Pacific Graphics 2019
C. Theobalt, J. Lee, and G. Wetzstein
(Guest Editors)

Volume 38 (2019), Number 7

Offline Deep Importance Sampling for Monte Carlo Path Tracing

Steve Bako[1], Mark Meyer[2], Tony DeRose[2], Pradeep Sen[1]

[1]University of California, Santa Barbara
[2]Pixar Animation Studios

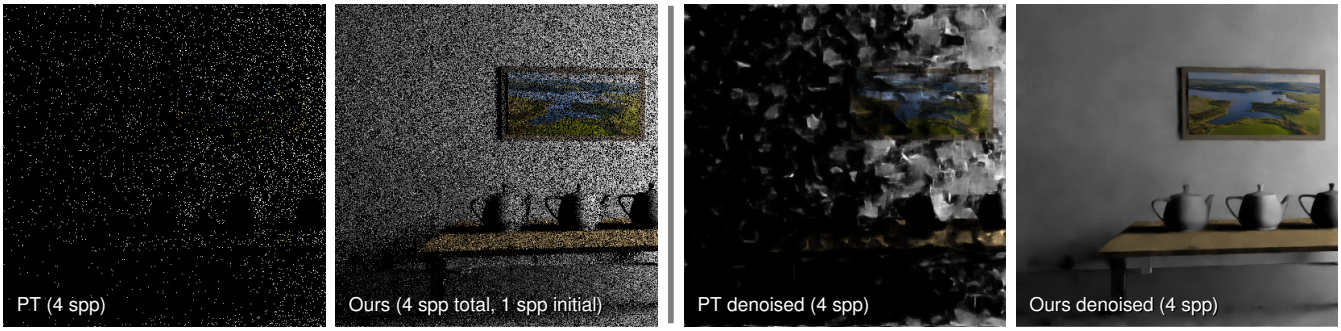

Figure 1: *We propose an offline, deep-learning approach to importance sample and improve convergence of path traced images, especially at low sample counts. The two leftmost images are results comparing standard path tracing (PT) to sampling using our neural network at 4 samples per pixel (spp) on a test scene in Mitsuba [Jak10]. Here, our network is given only an initial 1 spp as input to generate a sampling map that guides the remaining 3 spp. Unlike existing learning-based path-guiding methods, our network is used only in inference mode for test scenes and does not require any retraining or optimization per scene. Our system focuses on faster convergence for low sample counts in order to improve performance of post-process applications, such as Monte Carlo (MC) denoisers. On the right, we show how our method can be coupled with an off-the-shelf MC denoiser [BVM*17] to get better results.*

Abstract

Although modern path tracers are successfully being applied to many rendering applications, there is considerable interest to push them towards ever-decreasing sampling rates. As the sampling rate is substantially reduced, however, even Monte Carlo (MC) denoisers–which have been very successful at removing large amounts of noise–typically do not produce acceptable final results. As an orthogonal approach to this, we believe that good importance sampling of paths is critical for producing better-converged, path-traced images at low sample counts that can then, for example, be more effectively denoised. However, most recent importance-sampling techniques for guiding path tracing (an area known as "path guiding") involve expensive online (per-scene) training and offer benefits only at high sample counts. In this paper, we propose an offline, scene-independent deep-learning approach that can importance sample first-bounce light paths for general scenes without the need of the costly online training, and can start guiding path sampling with as little as 1 sample per pixel. Instead of learning to "overfit" to the sampling distribution of a specific scene like most previous work, our data-driven approach is trained a priori on a set of training scenes on how to use a local neighborhood of samples with additional feature information to reconstruct the full incident radiance at a point in the scene, which enables first-bounce importance sampling for new test scenes. Our solution is easy to integrate into existing rendering pipelines without the need for retraining, as we demonstrate by incorporating it into both the Blender/Cycles and Mitsuba path tracers. Finally, we show how our offline, deep importance sampler (ODIS) increases convergence at low sample counts and improves the results of an off-the-shelf denoiser relative to other state-of-the-art sampling techniques.

1. Introduction

Monte Carlo (MC) path tracing [Kaj86] can generate compelling images by simulating the physically-based light transport of a scene. However, producing noise-free results with the brute-force method requires substantial computation because many light paths (i.e., samples) must be evaluated to precisely estimate the light transport. Nevertheless, thanks to significant progress in variance reduction techniques, MC rendering has become increasingly commonplace in the past decade. For example, most of the high-end, film production pipelines now employ path tracing, and even real-time pipelines are moving towards physically-based rendering as modern games have begun to use raycasting with dedicated hardware. As a result, there is increasing demand to obtain these images faster and with much fewer samples than before.

Of all the variance reduction techniques proposed over the years, MC denoising [SD11, SD12, RMZ13, KBS15], in particular, has helped to fuel the recent, rapid adoption of path tracing. MC denoisers for both high-end production [BVM*17, VRM*18] and real-time [CKS*17, SKW*17, SPD18] rendering systems have demonstrated impressive results at low sampling rates for their respective applications (16 samples/pixel for production, 1-2 samples/pixel for real-time). Still, their results could substantially improve if they could be provided input images that are more converged than the ones they usually operate on. In other words, if we could speed up convergence, these denoisers could potentially obtain the same quality at even lower sampling rates or be able to denoise more complex scenes with the current number of samples.

One orthogonal approach for effectively improving image convergence is to importance sample paths during path tracing. Although the subject has received considerable attention over the years [VG95, LRR04, CJAMJ05, SA07], it is far from solved. Current state-of-the-art approaches are based on learning the sampling distribution directly in an online, per-scene fashion [VKv*14, DK17, MGN17, MMR*18, ZZ18]. Of these, the more powerful methods have tried to use deep networks to learn this distribution function [MMR*18, ZZ18], but this requires costly scene-dependent training time. Furthermore, these methods start learning the distribution from scratch for every new scene, so they require a significant number of samples and training time before they can guide future samples. Thus, they focus on convergence at high sample counts (e.g., hundreds or thousands of samples per pixel). For these reasons, none of the existing solutions were appropriate for our problem setting of extremely low sample counts.

In this paper, we address this problem by proposing an offline, deep learning framework for importance sampling in path space, called ODIS, that works even on low sample counts. Specifically, we wanted to design a system that could:

1. Generate an accurate sampling distribution for importance sampling from a small set of initial samples, and,
2. Start guiding future samples as quickly as possible.

To do this, we observe that the incident radiance in local regions of the scene is highly coherent. By gathering samples from these local regions, we can use them to estimate an accurate function for the incident radiance at any point in the scene, which can be converted to a distribution for guiding the next bounce in the path. For this, we leverage the power of deep learning to model the complex function that takes the local neighborhood of samples and reconstructs the desired sampling distribution. Furthermore, while not required, we found that training the network with a generative adversarial network (GAN) [GPAM*14] rather than the typical losses (e.g., ℓ_2 loss) can improve the quality of the reconstructions in some regions and thereby impacts the quality of the final result.

By posing the problem as a reconstruction (i.e., interpolation) of the incident radiance from existing samples instead of trying to model it directly with a learned function that fits to the specific scene like previous online methods, our distribution-generating function no longer has to be scene-specific. Instead, we can train our system to perform general incident radiance reconstruction *offline* across a wide range of scenes. Once trained, our network can then be applied to any new scene without retraining, so the only expense at render time is the inference time for the network that estimates the distribution function, which takes only 0.6 seconds in total for the entire image. Furthermore, our network can be used by the renderer to guide samples at low sample counts, even as early as after 1 sample per pixel. To validate this, we demonstrate improvements in both overall image convergence as well as with denoised outputs using an off-the-shelf denoiser [BVM*17] relative to current state-of-the-art importance sampling techniques.

However, our approach does have some limitations. Since we store our information in screen-space, it only uses our importance sampling framework in the first bounce of illumination and so it works best for scenes with direct and one-bounce indirect illumination. Furthermore, since we do not explicitly account for the BRDF when importance sampling, it works better for diffuse scenes where the reflected light field is not dominated by the BRDF. Despite these limitations (discussed in Sec. 6), however, our method is able to produce results that are comparable to state-of-the-art even in scenes with multiple bounces or specular/glossy materials.

To summarize, our work makes the following contributions:

- We present the first offline, machine-learning approach for importance sampling in path space. Our system is able to leverage prior knowledge from a wide variety of previous renders in order to reconstruct accurate incident radiance at the first bounce for an arbitrary scene. This allows path guiding with as few as 1 spp initial samples, something not possible with existing approaches.
- To our knowledge, we demonstrate the first practical use of generative adversarial networks (GANs) [GPAM*14] within the MC rendering pipeline. Although GANs have had success on a wide variety of complex tasks, the general, strict requirements placed on the final image tends to deter their use in computer graphics. However, we observed that since the network predicts sampling maps rather than final renders, we could utilize an additional adversarial term in the loss to generate sharper maps that are still constrained to be accurate to the ground truth, but that avoid wasting samples in directions with no radiance contribution.
- We introduce the first large-scale rendering dataset containing high-sample-count reference parameterizations of the incident radiance across a wide variety of scenes and which is suitable for training networks in path space.

2. Background and previous work

As is well known, the process of physically rendering a scene (assuming only surface reflection) can be expressed by the Rendering Equation [Kaj86], written here in hemispherical formulation:

$$\begin{aligned} L_o(x,\omega_o) &= L_e(x,\omega_o) + L_r(x,\omega_o) \\ &= L_e(x,\omega_o) + \int_\Omega L_i(x,\omega)\,\hat{f}(x,\omega,\omega_o)\,d\omega. \end{aligned} \tag{1}$$

Simply, the outgoing radiance L_o from a surface point x in direction ω_o is the sum of the emitted radiance at the point L_e and its reflected radiance L_r, the latter of which is the integral of the product of incident radiance L_i and the cosine-weighted BRDF (written for simplicity as $\hat{f}$) over the entire hemisphere of incident angles ω.

Modern path-tracers compute Eq. 1 with a Monte Carlo (MC) estimator that approximates the reflected radiance L_r at each bounce:

$$\langle L_r(x,\omega_o)\rangle = \frac{L_i(x,\omega)\hat{f}(x,\omega,\omega_o)}{P_\omega(\omega)}, \quad (2)$$

where $P_\omega(\omega)$ is the probability density function (PDF) used to randomly sample the incident-radiance direction, ω. Note, we do not have a summation over a number of samples in this MC estimator since usually only one reflection sample is traced at every bounce, and in this case we are ignoring the fact that path-tracers typically sample direct lighting as well.

The variance of the estimator in Eq. 2, which shows up as noise in the final image, can be greatly reduced by selecting a PDF $P_\omega(\omega)$ that closely matches the integrand in Eq. 1, a process known as *importance sampling* [VG95, Vea97, PJH16]. In the limit, when the distribution is perfectly proportional to the integrand, the variance from even only a single sample is zero, and the estimator produces exact, noise-free results. Hence, there has been considerable interest in the MC rendering community on ways to improve these sampling distributions for path tracing. Given the amount of work on importance sampling techniques (e.g., [LRR04, CJAMJ05, SA07]), we focus on path-guiding methods most related to our own.

2.1. Path guiding

The most recent approaches in path guiding are exploring the use of machine learning to help with importance sampling. The general idea is to train or fit a model from scratch on a per-scene basis to obtain a distribution that is proportional to the integrand from Eq. 1, usually focusing on either sampling the incident radiance or the full product with the BRDF. The differences in these approaches essentially lies in how they propose to generate such a distribution.

For example, Vorba et al. [VKv*14] use continuous Gaussian mixture models (GMM) to represent incident radiance by fitting them to the samples collected from a pre-rendering step using a specialized EM optimization. Additional improvements were achieved by modeling the full product with the BRDF [HEV*16] and by considering Russian roulette and splitting when determining a light path's contribution [VK16]. Furthermore, Simon et al. [SJHD18] also leverage the learning of GMMs to construct a guiding PDF that accurately models the distribution of slow-to-converge regions to better explore hard-to-find paths such as reflected caustics.

Dahm and Keller [DK17] introduced a novel reinforcement learning technique that was effectively coupled with a renderer to guide samples in a reward-driven way. In particular, this reward was directly proportional to the amount of radiance discovered along a specific path so that high radiance directions can be explored faster to improve the convergence rate. The state-of-the-art approach by Müller et al. [MGN17] built upon this with a dynamic data structure that parameterizes the incident radiance using a spatial octree and angular quadtree that are updated or split as additional samples are computed. Recently, Vevoda et al. [VKK18] used Bayesian online regression for direct-illumination sampling and light-picking, but it was not used for indirect bounces.

There are also recent approaches that utilize deep networks for this problem. These methods adapt recent network architectures

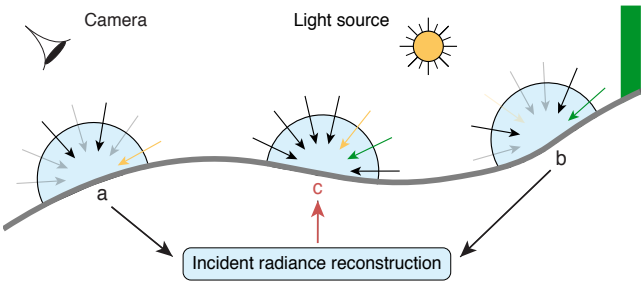

Figure 2: *Overview of our problem formulation. Given sparsely sampled hemispheres of the incident radiance at various first-bounce locations (e.g., points a and b), we must densely reconstruct the hemisphere of incident radiance at a novel location (e.g., point c). Here the solid arrows on the hemisphere represent existing samples, while the transparent arrows represent missing ones. Furthermore, the color of the line indicates the source or contribution of radiance in a certain direction: zero radiance (black), directly from the light source (yellow), or indirectly from the wall (green).*

that utilize transforms based on non-linear independent components estimation (NICE) [MMR*18] or real-valued non-volume preserving transformations (RealNVP) [ZZ18] for the problem of path guiding by leveraging their ability to model accurate densities in high dimensional manifolds. However, these approaches require expensive training in the rendering loop [MMR*18] or have a scene-dependent training phase [ZZ18] that currently take on the order of minutes to hours to train and are scene-specific.

Another learning-based approach that is more related to our own is presented in a thesis by Jiang [Jia18], where a network generates the incident radiance to be used in the final image. On the other hand, our model predicts a radiance map that is used for importance sampling, which avoids artifacts in the final image and when coupled with BRDF sampling is unbiased and will converge. Finally, although it is not used specifically for path guiding, it is worth noting an earlier network-based approach by Ren et al. [RWG*13] which learns to render global illumination effects in real-time by training many multilayer perceptrons (MLP) to model radiance regression functions. This approach requires training MLPs to overfit to each scene and each light source and is computationally expensive to use beyond a small number of point light sources.

We categorize all of the aforementioned methods as "online," since they are essentially "overfitting" on a per-scene basis. In all of these methods, scene-specific data is used to fit a distribution or guide the optimization from scratch. If the scene is changed, these algorithms need to be effectively retrained, so the cost of their training has to be taken into account into the cost of rendering a new scene, just like other scene-dependent costs such as precomputing an acceleration structure. On the other hand, our method is the first data-driven method for importance sampling in an *offline* manner. In other words, we train offline on a variety of scenes to learn how to estimate a good sampling distribution $P_\omega(\omega)$ from an input set of local samples. In this way, our approach is more general and can be used on any new scene without additional training. This not only means faster rendering times, but it can also help guide samples essentially from the start of rendering (e.g., after 1 spp).

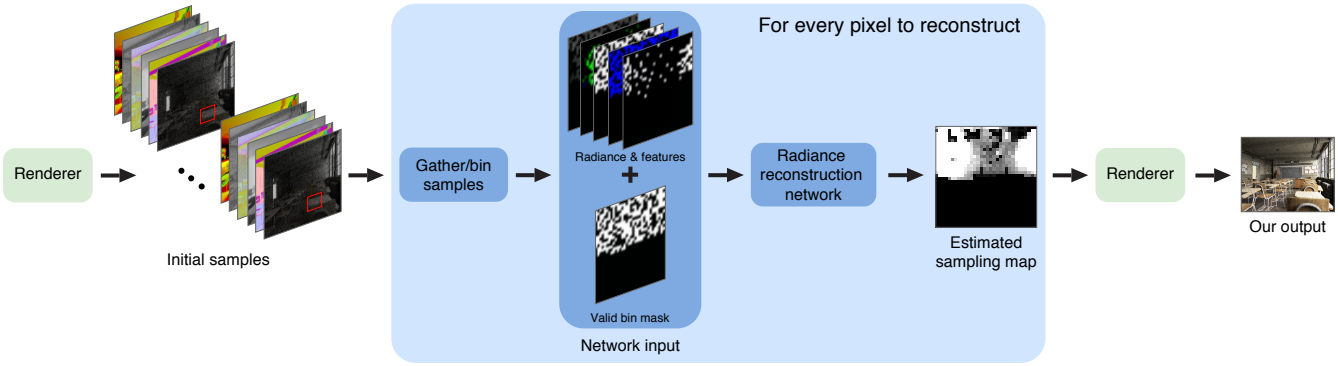

Figure 3: *Overview of our algorithm at run-time. First, we save an initial buffer of a few samples at every pixel from the renderer (e.g, 1 to 8 spp) containing the first bounce incident radiance (from both direct and indirect sources), helpful auxiliary features (e.g., first and second bounce normals and depth), and the incoming direction (as 2-D spherical coordinates). Next, for every pixel we want to reconstruct, we gather neighboring radiance samples (and their features) and bin each sample based on its direction into the corresponding pixel's uniform grid that we chose to parameterize the incident radiance. We store samples for our gather step and then average them (integrate over solid angle) within their respective angular bins. Since we work at low sample counts and with sparse data, we also save out a bit mask representing whether a bin in our grid received any samples that we utilize through masked convolutions in our network. Our CNN then acts on the sparse, uniform grid to produce a dense reconstruction of the incident radiance, which can then be normalized and used as a sampling map to guide the rendering system for the remaining samples and generate the final image. Despite having limited samples and never being trained on this scene, the reconstruction from the network accurately models both the direct and indirect illumination at the first bounce.*

3. Offline deep importance sampling

The goal of our importance sampling approach is to improve the convergence of MC renders by estimating a more accurate PDF $P_\omega(\omega)$ for sampling the outgoing ray directions during path tracing. Since a sizeable contribution to the final image comes from the first bounce and to reduce the memory footprint of our cached radiance data structure, our method focuses on importance sampling this bounce and uses standard multiple importance sampling for later bounces. Furthermore, unlike existing importance sampling methods, our algorithm is specifically designed to work on low-sample-count renderings where current approaches are unable to accurately estimate the sampling distribution and therefore produce extremely noisy results. Although our method still contains noise at these low sampling rates, the level of noise is significantly reduced as compared to other approaches so that it can be better removed by a post-process MC denoiser (see Sec. 5).

Like some previous path-guiding methods [DK17, MGN17, MMR*18], our method estimates the light transport of a new scene by first taking a few initial samples (i.e., tracing paths through the scene). Our approach is based on the observation that the light transport is highly coherent across local scene regions, so although we have only measured a few samples per pixel, we can leverage this coherency in a local neighborhood to improve our estimate of the incident radiance at any point. In other words, we use our noisy estimates of incident radiance across a local region to better estimate (i.e., interpolate) the incident radiance at a particular scene point at the center of the neighborhood (see Fig. 2).

In this way, our approach is similar to algorithms for MC denoising [SD12, RMZ13, KBS15, BVM*17], which also leverage local information (in that case, the colors of neighboring pixels) to improve the reconstruction of the center pixel color value. It is also related to work on light field interpolation and view synthesis [LH96, BBM*01, KWR16], where in our case we are measuring noisy samples of the incident light field sparsely at various scene positions and in a few directions and using that information to "interpolate" the incident radiance at a particular point. Like those methods, we can also leverage additional feature information from the rendering system (e.g., depth and surface normal) to help guide our estimation process to achieve more accurate results.

Inspired by the recent success of machine learning for MC denoising [KBS15, BVM*17, CKS*17, VRM*18], light-field interpolation [KWR16], and for rendering in general [RWG*13, KMM*17, ZWW18], we propose to use a deep network for estimating the sampling PDF from our sparse sets of samples. Deep networks are capable of representing complex, non-linear relationships, and yet perform well with sparse data like we have in our initial samples. The key design questions are then: 1) whether we should train the network in an online fashion to somehow "learn" the incident irradiance field for that scene directly, or rather simply as an interpolator that is trained offline on a set of scenes to learn how to use existing samples to fill in the "holes" and denoise[†] the incident radiance field, 2) what the parameterization of the output PDF from our network should be, and 3) what is the right architecture and training process for the system. We discuss the first two questions in the subsequent sections and the last one in Sec. 4.

3.1. Online vs. offline training

First, we must decide whether we want our network to learn to estimate the PDF in an online fashion as we compute samples, or be trained as a pre-process on other scenes as some kind of incident radiance "interpolator." After some consideration, we found that the

[†] The estimated incident radiance is noisy since we only compute a single path in these directions and have not fully evaluated the rendering equation.

recent approaches of online training for a specific scene [VKv*14, MGN17], especially with deep networks [MMR*18, ZZ18], were not practical. First, these methods require a fairly large number of samples (usually on the order of hundreds of samples per pixel) before they become effective since they train from scratch every time a scene is rendered, and at low sampling rates there simply might not be enough useful information to estimate a good PDF function. Since we wanted our method to work at low sampling rates where MC denoising methods usually fail, this was a significant problem. Furthermore, the network-based online methods require significant scene-dependent training times, which reduces the practical benefits of importance sampling in the first place.

Part of the problem with the online approaches is that they do not leverage any prior information on how nearby samples of incident radiance (even noisy ones) might provide information about the true incident radiance field at a particular point. Rather, these methods are essentially "fitting" the PDF directly to the measured samples, which in the case of a very small number of samples (e.g., 1 spp) is extremely difficult to do accurately without prior knowledge. For this reason, we instead propose to use an *offline*, data-driven approach and train our network on a wide variety of scenes in order to learn how to interpolate neighboring sample information to reconstruct incident radiance and create more accurate PDFs for sampling. This allows us to leverage knowledge from previous renderings and enables us to use our importance sampling earlier in the rendering process. Furthermore, since we perform all of our training offline, there is no expensive per-scene training cost that makes many of the online methods impractical. Rather, the only cost of our method is the feed-forward inference time to evaluate our network and estimate the sampling PDF, which is not significant (around less than a second for the results shown in the paper).

Because we are now optimizing over a large training set, it could be difficult for the system to properly model the entire integrand of Eq. 1 since it includes a BRDF term and thus is dependent on both incident *and* outgoing directions. This would therefore require a lot more training data to ensure we have enough examples. Furthermore, the function to be learned is more complex as the estimated PDF is a 2-D function that is essentially conditioned on both the position x *and* the outgoing direction ω_0 (i.e., $P_\omega(\omega|x,\omega_0)$) in order to account for the dependency of the BRDF on the outgoing direction ω_0. Therefore, to simplify things, we only estimate the incoming radiance which is a 2-D function at each point in the scene and make our PDF proportional to that. Although this assumption essentially ignores the BRDF of the surface when importance sampling, we find that this is sufficient as the incident radiance significantly impacts many scenes, such as those where there are complex light interactions due to occlusions or indirect illumination.

In summary, our network uses the initial sparse, noisy samples of incident radiance to predict a dense probability distribution that would ideally be proportional to the ground-truth incident radiance across the support of the hemisphere. Next, we discuss our parameterizations of the hemisphere to perform this importance sampling.

3.2. Hemisphere parameterization

After deciding to use a pre-trained deep network to estimate our sampling distribution $P_\omega(\omega)$, we now must determine how to represent the sampling distribution over the hemisphere so that we can sample from it. We explored various options, such as using Gaussian mixture models [VKv*14] or spherical harmonics [RH01,SKS02]. However, we found that in order to leverage the power of convolutional neural networks, the distribution of incoming radiance is best parameterized on a uniform 2-D grid of resolution $N \times N$. Each bin in this grid maps to spherical coordinates, θ and ϕ , representing azimuth and polar angles, respectively, about a full sphere parameterized over world space. Note we chose this parameterization rather than one over the local hemisphere since that would have required transformations to align them and leverage the coherence across neighboring samples. Moreover, we perform sampling uniformly along the solid angle rather than in spherical coordinates to avoid biasing samples towards the "up" direction.

3.3. Algorithm overview

To use this with our learning framework at run-time, we first send out a very small initial batch of samples and choose the direction of the subsequent bounce by sampling the hemisphere about the initial intersection point. Note this sample can be acquired by uniformly sampling the full hemisphere or by importance sampling the BRDF. At every pixel, we save each of our samples from our initial budget (which is typically small, e.g., 1 to 8 spp) in memory; samples include the incident radiance from direct light sources and indirect illumination, auxiliary feature information such as first and second bounce depth and normals, and the incoming direction in 2-D spherical coordinates. We gather the samples from a small region around each pixel (more on this later), find the index into the pixel's fixed resolution grid using θ and ϕ, and average the incident radiance and auxiliary features (discarding θ and ϕ) at the corresponding bin. Since we are dealing with a small number of samples, this grid is sparse *and* noisy because of indirect illumination. The sparse grid is then provided as input to the distribution generation network which in turn produces a dense reconstruction of the entire grid at that point. To more efficiently utilize the GPU, we evaluate the network for all the points (i.e., across the entire image) at once after this binning step (see Fig. 3 for an overview).

Since the samples we work with record incoming radiance, the network is essentially reconstructing the incident radiance of the entire hemisphere, which we assume is the distribution we desire to sample from. In order to properly sample from the distribution, we normalize the reconstructed output to generate a valid PDF. Converting this PDF to a CDF allows us to perform standard 2-D importance sampling to choose one of the bins in the grid, and then we can randomly choose a direction within the bin as our sampling direction. This way, as the resolution of the grid goes to infinity, each bin will correspond to only a single direction. Note that we currently treat each bin in the map as a piecewise constant region of the PDF and simply uniformly sample therein. More sophisticated approaches such as interpolating the values across bins (i.e., piecewise linear approximations) are a subject of future work. Finally, we use the predicted PDF to generate samples from our remaining budget and use only these samples to create the final output image. In other words, we use the initial sampling budget only as input for the network reconstruction and discard these samples afterwards.

Overall, there are some practical considerations when utilizing the grid parameterization and sampling strategy discussed above. We found that importance sampling only the first bounce provided

the best balance of improved sampling vs. computation costs as considering the subsequent bounces often resulted in increased computation/storage yet diminishing returns (e.g., from additional samples to process and for resolving occlusions from non-screen-space samples). Moreover, we found that the network does not need to be evaluated at every pixel and instead can use a parameter that skips pixels with a certain stride and interpolates the in-between pixels (e.g., by simply bilinearly interpolating the four nearest radiance reconstructions from our network). Due to the large coherence of the incident radiance in the spatial domain, we found this did not significantly impact quality, but provided substantial gains in speed.

Finally, if we consider only samples at the given pixel for input to our reconstruction network, most bins in the grid will be empty at low sampling rates and therefore the network will not be able to generate anything reliable. Therefore, we again leverage the spatial coherence of scenes and gather all the neighboring samples in a $k \times k$ pixel region around the pixel in question before binning them in the corresponding pixel's grid as is, without warping to account for the change in position. Although this is an approximation of the incident radiance to the given point, it significantly reduces sparsity to ~25% within the valid hemisphere and thereby improves reconstruction quality. Note that as the sampling rate increases, we are able to reduce the size of the window to gather from.

4. Network architecture and training

In this section, we detail components of our learning framework including architecture design, optimization loss, and data acquisition.

4.1. Architectural considerations

Masking invalid regions

The first challenge we faced was sparsity from the low sampling rate, which negatively impacted the optimization and required us to inject knowledge about the sampling; specifically, we mask out invalid regions so they do not contribute to the final output.

Specifically, since we are dealing with sparse data, there are many bins in the input that never received a sample. The radiance value of the bin is zero, yet the network convolutions cannot distinguish between the case that the bin is zero because there really was no contribution or if it is simply zero because it was never sampled in the first place. Furthermore, since we are also considering indirect illumination, there is additional noise coming from the sampling at the second bounce. Having this inconsistency makes it difficult for the network to pick up on patterns. To avoid this issue, we use masked convolutions [LRS*18] in the generator which will perform a convolution but apply a binary mask containing either 0 or 1 in each bin corresponding to whether the bin received any samples. This ensures regions that never received a value have no gradient flow so that the weights are not affected.

Since the grid at every pixel is parameterized in world space and over the entire sphere, half of the grid corresponds to invalid directions (e.g., under the surface) that will always have zero contribution since we are only considering importance sampling of the reflected direction. Therefore, we mask out any contribution from these regions of the grid for every pixel. This includes multiplying both the network input and the network output by a mask that is only on in the valid region of the hemisphere. Note that the current pixel's normals are used to find the valid hemisphere. In the case there are noisy normals at the pixel, we average them first.

Input features

Since our goal is to convert the incident radiance into a PDF, we only need the magnitude of the radiance and so we simply average across the RGB channels to produce a single-channel representation of incident radiance. Each rendered sample also contains the auxiliary features of depth (1 channel) and shading normal (3 channels) for both the first and second intersection points. Thus, our input grid has a total of 9 channels. We omit the sample's 2-D spherical coordinates from our network input, as they are already implicit in the grid parameterization. We also empirically found that they did not improve network performance when included as extra input channels. When reusing samples from adjacent pixels, it could be the case that a sample is binned from intersections on completely different objects that do not share the same incident directions. By including these additional features, the network has more information to use to discriminate against bins. For example, if samples are gathered from different objects, certain features, such as normals can help the network identify differences across bins to weight each one's contribution to the reconstruction appropriately. We experimented with these input features in Sec. 6.

Network design

For our radiance reconstruction network, we use a popular encoder-decoder architecture. The encoder portion will take the sparse, noisy grid of a particular pixel and extract features across multiple scales until it generates a latent feature vector at the lowest scale. The decoder is typically designed as a mirror image of the encoder and will take this latent feature vector and extract features back up all the scales to generate an output with the same spatial resolution as the original, but now contains a dense reconstruction of the incident radiance. This network is shown in the left portion of Fig. 4.

Although not a strict requirement, we also found using a generative adversarial network (GAN) [GPAM*14] framework, in addition to a standard ℓ_1 or ℓ_2 loss, helped produce sharper reconstructions, while still maintaining accuracy relative to the ground truth incident radiance map. The general idea of a GAN is to have two competing networks: a generator and a discriminator. More specifically, the generator produces an output that is sent to a separate discriminator network that determines whether the output is a sample from the true underlying distribution that the generator is trying to model. As training goes on, the generator continues to improve its output to fool the discriminator, while the discriminator also becomes better at discerning examples outside the true distribution. Once training is complete, the discriminator is discarded and only the generator is required for all future inferences.

GANs are a popular choice for many complex applications because they typically can output sharp content, especially compared to standard per-pixel metrics such as MSE, which tend to have blurrier results. Yet, GANs are not used for physically-based rendering because of the strict requirements on being able to produce an artifact-free, final render. However, we avoid such issues in our system by using a GAN for generating the sampling map, rather than the final image directly. Coupling these maps with BRDF sampling, as in previous work, ensures the resulting render will be unbiased.

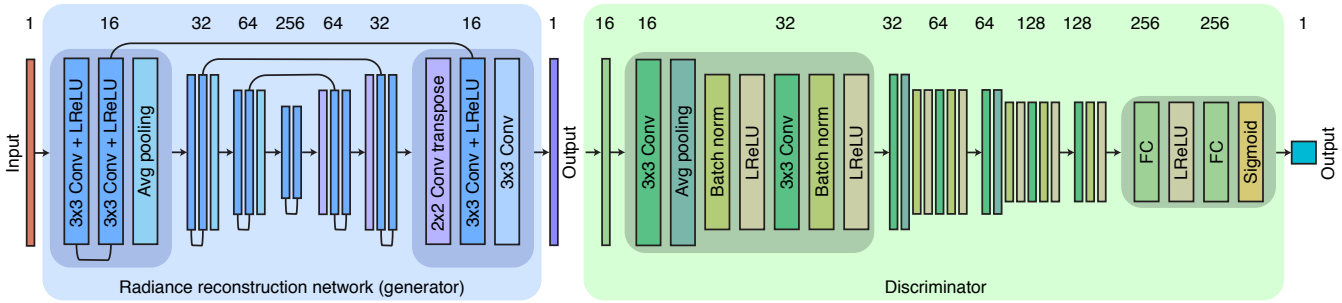

Figure 4: *An overview of the two network components of our full approach. The radiance reconstruction network (blue) generates a dense reconstruction of the incident radiance at the first bounce using only a sparse set of initial samples. During training, the network learns to match a high-sample-count reference of the incident radiance based on our grid parameterization. Furthermore, although not required, a discriminator (green) can be used to improve the quality of the reconstruction and, ultimately, the final rendered output. Note the discriminator is only used during training and then discarded. After the offline training is complete, the network can be used with any new test scene without any additional per-scene, online training. Specifically, the network is used in inference mode to reconstruct the incident radiance for an arbitrary scene, which can then be converted to a sampling map to guide the renderer for its remaining sampling budget.*

Figure 5: *Reference images from our 82 scene Blender training set.*

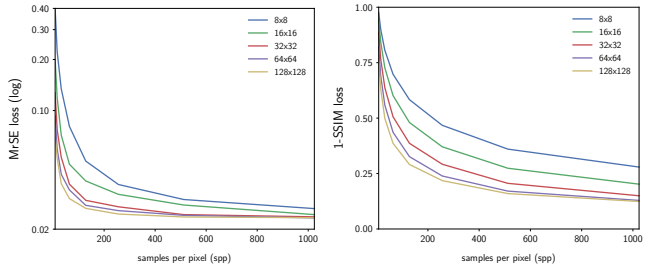

Figure 6: *Convergence plots for our validation set at various angular resolutions. We use* 32×32 *in our pipeline as it performs well without adding significant overhead to overall render times.*

Within the GAN framework, our radiance reconstruction network can be thought of as our generator. Meanwhile, for our discriminator, we use an architecture that is similar to the encoder portion of the generator and has convolutions across multiple scales. However, the coarsest scale has a fully connected layer with a sigmoid activation to output the confidence that the dense incident radiance map that it is given belongs to the true distribution (i.e., the distribution of reference maps across all scenes in the training set).

Both networks are shown in Fig. 4, while implementation details are provided in Sec. 4.4 (see architecture table in the supplemental). Through the rest of this section, we describe our full approach with a GAN, but as shown in Sec. 5, it is not a requirement for our approach to work. Furthermore, in Sec. 6, we provide additional analysis on the improvements stemming from using a GAN-based approach. To train with or without the GAN, we simply add/remove the discriminator and change the loss function, described next.

4.2. Loss

As noted in Sec. 3, the network outputs a dense reconstruction of the incident radiance parameterized as a uniform grid. This is normalized to create a valid sampling distribution to choose the next bounce direction. We chose not to directly optimize for the PDF since it is harder for the network to learn how to map noisy, raw radiances or their corresponding PDFs to a reference distribution.

Therefore, working directly with radiance values means our data has a very large dynamic range that can negatively impact network optimization as well as fail to match dark areas that contribute little to the overall error, yet still have non-zero radiance. Thus, we apply a transformation, similar to that of recent denoising works [BVM*17, VRM*18], to bring the data into a more reasonable range. However, instead of the typical logarithmic transformation, we found better performance when first applying a range compressor, as is done in audio processing and, recently, for learning-based high dynamic range imaging [KR17]. Specifically, we apply

$$T_y = \frac{log(1+\mu y)}{log(1+\mu)}, \tag{3}$$

where μ influences the amount of compression ($\mu = 5000$ in our implementation). Using this allows the dark and bright regions to have similar importance when minimizing the reconstruction loss. During testing, we apply the inverse range compressor to restore the relative intensities of the reconstructed incident radiance.

After accounting for the dynamic range, we utilize the following general loss that has been successfully applied for training generative adversarial networks (GAN) [GPAM*14] in various applications (e.g., super-resolution [LTH*17]),

$$L(T_y, T_{\hat{y}}) = L_{\text{accuracy}}(T_y, T_{\hat{y}}) + \alpha L_{\text{sharpness}}(T_y, T_{\hat{y}}). \tag{4}$$

The first term is a metric that minimizes average per-pixel distance and, in the context of our application, enforces accuracy between incident radiance maps: the range compressed network prediction, T_y, and the reference, $T_{\hat{y}}$. Meanwhile, the second term ensures a sharp output, as blurry maps can result in wasted samples (e.g., across occlusion boundaries) by using one of many potential adversarial losses from other GAN methods (e.g., least-squares [MLX*17], Wasserstein [ACB17, GAA*17]). We found ℓ_1

and the standard cross-entropy losses [GPAM*14] worked best as L_{accuracy} and $L_{\text{sharpness}}$ respectively, with α at 4.0×10^{-3}. Note, if it is desired to use our method without adversarial training (i.e., the reconstruction network only), one would simply set α to zero for training; this setup corresponds to the "Ours (NoGAN)" results in Sec. 5 and Sec. 6. Overall, our full loss with these two terms enables the reconstruction network to produce dense maps that are both accurate *and* sharp relative to the reference.

4.3. Training data

For our training data we collected 82 training scenes for Cycles, Blender's built-in production path tracer [Ble18] (see some sample scenes in Fig. 5). We made them all be diffuse-only to eliminate influence from the BRDF and thereby focus on the dense reconstruction of incident radiance without having to account for material attenuation, which facilitates network training. Furthermore, we render all scenes with direct and one-bounce indirect illumination as we importance sample the first bounce only.

Our ground truth grids were generated with a resolution of 32×32 and with 64K samples, which corresponds to 64 samples per bin. Specifically, at the first bounce we loop through every bin in our grid and send out 64 uniformly-distributed samples to get the average radiance integrated over the bin's solid angle.

To determine the angular resolution to use for representing our sampling distribution, we generated ground truth sampling maps for our validation set in Cycles at 8×8, 16×16, 32×32, 64×64, and 128×128 resolutions. We then used these maps during rendering at all power-of-two sample counts between 8 and 1024 spp and calculated the average MrSE and SSIM of the final rendered image (see Fig. 6). Note the plots were generated by rendering our validation set five times and averaging the error.[‡] Moreover, to account for the variance across scenes, we normalize each error by the error of the 8 spp input image before averaging. As expected, convergence improves with increasing angular resolution. However, there seems to be only marginal improvements beyond 32×32, yet the cost of inferencing increases significantly (i.e., the amount of computation scales linearly with resolution due to the sliding kernels of convolutional networks). Therefore, we chose to use 32×32 in our system as it provided good performance yet inferencing was still fast enough to only add a small overhead to rendering.

Since there is significant correlation across pixels and to save on compute time, we calculate the ground truth for a single pixel in every 8×8 region. Even with striding our scenes are high enough resolution that we still get on average approximately 16K usable patches per scene, giving us over one million total training patches.

Finally, the input data is rendered with 8 samples per pixel using uniform sampling over the valid hemisphere. Next, the input to the network is obtained by binning all the samples in an 11×11 window (or a 21×21 window for extremely low sample counts, i.e., below 8 spp for the initial sampling rate) around the current pixel.

We plan to release our large dataset, including both the scenes and training data, to inspire future research.

[‡] We use a trimmed MSE where the top 0.01 percent of pixels are removed in order to avoid skewing statistics from outliers/fireflies.

4.4. Implementation details

We implemented our method in TensorFlow [AAB*15] and used NVIDIA M6000 GPUs for training/testing. We found pre-training as in some GAN approaches [LTH*17] was not necessary and we trained L_{accuracy} and $L_{\text{sharpness}}$ from scratch simultaneously until we found a good optimum after $820K$ iterations. We used mini-batches of size 16 and both the generator and discriminator networks were optimized using ADAM [KB14] and a learning rate of 1.0×10^{-4}.

For our generator, we use the popular encoder-decoder architecture with three scales. The encoder portion starts with 16 feature maps and increases by a factor of two to reach 256 in the coarsest scale using average pooling as the downsampling scheme. Meanwhile, the decoder portion is the opposite and goes from 256 to 16 feature maps using transposed convolutions of stride 2 for the upsampling. The final output has only a single channel and is the same resolution as the input. At every scale of both the encoder and decoder we have two convolutions (using Xavier initialization [GB10]) with residual connections across them and Leaky ReLU (LReLU) ($\alpha = 0.2$) activation functions [XWCL15] in between. In addition to residual connections, we have skip connections from the encoder to the decoder, as we found both of these strategies helped promote gradient flow throughout the network.

Meanwhile, our discriminator is mostly made up of repeated convolution blocks at each scale, each of which contains a convolution (using truncated $\mathcal{N}(0, 0.2)$ initialization), batch normalization, and a LReLU ($\alpha = 0.2$) activation function. The number of the feature maps starts with 16 in the finest scale, increases by a factor of 2 in each scale, and ends at 128 in the coarsest scale before a dense layer of size 256 that outputs a single final value corresponding the confidence that the input was a real sampling map. We use a sigmoid activation function in the output layer and use average pooling when downsampling the spatial resolution.

5. Results

5.1. Setup

To evaluate the performance of our offline deep importance sampler (ODIS) and to illustrate its ease of use, we integrated our network within two rendering systems: Blender/Cycles [Ble18] and Mitsuba [Jak10]. In general, for a given scene, we render a fixed number of initial samples (e.g., typically 1 to 8 samples per pixel) that can be either uniformly or BRDF sampled about the hemisphere of the first intersection point at each pixel. As discussed in Sec. 3, we save out direct *and* one-bounce indirect illumination for each of these samples. Then, after binning these samples into our sparse, grid representation, we do a forward pass of network inference, without any online training, to reconstruct the full dense hemisphere. This output map is then converted to a PDF/CDF for importance sampling the *remaining* sample budget of the render. The network is fixed during rendering and no scene-specific training occurs. Furthermore, the initial samples are used only as the input for the network to reconstruct the sampling map, and their values are never used directly in the final output.

Our approach is orthogonal to light picking/next-event estimation (NEE), so for all methods and all comparisons shown we use full multiple importance sampling (MIS) to show the overall con-

Scene	Path			OLPM			PPG			Ours (NoGAN)			Ours (ODIS)		
	Trim	Non-trim	Timing	Trim	Non-trim	Timing	Trim	Non-trim	Timing	Trim	Non-trim	Timing	Trim	Non-trim	Timing
Veach Ajar	18.89	89.46	2.9 s	**1.67**	46.44	26.7 s	16.14	64.93	3.2 s	3.07	39.38	3.5 s	2.54	**9.73**	3.5 s
Classroom	6.52	**8.01**	1.9 s	6.01	39.68	26.6 s	9.89	14.32	2.1 s	4.04	45.03	2.5 s	**3.76**	13.50	2.5 s
Dining Room	12.60	**22.17**	2.6 s	16.95	119.18	23.8 s	18.44	49.84	2.8 s	5.72	537.96	3.2 s	**4.12**	31.54	3.2 s
Living Room	9.15	**43.83**	2.0 s	9.47	73.17	32.6 s	18.17	81.11	2.4 s	5.76	212.25	2.6 s	**5.09**	76.53	2.6 s
Kitchen	25.11	45.04	9.6 s	16.17	95.58	34.0 s	30.15	97.58	9.9 s	9.54	426.42	10.2 s	**8.79**	**27.32**	10.2 s

Table 1: *Error and timing comparisons across methods for the Mitsuba test scenes from Fig. 8. For each method, the left and middle columns are trimmed and non-trimmed versions of MrSE, respectively, and the right column is the total timings (including scene-specific, pre-processing steps) in seconds. High radiance samples (spikes) largely impact metrics such as MrSE, so we report trimmed errors throughout the paper. Moreover, our network inference adds only a small overhead to the render cost. See full reports with trimmed/non-trimmed errors.*

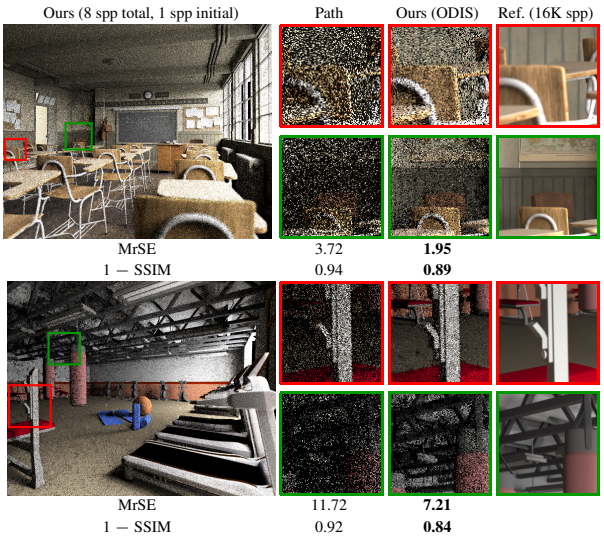

Figure 7: *Results compared to standard path tracing for two test scenes in Blender/Cycles that were not included in our offline training. Both methods use light-picking, while our approach uses our network to select the next bounce direction instead of BRDF sampling. Even with only an initial 1 spp input buffer, our network is able to reconstruct a dense sampling map to guide rendering for the remaining 7 spp and produce a significantly more converged result.*

vergence improvements in the typical use case.[§] Specifically, a single sample consists of direct light sampling *and* either the method's proposed sampling strategy or the standard sampling of the bidirectional reflectance distribution function (BRDF) as determined by a parameter. For our approach, we used no BRDF sampling (i.e., light-picking and our network-based sampling only) in our comparisons, unless otherwise stated (i.e., Sec. 5.3.1 and Fig. 13), in order to demonstrate the benefit of our learned approach. Meanwhile, for all other methods, we show their best parameter sets in the paper, based on the lowest average error across our test scenes, and provide the results of the full parameter sweep in the supplemental.[¶] It is worth noting that in practice all of the approaches compared here, including ours, would need to utilize some BRDF sampling to guarantee convergence at high sample counts.

Our method was designed to importance sample the bounce direction of primary rays, so we revert back to standard MIS for the second bounce. The scenes in this section are diffuse and are rendered with direct and one-bounce indirect illumination. Despite not having interactions with glossy materials, the evaluation scenes incorporate challenging cases with multiple light sources of different strengths and varying degrees of visibility resulting from complex scene occlusion. Moreover, we target single-digit sample-counts with extremely sparse information and severe noise, which make the task of predicting next-bounce directions even more difficult.

For quantitative evaluations, we report in the paper the commonly-used mean relative squared error (MrSE) and the popular, perceptually-based Structural Similarity Index (SSIM)[††], and in addition include mean square error (MSE), mean absolute percentage error (MAPE), and ℓ_1 in the supplemental. Note, all methods are affected by fireflies, so in order to draw meaningful observations about convergence and performance, we use a *trimmed* metric where we discard the highest 0.01% of pixel errors when calculating the average for all methods when reporting metrics influenced by outliers such as MrSE [BVM*17, VRM*18]. However, in Table 1, we report both trimmed and non-trimmed versions of MrSE for reference. Please see the supplemental reports to view these results and more in full resolution using our interactive image viewer including both trimmed and non-trimmed results.

5.2. Blender/Cycles

We first apply our method in Blender's production-quality path tracer, Cycles [Ble18], on test scenes that were *not* part of the training set. Fig. 7 shows comparisons between our approach and standard, unidirectional path tracing with MIS at a total budget of 8 spp. For these scenes, we render an initial 1 spp buffer that our network uses to generate a sampling map to guide the renderer for the remaining 7 spp. Note these 7 spp are used to generate the final image

[§] The error plots from Fig. 6 are the only results without MIS, since the experiment was to compare the performance of various angular resolutions, independently of light-picking.

[¶] We choose between three options: no BRDF sampling, fifty-fifty split between the method and BRDF sampling (the default parameter of both Vorba et al. [VKv*14] and Müller et al. [MGN17]), or full BRDF sampling for the second bounce. Note, standard path tracing always uses BRDF sampling.

[††] Since higher SSIM scores correspond to closer images, we report 1-SSIM for consistency with our other metrics.

Figure 8: *Equal-sample comparisons with state-of-the-art importance sampling techniques in Mitsuba including OLPM [VKv*14] and PPG [MGN17]. The top row shows the rendered output using the sampling technique, while the bottom row demonstrates the results of also applying KPCN [BVM*17], an off-the-shelf MC denoiser. Both MrSE and 1-SSIM are reported below for the rendered output and the denoised result (in parentheses). Ours and Ours (NoGAN) are relatively more converged, which significantly improves the final denoised output. Although the NoGAN version outperforms previous methods, there are still regions where it underperforms relative to our full algorithm. Our networks were trained with Blender/Cycles scenes and were not retrained for Mitsuba. Full images are in the supplemental.*

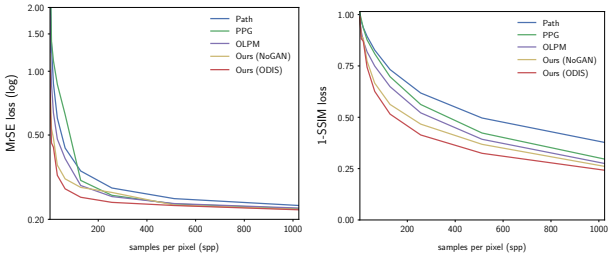

Figure 9: *Convergence plots among the different importance sampling techniques averaged across all of our Mitsuba test scenes for MrSE and 1-SSIM. In particular, both Ours and Ours (NoGAN) perform favorably relative to previous approaches, especially at low sample counts. However, the NoGAN version of our method still slightly underperforms relative to our full approach.*

and the initial 1 spp used by the network is discarded, while path tracing uses the full 8 spp to generate its result. The images have a resolution of 1280×1024 and the network is inferenced at a stride of 8, using bilinear interpolation for the in-between pixels.

The **Classroom** scene is a difficult case to render with complex direct and indirect illumination since there are no emitters on in the room itself. Instead, there are four large, bright area light sources that are all at least partially occluded. Essentially, each primary ray has to decide which direction to leave the room (e.g., the door on the left or the window on the right). The path tracing result is extremely noisy as it has to both pick one of the emitters as well as the position on the light, both of which have a high likelihood of being blocked. Meanwhile, our network's dense prediction of the incident radiance provides a more reliable way of finding which are the best paths out of the room and it even enables recovery of part of the map texture from the nook, that is otherwise lost with path tracing. Note, the inferred sampling map for this scene, shown in Fig. 3, demonstrates the network's robustness to sparse samples.

Similarly, the **Gym** scene also has all of its six light sources off-camera and outside of the large building. It is challenging to find which light source to sample from and which path out the windows has the most contribution. Furthermore, various structures, such as the beams in the rafters on the ceiling, provide additional occlusion that the path tracer cannot explicitly avoid through simple light and BRDF sampling. Meanwhile, our approach can model the scene's visibility more accurately to avoid such structures when selecting a path and gives a relatively more converged result (see Fig. 7).

5.3. Extrapolation to Mitsuba

Next, we demonstrate the robustness and ease-of-integration of our approach by taking the network trained on Blender data and plugging it directly into the Mitsuba rendering system [Jak10]. The only instrumentation that is required is slightly modifying the standard sampler to importance sample from our network generated map or the BRDF during MIS based on the aforementioned user-set parameter. More importantly, we *do not retrain* the network and use the same weights as before with Blender/Cycles to generate all results.

We compare against standard, unidirectional path tracing as well as state-of-the-art online sampling techniques: online learning of parametric mixture models (OLPM) [VKv*14] and practical path guiding (PPG) [MGN17]. The OLPM method fits samples to Gaussian mixture models as a way to parameterize the incident radiance for importance sampling. Meanwhile, PPG instead uses a dynamic spatial octree and angular quadtree as its representation, which are continually refined with additional samples during rendering.

For comparisons, we used the author-released implementations and their default values for most parameters. However, since we primarily target sample-counts that are relatively lower than what these methods were optimized for, we performed parameter sweeps to find the best settings for each method at all sampling rates. The first of which is the optimal BRDF sampling fraction mentioned earlier. We repeat this parameter sweep for both bounces since, for example, it might be more beneficial to exclusively use a method's technique at the first bounce and then switch to BRDF sampling at the next bounce, as we do in our approach.

Furthermore, for Müller et al. [MGN17], we also explore one of their parameters that controls the number of samples used during their training passes, which can be a sensitive parameter when dealing with only a few spp for the *total* budget. Finally, for Vorba et al. [VKv*14], we used 30 pre-training passes and 300,000 for the number of importons and photons, the default parameters found in their released scenes. To their advantage, we do not count these towards the sample budget even though they are equivalent to tracing paths through the the scene. For simplicity, we include only the best results from the competing approaches in the paper and provide the results from all the parameters in our sweep in the supplemental as well as the selected best parameters from each sampling rate.

Finally, to demonstrate the benefits of using a GAN in our full approach, we provide both quantitative and qualitative results with "Ours (NoGAN)." This network has the same architecture and training setup as our full proposed approach except it is trained using only an ℓ_1 loss (i.e., the α parameter from Eq. 4 is set to zero) and it is trained without the discriminator network from Fig. 4.

One of the main benefits of our approach is significantly improved convergence at low sample counts which can be utilized by other applications further downstream in the rendering pipeline. To demonstrate this, we took KPCN [BVM*17] a recent, off-the-shelf Monte Carlo (MC) denoiser and applied it to the rendered result of each method to show that improved convergence corresponds to improved denoising performance. Note that KPCN is not retrained and is applied with fixed weights, but we verified that it gave comparable results in Mitsuba for the same scenes and sample counts shown in the original paper with the Tungsten [Bit16] renderer.

In Fig. 8, we show equal-sample comparisons on five test scenes from Mitsuba at resolution 700×400. The raw output from the renderer is shown in the top row, while the denoised output from KPCN is in the bottom row. All methods are given a total budget of 4 spp to use as desired. For our approach, we discard the initial 1 spp buffer after our network inference, as in Fig. 7, but this time we only render three additional samples. These results along with additional results using a higher 8 spp budget are all contained in the supplemental materials. Note that since our method operates in pixel-space, we report traditional samples per pixel, but some methods (e.g., Müller et al. [MGN17]) are parameterized in world space and are resolution-independent. Recent work [MMR*18] report sample count as mega samples (MS) instead to reflect this, but

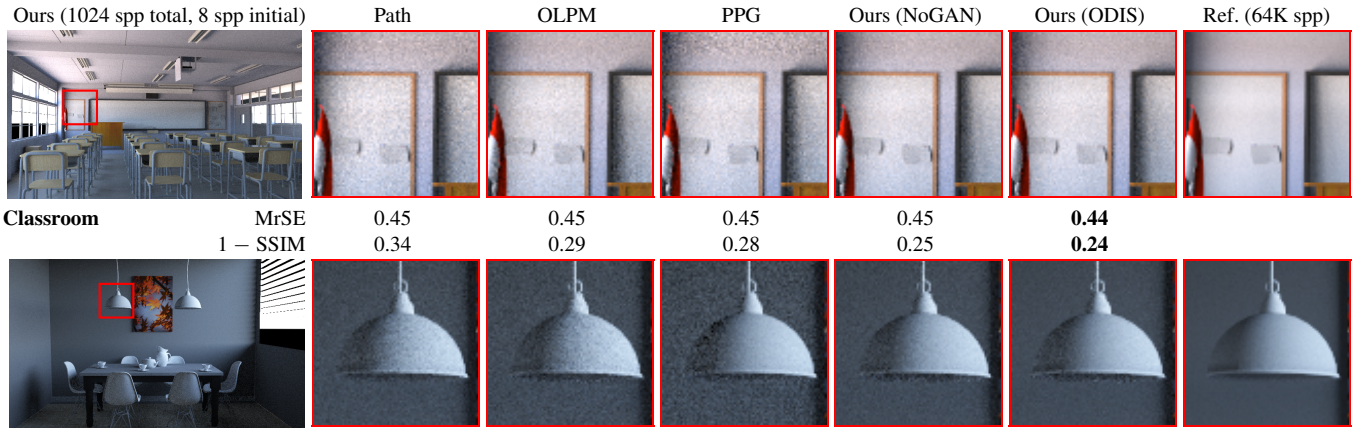

Figure 10: *We compare favorably against state-of-the-art approaches designed for high sample count renders (e.g., 1,024 spp shown here). Our results use only 8 initial samples to reconstruct the map and guide the remaining samples. The network is applied only once and is not trained on these scenes. Best to zoom into the electronic version to see the differences. Full results can be found in the supplemental.*

they still tend to focus on significantly higher sample counts than shown here (e.g., ~100 MS vs ~1 MS for the results in Fig. 8).

In the first row, we show results on the **Veach Ajar** scene, a notoriously challenging scene in terms of convergence since almost the entire scene is lit by indirect illumination flowing through a cracked door. Since almost no light can be obtained at the first bounce and most of the contribution comes from the second bounce, it is important that the sampling strategy guide rays to have their next bounce close to the door. BRDF and light sampling are not very effective in this scenario so both PPG and path tracing are slow to converge. On the other hand, Ours, Ours (NoGAN), and OLPM perform perform reasonably in terms of the denoising results, especially when considering the input samples.

However, although OLPM has regions including the wall that converge well, there are other regions with artifacts such as the large dark holes by the table where the method cannot match paths between the camera and the light during the method's bidirectional training stage and which are too severe to reconstruct with the denoiser used. To be sure this was correct, we verified that the OLPM implementation converged to the correct result given long enough training times (see additional details in the supplemental).

It is also worth noting our improvements on the difficult **Kitchen** scene. The limited illumination is coming from an environment map outside the room. Most of the light enters the room through the small window, so direct lighting samples will be occluded and BRDF samples will typically fail to exit the room resulting in extremely noisy results from path tracing. Furthermore, at such a low sample count Müller et al. [MGN17] still has not refined the quadtree/octree enough to see a benefit. Sampling from their coarse data structure is counterproductive and reduces the quality in certain regions relative to standard MIS with BRDF sampling. Meanwhile, Vorba et al. [VKv*14] also runs into problems since the Gaussian mixture models (GMMs) that were fit in their prerendering training stage are not reliable and would need significantly more samples to discover additional useful light paths and generate more accurate sampling densities.

Overall, both Ours and Ours (NoGAN) perform favorably relative to previous approaches both before and after denoising and we encourage readers to view the supplemental reports to better compare results. There is notable improvement when using our full approach relative to the NoGAN version both perceptually and numerically. In particular, the insets for **Dining Room** and **Veach Ajar** in Fig. 8 highlight regions where the rendered output is considerably less converged and which leave residual black artifacts even after denoising. We provide additional discussion about our performance relative to that without a GAN in the next section.

For completeness, we also report trimmed and non-trimmed versions of MrSE across 5 runs with different random seeds for all methods in Table 1. Note we took the median of the non-trimmed errors as we saw all approaches were largely impacted (i.e., due to spikes from rare samples of high radiance). It is interesting also that the NoGAN version tends to have more fireflies than our full approach as shown by the large values for the non-trimmed errors. The NoGAN version produces overblurred sampling maps that will sample relatively more frequently along occlusion boundaries and will more often get extremely high-radiance, low-probability samples, which will largely impact the error.

It is worth noting that in addition to the benefit of not having to retrain for a new rendering system, we also still do not require any online training, which can be difficult to inject into established production pipelines, and require only a single forward pass of the network. The network is also robust and extrapolates well and we were able to generate most of our results (unless otherwise noted) using only an initial 1 spp buffer despite the network only being trained on reconstructions from 8 spp buffers (i.e., the network extrapolates to lower sampling rates than what it was trained on).

5.3.1. Convergence

We also compare the convergence characteristics of our approach to those of other path-guiding methods as well as our method without a GAN. Fig. 9 shows both the average MrSE and SSIM errors across our 5 Mitsuba scenes using the rendered outputs of each

method at power-of-2 sampling rates from 2 to 1024 spp. We account for different scene variances by dividing each method's error by the error of the 2 spp path traced result for each scene before averaging. Our methods used 1 spp initial buffers to generate results at a total of 2, 4, and 8 spp, and 8 spp initial buffers when the sampling budget was 16 spp or beyond. All of our results are generated without any online training or additional inferencing (beyond the initial one). Note, our 1024 spp results are done with fifty-fifty BRDF sampling, as the other approaches, to ensure convergence.

Both Ours and Ours (NoGAN) converge significantly faster than the other approaches at the lower sampling rates, yet continue to perform competitively at higher ones, especially in terms of SSIM. Moreover, we again still see the NoGAN version of our approach underperforming compared to our full approach due to its relatively less accurate maps which slightly decreases its convergence rate. Note for these plots we used the best parameter setting for each method at each sampling rate since previous methods' default settings are not optimized for low sample counts. Please see the supplemental for additional convergence results with both the best settings and the default parameters.

Moreover, Fig. 10 demonstrates that our approach is also practical for high-sample-count renders, in addition to denoising. Even at 1024 spp, our rendered result has visibly less noise, despite only using 8 initial spp to guide the remaining budget (1016 spp), without any retraining or additional inferencing. Note, the images are best viewed at full resolution in the supplemental.

5.3.2. Complexity and timings

Since the reconstruction network operates in screen space, our method's cost scales linearly with the number of image pixels and is independent of scene complexity. As mentioned in Sec. 3.3, we can perform the incident radiance reconstruction at a subset of pixels to reduce the total runtime drastically without compromising significantly on quality. For example, in our evaluation, we found using an 8×8 uniform stride allowed us to perform the inference on the Mitsuba scenes shown here in only 0.6 seconds in total for all pixels, a negligible addition to the overall render time (see Table 1 for timing comparisons on an eight core machine), all while still maintaining quality. In general, this stride can be decided by the user based on the application (e.g., relatively faster rendering pipelines might use even larger strides at the cost of accuracy).

6. Discussion, limitations, and future work

Ideally, we would learn to generate samples that would minimize a pixel-wise ℓ_1 or ℓ_2 loss between the *final* rendered output and the high-sample-count reference image. Unfortunately, this requires the renderer to be inside the loop of training and would require the ability to differentiate through the renderer to optimize the network. Differentiable renderers have recently been receiving more attention [LADL18], but they are an active area of research and still not easily incorporated into such deep learning frameworks.

Instead, we optimize the incident radiance sampling maps directly. We initially attempted to optimize them using an ℓ_1 or ℓ_2 loss with highly-sampled, reference incident radiance maps, which is akin to the NoGAN version we compare against in Sec. 5. Since

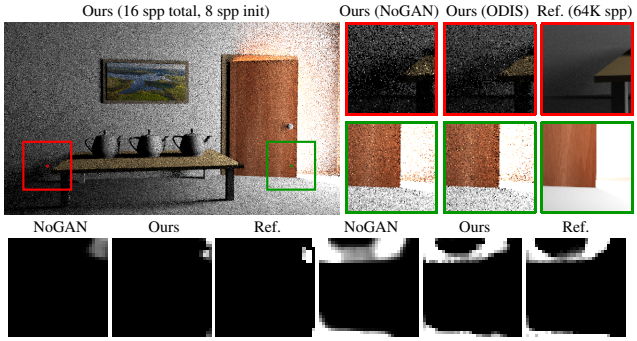

Figure 11: *The red inset on the left shows a slow-to-converge region of the NoGAN approach on the **Veach Ajar** scene at 16 spp. The corresponding radiance reconstructions are shown on the bottom left. In this region, the slight overblurring negatively impacts convergence. On the other hand, the green inset and radiance maps on the lower right show a region with similar quality in the final output from the two networks. Since there are many valid directions to sample from, overblurring will have a smaller influence. Meanwhile, our full approach with a GAN is able to generate accurate and sharp sampling maps in both regions for improved quality and convergence. Note, both reconstruction maps are from a strided, non-interpolated pixel at the center of each inset shown with a dot.*

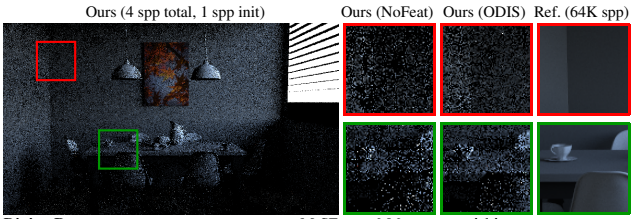

Figure 12: *Comparison to our network trained without auxiliary features (i.e., using sparse incident radiance as the only input). Without features (i.e., "NoFeat"), the network produces a less converged result at a given sampling rate (e.g., 4 spp) relative to our full approach. Features related to depth and normals are helpful in deciding a bin's reliability, thereby guiding the network to have a more accurate reconstruction and, ultimately, better sampling.*

our input data is extremely sparse (just a few sampled directions in each hemisphere), the network can have difficulties properly filling in the large missing regions/holes and outputs blurry maps instead. These maps waste samples on areas with no light contribution (e.g., occluded regions) and ignore directions with significant incoming radiance, thereby significantly slowing convergence. Instead, we chose to use an ℓ_1 + GAN loss to have both sharp *and* accurate sampling maps, a strategy that has achieved state-of-the-art results in applications such as super-resolution [LTH*17].

For example, in the red inset of Fig. 11, we highlight a region of the **Veach Ajar** scene rendered with an 8 spp initial budget and a 16 spp total budget where Ours (NoGAN) is clearly less converged than our full approach with a GAN ("Ours"). In the bottom left, we show the radiance maps at one of the less converged pixels from the NoGAN image and compare it to Ours and the high-sample-count reference. In this scene, light enters only through the cracked door

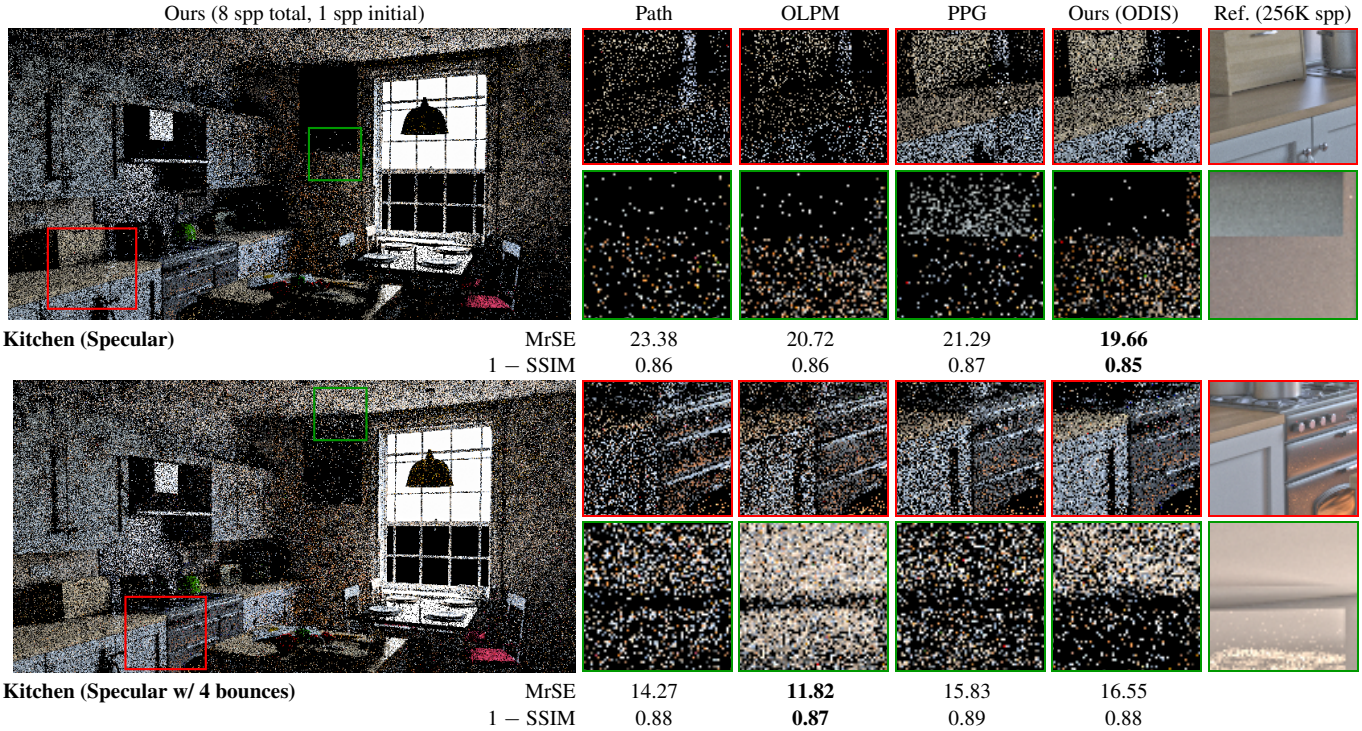

Figure 13: *Examples of limitations of our method. The top row shows an example of the original Kitchen scene from Müller et al. [MGN17] that contains specular/glossy materials. Since our method only importance samples the incident radiance and not the full product with the BRDF, it could waste samples in directions that have no final radiance contribution. Meanwhile, the bottom row has the same scene but it includes additional indirect bounces (i.e., 4 total bounces) that have significant radiance contributions at higher bounces (e.g., the mirror in the green inset). Since our approach only importance samples our network generated maps at the first bounce and uses standard MIS for the subsequent bounces, it will have difficulty discovering directions where only a small solid angle has contribution and also cannot be utilized to choose better sampling directions at later bounces.*

on the right and this pixel receives most of its radiance contribution through a specific direction from another bounce towards the door.

Our radiance map is sharper and closer to the reference compared to the NoGAN version, which is blurry and thereby more frequently assigns nonzero probability for directions with no contribution. For low-sample-count renders, it is important to avoid wasting samples in these directions as it will noticeably impact convergence. On the other hand, there are regions such as the green inset on the right where many directions have non-zero contribution and a slight overblurring will have less impact. Thus, the final output quality of the two networks is comparable in these regions.

Ultimately, our full network with a GAN is more robust as it can handle both cases and produces maps that are both sharp and accurate. This translated to consistently better performance in all of our evaluations relative to the NoGAN version of our network.

In Fig. 12, we show the importance of our auxiliary features (e.g., first and second bounce normals and depth) in the quality of the results. The "NoFeatures" comparison uses only sparse incident radiance to predict the dense reconstruction without the guidance of the features present in our full approach, and, as a result, produces less accurate sampling maps and worse convergence both numerically and perceptually. In the red inset, we see slower convergence along the wall where, for example, second bounce depth could be helpful in determining which bins are being occluded by an object and which ones are seeing the light source. Meanwhile, the shading normals of the chairs and cups (green inset) can be leveraged by the network to determine radiance coming from different objects. We found the auxiliary features were useful across all the scenes we tested on, but it would be interesting to further explore additional features that can be leveraged for even better results.

Our approach has some limitations, however, which are subjects of future work. For example, in Fig. 13, we show how our algorithm performs on the Kitchen scene from Müller et al. [MGN17], which is a challenging case for our approach due to the specular/glossy materials as well as the significant radiance contributions coming from higher order bounces such as the highlight in the mirror shown in the green inset. For all of these comparisons, except path tracing, we randomly choose with equal probability (i.e., fifty-fifty) between BRDF sampling or sampling using the method.

The top row shows comparisons of the scene with specular interactions but still using two bounces (i.e., direct and one-bounce indirect). The red inset shows a countertop with some specular highlights where our approach and PPG are relatively more converged than OLPM and standard path tracing because of the counter's diffuse lobe. On the other hand, for the green inset, we show how the different methods handle the boundary between a mirror and a diffuse wall. Both OLPM and our approach are more converged on

the wall, while PPG is able to discover more radiance contributions along paths from the mirror. Due to the mirror's perfectly specular BRDF, most incident directions will have no contribution even if they are otherwise visible. Since the BRDF is not accounted for in our reconstructed sampling maps, we will waste samples on these zero-radiance paths and have a relatively less-converged result.

In order for our system to handle these scenes properly, we need a way to account for the BRDFs at each surface. We believe that learning to directly reconstruct the full product of the incident radiance with the BRDF from sparse samples is a complex relationship that could be difficult to properly capture with our network. However, one alternative to better handle BRDFs would be to precompute the BRDF with the same grid parameterization as our incident radiance. During runtime, this data could be queried to evaluate the product and importance sample the result instead of just the incident radiance. This has the extra overhead that the BRDF would have to be precomputed as a 4-D table (accounting for incoming and outgoing directions), but this would be done only once per BRDF (not per-scene) and would not significantly impact runtimes.

In the bottom row, we show the scene with specular materials, but having 4 total bounces. The red inset shows that we continue to converge relatively well along the diffuse cabinets, but we miss more of the orange highlights compared to other approaches. In particular, OLPM is able to collect the most orange highlights along the window in the lower right corner of the inset. Meanwhile, the green inset shows OLPM continues to perform particularly well along the boundary between the wall and the mirror and already has the correct brightness with only a few samples. Furthermore, PPG is also more converged in the specular mirror region when compared to our approach, while ours does better on the wall. Both OLPM and PPG can utilize radiance contributions that occur many bounces away. However, our approach works in screen space and uses standard MIS beyond the second bounce, so it is unable to utilize the network's reconstructed maps at later bounces to improve convergence. Since the previous approaches work in world space instead, they are able to importance sample from their fitted model (OLPM) or data structure (PPG) at every bounce. We had initially explored reconstruction in path space, but found that 4-D and 5-D convolutions were slow, difficult to train, and require more study.

Another limitation is that we use fixed grid resolutions and if the resolution is too coarse then our method could fail to discover and properly sample small light features (e.g., caustics) and could also have reduced performance with tight, specular BRDFs that only allow light through a small solid angle. Finally, our code is not optimized and it might be possible to get inference times that are faster than 0.6 seconds and to additionally explore inferencing at multiple stages of rendering to continually update our sampling map, thereby further improving convergence at high sample counts.

7. Conclusion

In this paper, we presented an algorithm for importance sampling the first-bounce direction during path tracing by estimating the incident radiance at a point on the scene using neighboring sample information. As with related work on MC denoising, we do this by leveraging deep learning networks that can learn the complex interpolation function that maps the neighborhood of samples to an accurate estimate of the incident radiance. Unlike existing path-guiding algorithms which require an expensive online training step during rendering, this formulation allows our method to be trained entirely offline on a set of training scenes, making the algorithm much faster at runtime and leverages this multi-scene model to produce good results even at extremely low sampling rates. Our method is straightforward to integrate into existing rendering pipelines, as we demonstrate by incorporating it into both the Mitsuba and Blender/Cycles renderers, and we plan to release our code, trained weights, and the training set upon publication.

8. Acknowledgments

We gratefully thank Chris Hellmuth for helping with the production of this paper including generating the denoising results and supplemental reports. We thank the following artists for the scenes used in the comparisons: Benedikt Bitterli (Veach Ajar), Wig42 (Living Room/Dining Room), NovaAshbell (Classroom), Jay-Artist (Kitchen), Christophe Seux (Classroom in Cycles), and muhtesemozcinar (Gym). Additional license information and artist acknowledgments are in the release materials. We thank Pixar for their gift and support. This work was partially funded by National Science Foundation grants #IIS-1321168 and #IIS-1619376.

References

[AAB*15] ABADI M., AGARWAL A., BARHAM P., ET AL.: TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. URL: http://tensorflow.org/. 8

[ACB17] ARJOVSKY M., CHINTALA S., BOTTOU L.: Wasserstein GAN. *arXiv preprint arXiv:1701.07875* (2017). 7

[BBM*01] BUEHLER C., BOSSE M., MCMILLAN L., GORTLER S., COHEN M.: Unstructured lumigraph rendering. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 425–432. 4

[Bit16] BITTERLI B.: Rendering resources, 2016. https://benedikt-bitterli.me/resources/. 11

[Ble18] BLENDER ONLINE COMMUNITY: *Blender - a 3D modelling and rendering package*. Blender Foundation, Blender Institute, Amsterdam, 2018. URL: http://www.blender.org. 8, 9

[BVM*17] BAKO S., VOGELS T., MCWILLIAMS B., MEYER M., NOVÁK J., HARVILL A., SEN P., DEROSE T., ROUSSELLE F.: Kernel-predicting convolutional networks for denoising monte carlo renderings. *ACM Transactions on Graphics (TOG) 36*, 4 (July 2017). 1, 2, 4, 7, 9, 10, 11

[CJAMJ05] CLARBERG P., JAROSZ W., AKENINE-MÖLLER T., JENSEN H. W.: Wavelet importance sampling: Efficiently evaluating products of complex functions. *ACM Transactions on Graphics (TOG) 24*, 3 (July 2005), 1166–1175. 2, 3

[CKS*17] CHAITANYA C. R. A., KAPLANYAN A., SCHIED C., SALVI M., LEFOHN A., NOWROUZEZAHRAI D., AILA T.: Interactive reconstruction of noisy monte carlo image sequences using a recurrent autoencoder. *ACM Transactions on Graphics (TOG)* (July 2017). 2, 4

[DK17] DAHM K., KELLER A.: Learning light transport the reinforced way. *arXiv preprint arXiv:1701.07403* (2017). 2, 3, 4

[GAA*17] GULRAJANI I., AHMED F., ARJOVSKY M., DUMOULIN V., COURVILLE A. C.: Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems* (2017), pp. 5769–5779. 7

[GB10] GLOROT X., BENGIO Y.: Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics* (2010), pp. 249–256. 8

[GPAM*14] GOODFELLOW I., POUGET-ABADIE J., MIRZA M., XU B., WARDE-FARLEY D., OZAIR S., COURVILLE A., BENGIO Y.: Generative adversarial nets. In *Advances in Neural Information Processing Systems* (2014), pp. 2672–2680. 2, 6, 7, 8

[HEV*16] HERHOLZ S., ELEK O., VORBA J., LENSCH H., KŘIVÁNEK J.: Product importance sampling for light transport path guiding. In *Computer Graphics Forum* (2016), vol. 35, Wiley Online Library, pp. 67–77. 3

[Jak10] JAKOB W.: Mitsuba renderer, 2010. http://www.mitsuba-renderer.org. 1, 8, 11

[Jia18] JIANG G.: *One Shot Radiance*. Imperial College London PhD thesis, 2018. 3

[Kaj86] KAJIYA J. T.: The rendering equation. *SIGGRAPH Computer Graphics 20*, 4 (Aug. 1986), 143–150. 1, 2

[KB14] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. *CoRR abs/1412.6980* (2014). URL: http://arxiv.org/abs/1412.6980. 8

[KBS15] KALANTARI N. K., BAKO S., SEN P.: A machine learning approach for filtering Monte Carlo noise. *ACM Transactions on Graphics (TOG) 34*, 4 (July 2015), 122:1–122:12. 2, 4

[KMM*17] KALLWEIT S., MÜLLER T., MCWILLIAMS B., GROSS M., NOVÁK J.: Deep scattering: Rendering atmospheric clouds with radiance-predicting neural networks. *ACM Transactions on Graphics (TOG) 36*, 6 (Nov. 2017). 4

[KR17] KALANTARI N. K., RAMAMOORTHI R.: Deep high dynamic range imaging of dynamic scenes. *ACM Transactions on Graphics (TOG) 36*, 4 (2017). 7

[KWR16] KALANTARI N. K., WANG T.-C., RAMAMOORTHI R.: Learning-based view synthesis for light field cameras. *ACM Transactions on Graphics (TOG) 35*, 6 (Nov. 2016), 193:1–193:10. 4

[LADL18] LI T.-M., AITTALA M., DURAND F., LEHTINEN J.: Differentiable monte carlo ray tracing through edge sampling. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 37*, 6 (2018), 222:1–222:11. 13

[LH96] LEVOY M., HANRAHAN P.: Light field rendering. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1996), SIGGRAPH '96, ACM, pp. 31–42. 4

[LRR04] LAWRENCE J., RUSINKIEWICZ S., RAMAMOORTHI R.: Efficient brdf importance sampling using a factored representation. *ACM Transactions on Graphics (TOG) 23*, 3 (Aug. 2004), 496–505. 2, 3

[LRS*18] LIU G., REDA F. A., SHIH K. J., WANG T.-C., TAO A., CATANZARO B.: Image inpainting for irregular holes using partial convolutions. *arXiv preprint arXiv:1804.07723* (2018). 6

[LTH*17] LEDIG C., THEIS L., HUSZAR F., CABALLERO J., CUNNINGHAM A., ACOSTA A., AITKEN A., TEJANI A., TOTZ J., WANG Z., SHI W.: Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 4681–4690. 7, 8, 13

[MGN17] MÜLLER T., GROSS M., NOVÁK J.: Practical path guiding for efficient light-transport simulation. In *Computer Graphics Forum* (2017), vol. 36, Wiley Online Library, pp. 91–100. 2, 3, 4, 5, 9, 10, 11, 12, 14

[MLX*17] MAO X., LI Q., XIE H., LAU R. Y., WANG Z., SMOLLEY S. P.: Least squares generative adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), IEEE, pp. 2813–2821. 7

[MMR*18] MÜLLER T., MCWILLIAMS B., ROUSSELLE F., GROSS M., NOVÁK J.: Neural importance sampling. *arXiv preprint arXiv:1808.03856* (2018). 2, 3, 4, 5, 11

[PJH16] PHARR M., JAKOB W., HUMPHREYS G.: *Physically Based Rendering: From Theory to Implementation*, 3rd ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2016. 3

[RH01] RAMAMOORTHI R., HANRAHAN P.: An efficient representation for irradiance environment maps. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 497–500. 5

[RMZ13] ROUSSELLE F., MANZI M., ZWICKER M.: Robust denoising using feature and color information. *Computer Graphics Forum 32*, 7 (2013), 121–130. 2, 4

[RWG*13] REN P., WANG J., GONG M., LIN S., TONG X., GUO B.: Global illumination with radiance regression functions. *ACM Transactions on Graphics (TOG) 32* (July 2013). 3, 4

[SA07] SUBR K., ARVO J.: Steerable importance sampling. In *Proceedings of the 2007 IEEE Symposium on Interactive Ray Tracing* (Washington, DC, USA, 2007), RT '07, IEEE Computer Society, pp. 133–140. 2, 3

[SD11] SEN P., DARABI S.: *Implementation of Random Parameter Filtering*. Tech. Rep. EECE-TR-11-0004, University of New Mexico, May 2011. 2

[SD12] SEN P., DARABI S.: On filtering the noise from the random parameters in Monte Carlo rendering. *ACM Transactions on Graphics (TOG) 31*, 3 (June 2012), 18:1–18:15. 2, 4

[SJHD18] SIMON F., JUNG A., HANIKA J., DACHSBACHER C.: Selective guided sampling with complete light transport paths. *ACM Transactions on Graphics (TOG) 37*, 6 (Dec. 2018). 3

[SKS02] SLOAN P.-P., KAUTZ J., SNYDER J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Transactions on Graphics (TOG) 21*, 3 (July 2002), 527–536. 5

[SKW*17] SCHIED C., KAPLANYAN A., WYMAN C., PATNEY A., CHAITANYA C. R. A., BURGESS J., LIU S., DACHSBACHER C., LEFOHN A., SALVI M.: Spatiotemporal variance-guided filtering: real-time reconstruction for path-traced global illumination. In *Proceedings of High Performance Graphics* (2017), ACM, p. 2. 2

[SPD18] SCHIED C., PETERS C., DACHSBACHER C.: Gradient estimation for real-time adaptive temporal filtering. *Proceedings ACM Computer Graphics and Interactive Techniques 1*, 2 (Aug. 2018), 24:1–24:16. 2

[Vea97] VEACH E.: *Robust monte carlo methods for light transport simulation*. No. 1610. Stanford University PhD thesis, 1997. 3

[VG95] VEACH E., GUIBAS L. J.: Optimally combining sampling techniques for monte carlo rendering. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques* (1995), SIGGRAPH '95, pp. 419–428. 2, 3

[VK16] VORBA J., KŘIVÁNEK J.: Adjoint-driven russian roulette and splitting in light transport simulation. *ACM Transactions on Graphics (TOG) 35*, 4 (July 2016), 42:1–42:11. 3

[VKK18] VÉVODA P., KONDAPANENI I., KŘIVÁNEK J.: Bayesian online regression for adaptive direct illumination sampling. *ACM Transactions on Graphics (TOG) 37*, 4 (2018), 125. 3

[VKv*14] VORBA J., KARLÍK O., ŠIK M., RITSCHEL T., KŘIVÁNEK J.: On-line learning of parametric mixture models for light transport simulation. *ACM Transactions on Graphics (TOG) 33*, 4 (aug 2014). 2, 3, 5, 9, 10, 11, 12

[VRM*18] VOGELS T., ROUSSELLE F., MCWILLIAMS B., RÖTHLIN G., HARVILL A., ADLER D., MEYER M., NOVÁK J.: Denoising with kernel prediction and asymmetric loss functions. *ACM Transactions on Graphics (TOG) 37*, 4 (2018), 124:1–124:15. 2, 4, 7, 9

[XWCL15] XU B., WANG N., CHEN T., LI M.: Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853* (2015). 8

[ZWW18] ZSOLNAI-FEHÉR K., WONKA P., WIMMER M.: Gaussian material synthesis. *ACM Transactions on Graphics (TOG) 37*, 4 (2018), 76:1–76:14. 4

[ZZ18] ZHENG Q., ZWICKER M.: Learning to importance sample in primary sample space. *arXiv preprint arXiv:1808.07840* (2018). 2, 3, 5