to the final pixel color. In contrast, our heuristic encourages sending more guiding samples to the regions based on the statistics of previously traced path samples. And for very glossy surfaces such as the metal armrest in this scene, we send more BSDF samples since many guided directions can have very small or zero BSDF value. Although the proposed heuristic may be sub-optimal, it is straightforward to implement and does not introduce extra online optimization overhead. In the future, we believe our heuristic can provide a good starting point to initialize other methods that try to optimize $\alpha$ in path guiding [Müller 2019; Rath et al. 2020].

11.5 Guided Photon Tracing
We leverage photons in our neural path guiding, which is very effective for dominant indirect lighting. However, photons are only useful when they are visible to the camera and in some cases many wasted photons can be traced. In order to handle some special light transport cases where many photons are invisible, we add the guided photon tracing module to our system as a simple extension. Similarly to [Vorba et al. 2014], we reconstruct the importance sampling maps from the accumulated path samples in each voxel. These path samples are virtual particles (i.e., importons [Peter and Pietrek 1998]) containing a value describing with what factor an illumination at a certain location would contribute to the final image. Here, we use the same pre-trained network for such reconstruction. The reconstructed importance sampling maps are then used for guiding photon tracing in every learning iteration. We use the Kitchen scene as an example since many emitted photons from outside sunlight cannot land inside the room without guided photon tracing, unless they are explicitly programmed to do so by manually providing the location of windows as our default experimental setting. Figure 17 shows the reduced variance in regions that are indirectly illuminated by the white sunlight and comparable results on regions that are lit by the indoor orange-color lights. Apart from these special cases, it is not always necessary to add this extra module for most common lighting setups created by the lighting artists when most photons are visible to the camera. Besides, our neural network is trained to properly handle the input maps with multiple levels of sparsity so our system can work well as long as the photons are not too sparse.

11.6 Additional Sampling Map Visualization
Some additional sampling maps are visualized in Fig. 15. After pre-training on an offline dataset, our neural network can progressively reconstruct higher-quality sampling maps with more accumulated photon energies through iterations on new scenes. Unlike previous Monte-Carlo denoising networks [Chaitanya et al. 2017; Bako et al. 2017; Vogels et al. 2018] which only process the input image once and stop after the inference, our reconstruction is getting better and closer to the ground-truth sampling maps over time. More specifically, the network reconstructs blurrier sampling maps due to low confidence in the early iterations to encourage more exploration of the directional space for the following path tracing; in the later iterations the reconstructed sampling maps get sharper and there emerge more accurate details of the incident radiance distribution due to a higher level of confidence.

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 Rouxelle. 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.

Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. 2018. Model compression and acceleration for deep neural networks: The principles, progress, and challenges. IEEE Signal Processing Magazine 35, 1 (2018), 126–136.

Djörk-Arne Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and accurate deep network learning by exponential linear units (elus). arXiv preprint arXiv:1511.07289 (2015).

Lei Deng, Guoqi Li, Song Han, Luping Shi, and Yuan Xie. 2020. Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey. Proc. IEEE 108, 4 (2020), 485–532.

Stavros Diotlantis, 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 Evermorton. 2012. Evermorton 3d models.

Marc-André Gardner, Kalyan Sunkavalli, Ersin Yumer, Xiaohui Shen, Emiliano Gambaretto, Christian Gagné, and Jean-François Lalonde. 2017. Learning to predict indoor illumination from a single image. arXiv preprint arXiv:1704.00099 (2017).

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

Toshiya Hachisuka and Henrik Wann Jensen. 2009. Stochastic progressive photon mapping. In ACM SIGGRAPH Asia 2009 papers, 1–8.
Toshiya Hachisu, Shinya Ogaki, and Henrik Wann Jensen. 2008. Progressive photon mapping. In ACM SIGGRAPH Asia 2008 papers, 1–8.

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

Yuchi Huo, Rui Wang, Zuxiang Zheng, Huaxin Xu, Hujian 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.

Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. 2017. Globally and locally consistent image completion. ACM Transactions on Graphics (ToG) 36, 4 (2017), 1–14.

Robert A. Jacobs, Michael I Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. 1991. Adaptive mixtures of local experts. Neural computation 3, 1 (1991), 79–87.

Wened 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 Knaps and Matthias Zwickler. 2011. Progressive photon mapping: A probabilistic approach. ACM Transactions on Graphics (ToG) 30, 3 (2011), 25.

Eric Lafortune. 1996. Mathematical models and Monte Carlo algorithms for physically based rendering. Department of Computer Science, Faculty of Engineering, Katholieke Universiteit Leuven 20 (1996), 74–79.

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

Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. 2015. Deeply-supervised nets. In Artificial intelligence and statistics. 562–570.

Guilin Liu, Futsun A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. 2018. Image inpainting for irregular holes using partial convolutions. In Proceedings of the European Conference on Computer Vision (ECCV). 85–100.

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.

Steven G Parker, Sam Gross, Francisco Massa, Adam Lerer, Keith Morley, Austin Robison, Morgan McGuire, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Catanzaro, and Jaroslav Krivánek. 2020. Varience-Aware Mixtures for Path Guiding. ACM Transactions on Graphics (Proceedings of SIGGRAPH 2020) 39, 4 (July 2020), 151:1–151:12. https://doi.org/10.1145/3385659.3392441

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. Optimaly combining sampling techniques for Monte Carlo rendering. In Proceedings of the 22nd annual conference on Computer graphics and interactive techniques. 419–428.

Eric Veach and Leonidas J Guibas. 1997. Metropolis light transport. In Proceedings of the 24th annual conference on Computer graphics and interactive techniques. 65–76.

Thajj 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.

Jiri Vorba, Johannes Hanika, Thomas Müller, Jaroslav Krivánek, 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

Jiri Vorba, Ondrej Karlik, Martin Šík, Tobias Ritschel, and Jaroslav Krivánek. 2014. On line learning of parametric mixture models for light transport simulation. ACM Transactions on Graphics (TOG) 33, 4 (2014), 1–11.

Saining Xie and Zhuowen Tu. 2015. Holistically-nested edge detection. In Proceedings of the IEEE international conference on computer vision. 1395–1403.

Zixiang Xu, Kalyan Sunkavalli, Sumi Haslap, and Ravi Ramamoorthi. 2018. Deep image-based relighting from optimal sparse samples. ACM Transactions on Graphics (TOG) 37, 4 (2018), 126.

Zhi Yi, Qiang Tang, Shekoofeh Azizi, Daesik Jang, and Zhan Xu. 2020. Contextual Residual Aggregation for Ultra High-Resolution Image Inpainting. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 7508–7517.

Jiahu Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. 2019. Free-form image inpainting with gated convolution. In Proceedings of the IEEE International Conference on Computer Vision. 4471–4480.

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

Mark Paul, Thomas Kollig, and Alexander Keller. 2000. Metropolis light transport for participating media. In Rendering Techniques 2000. Springer, 11–22.

Ingmar Peter and Georg Pfister. 1998. Importance-driven construction of photon maps. In Eurographics Workshop on Rendering Techniques. Springer, 269–280.

Charles R. Qh, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017. PointNet: Deep learning on point sets for 3d classification and segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 652–660.

Alexender Rath, Pascal Grittmann, Sebastian Herholz, Petr Vêvoda, Philipp Slussalek, and Jaroslav Krivánek. 2020. Variance-Aware Path Guiding. ACM Transactions on Graphics (Proceedings of SIGGRAPH 2020) 39, 4 (July 2020), 151:1–151:12. https://doi.org/10.1145/3385659.3392441

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