Active Exploration for Neural Global Illumination of Variable Scenes

Stavros Diolatzis, Julien Philip, George Drettakis

Active Exploration for Neural Global Illumination of Variable Scenes

STAVROS DIOLATZIS, Inria & Université Côte d'Azur, France
JULIEN PHILIP, Inria & Université Côte d'Azur, France and Adobe Research, UK
GEORGE DRETTAKIS, Inria & Université Côte d'Azur, France

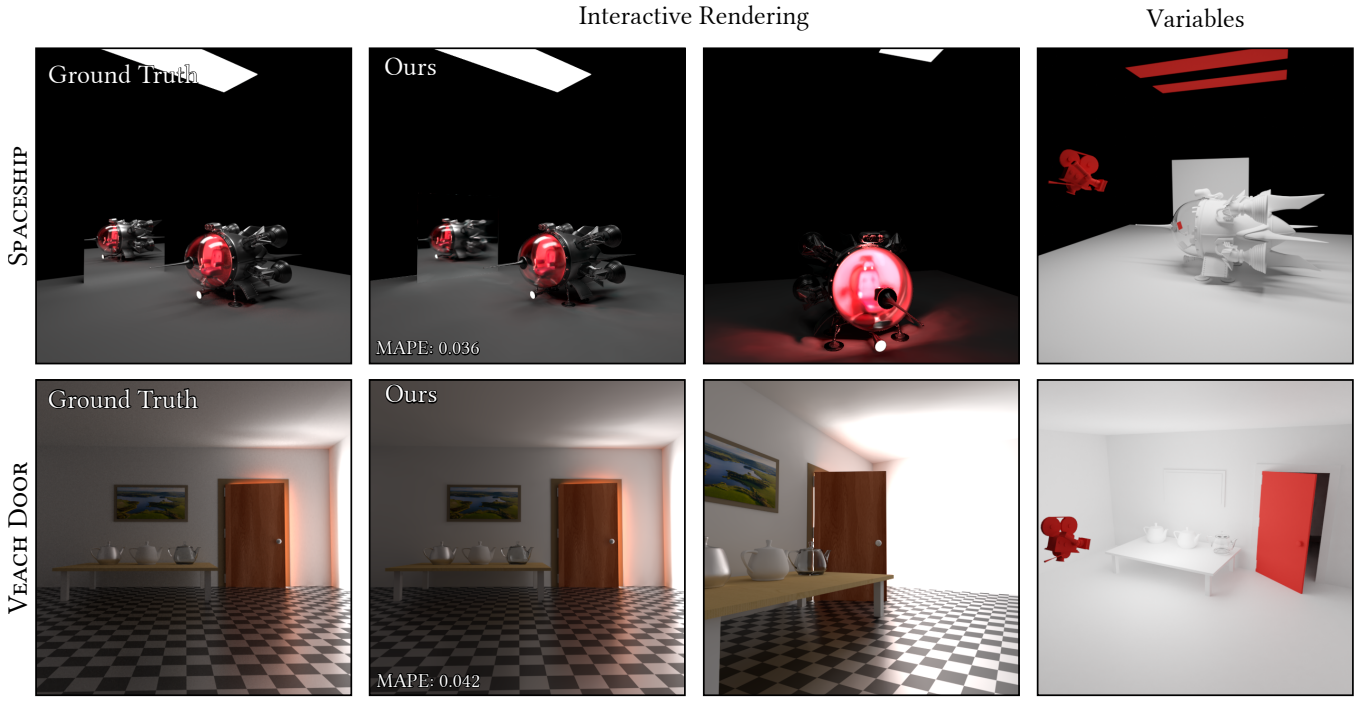

Fig. 1. We introduce a neural rendering method that allows interactive navigation in a scene with dynamically changing properties, i.e., viewpoint, materials and geometry position and full global illumination effects. With our *Active Exploration* we can train a neural network efficiently to learn global illumination for all the configurations of these variable properties, allowing interactive rendering at runtime. Left to right: ground truth path traced images; our prototype interactive neural renderer, running at 4-6 fps with a variation of each scene and the variable parts of the scene depicted in red; each variable property (light intensity, camera position, object rotation, etc.) is controlled by an interactive slider (please see video).

Neural rendering algorithms introduce a fundamentally new approach for photorealistic rendering, typically by learning a neural representation of illumination on large numbers of ground truth images. When training for a given *variable* scene, i.e., changing objects, materials, lights and viewpoint, the space $\mathcal{D}$ of possible training data instances quickly becomes unmanageable as the dimensions of variable parameters increase. We introduce a novel *Active Exploration* method using Markov Chain Monte Carlo, which *explores* $\mathcal{D}$, generating samples (i.e., ground truth renderings) that best help training and interleaves training and on-the-fly sample data generation. We introduce a self-tuning sample reuse strategy to minimize the expensive step of rendering training samples. We apply our approach on a neural generator that learns to render novel scene instances given an explicit parameterization of the scene configuration. Our results show that Active Exploration trains our network much more efficiently than uniformly sampling, and together with our resolution enhancement approach, achieves better quality than uniform sampling at convergence. Our method allows interactive rendering of hard light transport paths (e.g., complex caustics) – that require very high samples counts to be captured – and provides dynamic scene navigation and manipulation, after training for 5-18 hours depending on required quality and variations.

Additional Key Words and Phrases: Computer Graphics, Global Illumination, Neural Rendering, Neural Networks, Deep Learning

ACM Reference Format:
Stavros Diolatzis, Julien Philip, and George Drettakis. 2022. Active Exploration for Neural Global Illumination of Variable Scenes. *ACM Trans. Graph.* 1, 1 (March 2022), 18 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Authors' addresses: Stavros Diolatzis, Inria & Université Côte d'Azur, GraphDeco, Sophia Antipolis, 06902, France, stavros.diolatzis@inria.fr; Julien Philip, Inria & Université Côte d'Azur, GraphDeco, Sophia Antipolis, 06902, France and Adobe Research, London, UK, juphilip@adobe.com; George Drettakis, Inria & Université Côte d'Azur, GraphDeco, Sophia Antipolis, 06902, France, george.drettakis@inria.fr.

1 INTRODUCTION

Neural rendering is an exciting emerging alternative to traditional physically-based rendering; Initial attempts [Ren et al. 2013] were restricted to indirect light with static geometry, while more recent methods [Eslami et al. 2018; Granskog et al. 2020] are trained on

large numbers of rendered images for variable scenes, i.e., with changing objects, materials, lights and viewpoint. Other recent solutions learn encodings of appearance or lighting [Baatz et al. 2021; Nalbach et al. 2017; Zhu et al. 2021], again training on rendered images. Uniformly sampling the space of these path-traced images is expensive; in the case of a high-dimensional space $\mathcal{D}$ containing all the possible configurations of a variable scene, it quickly becomes unmanageable. To address this limitation, we introduce an *Active Exploration* strategy, that guides sampling to parts of the space $\mathcal{D}$ that are most challenging for neural rendering.

We demonstrate the efficiency of our Active Exploration approach on a neural renderer, by training a generator network that can interactively render global illumination with dynamic modifications (moving viewpoint, lights, objects etc.). Training time varies from minutes to hours, depending on the scene variability, complexity and available hardware. To represent a variable scene (see Fig. 1), we use an explicit representation with a vector v of variable parameters that precisely define an instance of the possible scene configurations in $\mathcal{D}$, enabling fine control of each parameter and interactive rendering.

We *interleave* training with on-the-fly generation of the data it needs. Uniformly sampling the space of parameters to generate the data for training does not allow the network to achieve satisfactory visual quality, especially when increasing the dimensions of $\mathcal{D}$, because in many cases light transport has hard, localized effects that have low probability of being observed.

Our Active Exploration method finds samples best suited for training but also locally explores these regions of $\mathcal{D}$ which is crucial in our context especially for enabling high resolution training (6.2), compared to Active Learning (see Sec. 2). For this we use a Markov Chain Monte Carlo (MCMC) approach, with small and large steps and a custom acceptance policy.

Despite our focused Active Exploration, training data generation – i.e., ground truth path-tracing – is still expensive; it is thus beneficial to *reuse* such rendered samples during training. For best results, we introduce a *self-tuning sample reuse* strategy that optimizes the probability for training sample reuse, further reducing training time.

Active Exploration, together with sample reuse and resolution enhancement allow us to train our neural renderer network very efficiently. In contrast, uniform sampling converges to a low quality solution, while our guided exploration of the training space allows us to significantly improve visual quality, especially for reflections and hard light paths. Therefore our renderer is well suited for interactive rendering of effects such as complex caustics or specular-diffuse-specular paths, that are not handled by other real time methods.

In summary our contributions are:

- A novel Active Exploration approach, interleaving training with on-the-fly generation of training data, together with an adaptively increasing resolution method.
- A self-tuning sample reuse approach, further optimizing training time and storage.
- A neural renderer that allows direct control of parameters for global illumination and interactive inference, based on an explicit scene parameterization.

We demonstrate our system that allows interactive modifications of lighting, geometry, materials and viewpoint (at 4-6 fps in our prototype Python implementation, see Fig. 1 and video). We will provide all source code and data on publication.

2 RELATED WORK

We review the most closely related work in traditional and neural rendering, and discuss some aspects of deep learning that inspired our Active Exploration and training sample reuse methods.

2.1 Traditional Global Illumination

Monte Carlo methods, and in particular path-tracing and variants, are now the standard method for realistic, physically-based rendering, used extensively in industry [Keller et al. 2015]. In this paper we use the modern GPU-accelerated Mitsuba 2 path tracer [Nimier-David et al. 2019] for generation of training data.

Markov Chains are used extensively in Monte Carlo rendering since they converge to generating samples according to their importance by only evaluating the contribution of each path. Veach and Guibas [1997] used Metropolis Sampling to explore the space of all possible paths. Kelemen et al. [2002] later applied the same sampling in the space of random numbers that create the paths instead of the paths themselves, i.e., in Primary Sample Space. The space of random numbers that we use to create each scene instance during the data generation of neural rendering has many similarities with this approach: They are both in general high dimensional and for both importance is distributed unevenly in pockets of each space. Also related to our method is the work by Bitterli et al. [2019] that combines a simple path tracing integrator with MCMC by using the random seeds of high variance paths as starting points for the Markov Chains.

Numerous methods have been developed for interactive global illumination approximations [Ritschel et al. 2012]. These methods are typically approximations either of light transport, e.g., simulating only one bounce of indirect lighting [Ritschel et al. 2009] or limiting light transport to a subset of possible paths (e.g., diffuse-only [Nichols et al. 2009]). Dynamic scenes (moving objects or lights) are rarely treated in such methods, e.g., the method of Majercik et al. [2019], that is limited to diffuse scenes.

One popular approach is the use of light probes [Greger et al. 1998; McGuire et al. 2017; Rodriguez et al. 2020], and in some cases limited dynamic behavior is possible (e.g., dynamic lights [Silvennoinen and Lehtinen 2017]). Seyb et al. [2020] provides a solution for a scene with variations – or tunable parameters – using extensive precomputation, demonstrating the importance of this use case in production. For such methods, overall accuracy often depends on probe density and limitations of the probe representation and reconstruction method, and they are usually restricted in the type of light transport supported (e.g., diffuse only). In contrast, we learn complete light transport for a variable scene, avoiding the explicit representation and reconstruction of lighting at test time and using Active Exploration to find the most useful ground truth, fully path-traced samples.

Recent hardware advances, and in particular ray-tracing hardware [Benty et al. 2020; Burgess 2020] open a new horizon for interactive global illumination, including resampling algorithms that greatly accelerate real-time rendering of very complex, dynamic direct lighting [Bitterli et al. 2020]. Nonetheless, complex global light transport paths are hard to compute on-the-fly, suggesting that precomputation-based solutions could be combined with real-time ray-tracing for future interactive Global Illumination algorithms. In the results (Sec. 7) we show examples of difficult light transport configurations that can be rendered interactively with our method.

2.2 Neural Rendering

Using neural networks for rendering is a rapidly growing research topic with applications in many different contexts. The area is vast; we briefly review a few representative papers, and refer the reader to a survey [Tewari et al. 2020] for more information.

The most widespread usage of neural rendering has been in the context of *inverse* problems, i.e., capturing and rendering real scenes. Even though we treat only synthetic scenes, the methodologies developed in this context can be applied in our case. Methods vary from 3D reconstructed geometry-aware solutions for rendering [Hedman et al. 2018; Riegler and Koltun 2020] or multi-plane methods [Srinivasan et al. 2019] to continuous volumetric representations [Mildenhall et al. 2020]. SIREN [Sitzmann et al. 2020] uses sinusoidal activation functions that greatly improve reconstruction quality. Such view-synthesis methods are restricted to the lighting condition at capture. More recent methods allow modification of lighting directly [Philip et al. 2019], and a recent method represents materials, geometry and lighting to estimate and modify lighting [Srinivasan et al. 2021]. X-Fields [Bemana et al. 2020] also map view, time and light coordinates for small-baseline camera motion. These methods use ideas from global illumination methods and have inspired aspects of our work, e.g., our choice of the neural network. They focus mostly on real scenes, while we work exclusively with *synthetic* content.

For such synthetic content, there has been significant effort to improve sampling [Müller et al. 2019] or post-process denoising [Bako et al. 2017; Chaitanya et al. 2017; Gharbi et al. 2019; Işık et al. 2021] for Monte Carlo global illumination algorithms. Denoising methods fail to reproduce complex lighting effects (caustics, specular-diffuse-specular paths) if they do not exist in the noisy input, while our method can reproduce them at interactive rates in high resolution (see Sec. 7.2.3). Deep learning methods have been used to improve importance sampling [Bak 2019; Zheng and Zwicker 2019]. In more recent work Müller et al. [2019] use a neural importance sampler that is trained in an online fashion to guide the training, similar to our exploration scheme. The basic premise for the Noise2Noise approach of Lehtinen et al. [2018] helps guide the level of image quality required for our on-the-fly path-traced training data (see Sec. 4.2)

Our work was inspired by the use of neural networks to replace or augment rendering with global illumination. In pioneering work, Ren et al. [2013] learned indirect illumination with dynamic lights and roughness using neural networks, trained on a fixed set of precomputed renderings. While sharing similarities to our method, our Active Exploration method allows the treatment of more variability (e.g., moving objects) since it efficiently identifies important samples as the dimensionality of the training space increases. Our adaptive resolution approach also allows direct lighting to be learned.

A later solution learned screen-space shading effects [Nalbach et al. 2017]. More recently, Eslami et al. [2018] train an encoder decoder network on a variable scene by encoding observations into a scene representation vector, which is then used to render the scene from a novel view point. Granskog et al. [2020] expand on this idea by using buffers to help the network and by enforcing structure on the neural scene representation. Such representations are compact compared to traditional representations, such as voxel grids and point clouds, scaling more gracefully with scene complexity and size. For Granskog et al. [2020] generating a novel configuration still requires rendering three observations, i.e., full path-traced images.

The Neural Radiance Cache by Mueller et al. [2021] uses a smaller neural network with online training, that is used to query indirect illumination for real-time path tracing. Like denoising methods, it depends on noisy real-time path tracing that often misses difficult light transport paths which our method handles well. Hadadan et al. [2021] propose to minimize the rendering equation residual using a neural network. Similar to us, they use scene information such as material parameters and normals to aid the network, but they cannot handle variable scenes, without retraining, which is the focus of our method. Additionally, during training they uniformly sample positions and incoming directions on the scene surfaces, compared to our Active Exploration.

2.3 Machine Learning

Our on-the-fly data generation, Active Exploration and training sample reuse approach do not have obvious equivalents in related work to our knowledge. However, several sub-fields of machine learning explore ideas with some similarities; we review these briefly.

Active Learning. Parallels can be drawn between our on-the-fly training data generation and Active Learning, where the data generation (labeling) is done procedurally to decrease cost. As reviewed by Settles [2009] in Active Learning an algorithm chooses when a data sample needs to be labeled, i.e., to be given the ground truth. Active learning has been applied to convolutional neural networks [Sener and Savarese 2017] and generative adversarial networks [Zhu and Bento 2017]. Different metrics can be used to define the importance of each sample; some are related to our metrics to identify the most important samples during Active Exploration (Sec. 5.1). Our context of working with synthetic scenes allows us to expand on Active Learning and introduce Active Exploration. We not only identify hard samples for training but we also use mutations to propose new hard samples which helps with catastrophic forgetting [Kemker et al. 2018] and overfitting.

Curriculum Learning. Importance sampling methods with Stochastic Gradient descent have been developed under the general curriculum learning framework [Bouchard et al. 2015]. They learn the probability distribution of choosing a training sample and use it for importance sampling. Similarly Hazan et al. [2011] learn a distribution for picking training data. In contrast to such methods,

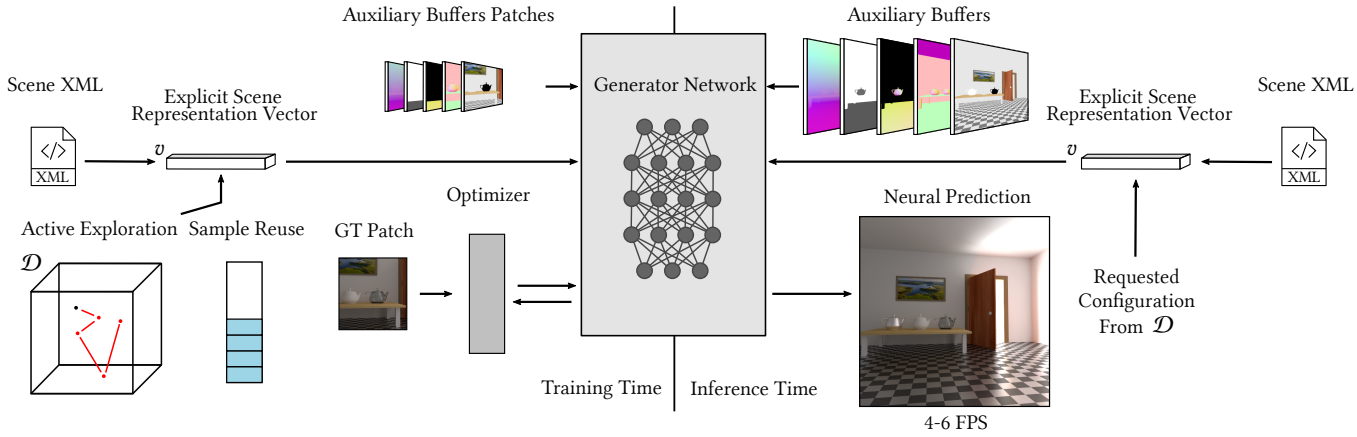

Fig. 2. Overview of our approach. Left: During training we define a scene and the set of variable parameters via an *xml* file resulting in a explicit scene representation vector v. Using Active Exploration we guide the configurations of the *variable scene* towards more difficult instances that are important for the PixelGenerator network. Right: After 5-18 hours of training – depending on the complexity of the variable parameters and quality required – we can *interactively* request any variation of the scene with visual dynamic changes in illumination, move objects, the viewpoint and modify materials.

we know the exact dimensions of our data space and can sample them at will, making the task easier.

Recent work investigates issues with adaptive sampling, and the cost of using the ideal target function [Stich et al. 2017], and provide guarantees about the quality of sampling given limited information on the gradients. There have been some techniques that use self-augmentation with synthetic rendering to overcome the lack of labelled or real-world ground truth data [Kim et al. 2018; Li et al. 2017]; the goal is to match the synthetic and real distributions, which implies different design choices from our context.

Compared to all these methods the major difference is that we have a forward problem, and thus have full knowledge of the parameters that define the space of training data and their dimensions. We can thus sample any part of this space on-the-fly. This aspect of the space of training data makes it amenable to an MCMC exploration method, which is not the case of static, pre-captured training datasets.

Learning and MCMC. MCMC methods have been used in Bayesian learning from the early days of neural networks [Neal 1996]. More recently, Stochastic-Gradient MCMC has been proposed [Welling and Teh 2011; Zhang et al. 2019] with various applications [Li et al. 2016]. We also use MCMC for deep learning, but in a different context: since we solve a *forward* problem and can generate training samples on-the-fly, we use an MCMC approach inspired by Metropolis-Hastings to guide the sampling process.

3 OVERVIEW

Our goal is to significantly improve the efficiency of the training process in neural renderers that are trained on synthetic rendered data by introducing Active Exploration of the high-dimensional sample space and re-using these samples. With this scheme we are able to efficiently train our neural renderer which provides explicit control of the scene parameters and has constant rendering performance regardless of the difficulty in the underlying lighting effect.

We represent the scene variability by an explicit scene parameter vector v (Fig. 2), which defines the space $\mathcal{D}$ of all possible configurations of the scene; thus any $v \in \mathcal{D}$ corresponds to an individual scene configuration. Our goal is to train a network to take a specific v and the set of corresponding G-buffer images (normals, albedo, etc.) as input, and generate full global illumination images (Inference Time in Fig. 2).

When training the network some visually significant effects are very localized in the high dimensional space $\mathcal{D}$. Finding sufficiently useful samples in $\mathcal{D}$ to train our network for those effects is very unlikely using uniform sampling and our limited budget. It becomes more unlikely as the dimensionality of $\mathcal{D}$ grows.

Since we are solving a *forward* problem, we can generate ground truth training samples on-the-fly using a fast path-tracer. A *training batch* will be 16 *samples* each consisting of a 32x32 patch of ground truth, path-traced image, each patch in the batch corresponding to a different configuration of the vector v. Each patch is sampled by a different Markov Chain and rendered in parallel. Using patches allows more efficient exploration of $\mathcal{D}$. In terms of pixels generated, this is equivalent to an image of resolution 128x128.

Even though our path tracer is fast, the cost of generating a training batch is still high. We thus *reuse* training samples as much as possible. We introduce a sample reuse strategy that further improves the speed of training. Training times vary from 5-18 hours depending on the complexity of the variable parameters and the quality required.

Once trained, the generator network allows interactive rendering of dynamic global illumination effects (Fig. 2, right) for the variable scene, e.g., interactively navigating in the scene, opening the door, change lighting etc. (please see video).

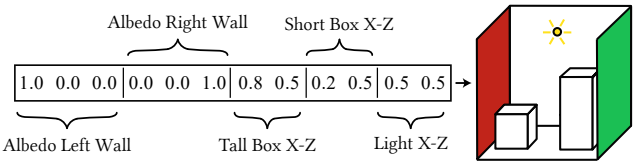

Fig. 3. The explicit scene representation vector v that defines this instance of the variable Cornell box scene.

4 EXPLICIT ENCODING AND ON-THE-FLY DATA GENERATION

We explain the explicit scene representation, the generator network and the on-the-fly data generation process, used in our neural rendering algorithm, before presenting the actual Active Exploration approach in Sec. 5.

4.1 Explicit Scene Representation

Previous methods [Eslami et al. 2018][Granskog et al. 2020] use an encoder network to create a neural scene representation vector of a scene configuration (Sec. 2.2).

However, this representation lacks interpretability and editability. In addition, rendering a new scene configuration requires new observations (i.e., ground truth renderings) to be generated, since the parameters of the scene representation have no explicit interpretation or "meaning", and thus the renderings are needed to generate the new neural scene representation vector.

We focus on variable scenes, commonly used in production [Seyb et al. 2020]. Given that we know explicitly which parts of a scene are variable and how much they can vary, we avoid training an encoder network to represent this variability and instead create the scene representation vector from the scene definition. As a result all fixed properties are stored in the generator and associated with a set of inexpensive G-buffers, while scene variability is compactly represented in the explicit vector. This vector contains the normalized values of the variable scene parameters for a given scene instance.

Consider a variable Cornell box scene (Fig. 3). Here we vary the two vertical wall albedos, positions of the boxes and light source position, and define the ranges of these parameters.

The normalized parameter values make up the explicit scene representation vector v (Fig. 3). The scene representation vector along with the camera position and lookat vector are repeated along the width and height dimension to be the same size as the G-buffers, and also passed to the neural network. Since our generator operates on a per pixel basis, this repeated vector injects the global scene information to all pixels.

4.2 Network Architecture, Buffers and Training Data

We optimize a modified PixelGenerator architecture [Granskog et al. 2020; Sitzmann et al. 2019] (a Multilayer Perceptron network with skip connections) to map the inputs for each pixel to the final pixel color value. We choose this over a convolutional neural network such as a UNet since [Granskog et al. 2020] has demonstrated the PixelGenerator architecture to perform better at upscaling. Unless stated otherwise, we use 512 hidden features and 8 hidden layers. For the optimization we use the ADAM [Kingma and Ba 2015] optimizer with learning rate 1×10^{-4}.

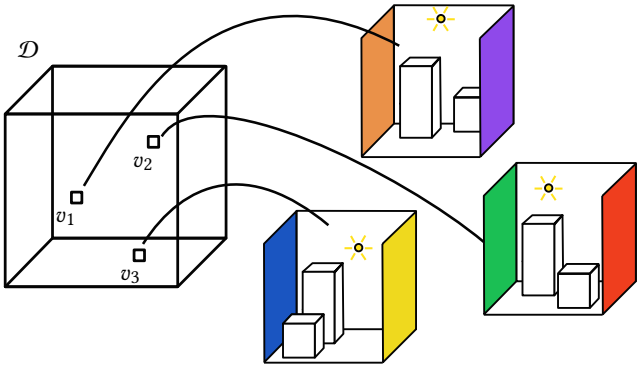

Fig. 4. A point v_i in the data space $\mathcal{D}$ defines a scene instance out of all the possible configurations of the *variable* scene.

For the G-buffers, we provide all the information a traditional path tracer would require to evaluate the rendering equation of path tracing. The Rendering Equation gives outgoing radiance L_o:

$$L_o(x, \omega_o) = L_e(x, \omega_o) + \int_\Omega L_i(x, \omega_i)\, \rho(x, \omega_o, \omega_i)\, \cos\theta_i \, \mathrm{d}\omega_i \quad (1)$$

with L_e the emitted radiance and θ_i the angle between the surface normal and the incoming direction ω_i; We create first-intersection G-buffers with the world position of the intersection x, normal of the surface n, reflectance and roughness of the BSDF ρ and outgoing direction ω_o. The normal and material information help the network understand the existing correlations between these signals and outgoing radiance L_o.

We optimize the neural generator to map this input to the value of the integration over the hemisphere. Emission is also computed as a first-intersection buffer and is passed through to the output directly.

The world position x conditions all the other inputs since it is where the integration happens. For this reason we precondition the PixelGenerator to the position G-buffer by passing it alone through the first network layer. In subsequent layers all the buffers are concatenated with the global information of the scene representation vector and passed to the network; this is similar in spirit to NeRF [Mildenhall et al. 2020] that inputs only position to the first layers. We experimented with Fourier features [Tancik et al. 2020], but this resulted in artifacts due to the noise in the training data. We show the effect of this choice in Sec. 7.3, Fig. 16.

5 ACTIVE DATA SPACE EXPLORATION

For a given variable scene, we will optimize a neural generator using on-the-fly synthetic training data; we describe the training process and loss in Sec. 6. This training data is generated within the space $\mathcal{D}$ of all possible configurations of the scene.

Each point in this space is defined by the values of the scene variables of the explicit scene representation vector v. Since the

scene variables are normalized, the data space $\mathcal{D}$ can be seen as a hypercube, see Fig. 4.

A uniform random sampling of this high-dimensional space converges to a local minimum with low quality (see Sec. 7.3).

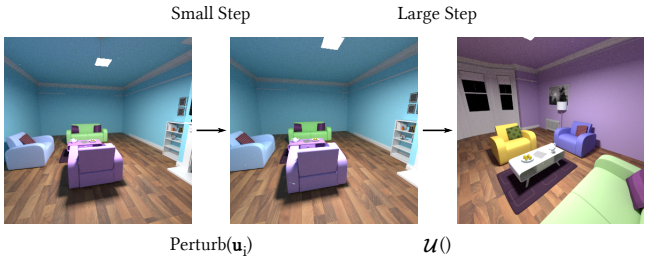

Fig. 5. Visualization of the impact of a small and large step on a variable scene. In small steps (left) minor perturbations are applied – here the light source, furniture positions and materials have been altered slightly. In large steps (right), major changes have been applied to the scene (position of furniture, albedo of objects etc.).

To overcome this difficulty, we propose *Active Exploration* of the space $\mathcal{D}$ of training samples. The ability to generate on-the-fly training samples defined by the explicit vector v offers great flexibility, allowing us to *interleave* sample generation and training.

The high-level goal is to find a sampling strategy that will find samples in $\mathcal{D}$ that maximize the progress of training and locally explore these pockets of importance. We introduce a MCMC exploration strategy that guides sampling of $\mathcal{D}$, towards scene configurations where the network struggles to recreate global illumination. MCMC is well suited to searching such high-dimensional spaces, and has proven its utility both in learning [Welling and Teh 2011] and illumination [Veach and Guibas 1997].

The Markov Chain is initialized with a state picked uniformly from the hypercube of the data space $\mathbf{u} = \mathbf{u}_0 \in \mathcal{D}$, i.e., a random configuration of the variable scene. The next proposed state is sampled from the proposal distribution $\mathbf{v} \in T(\mathbf{u}_i \rightarrow \mathbf{v})$. Similar to the Primary Sample Space exploration [Kelemen et al. 2002] we balance global and local exploration of the space with large and small steps, by choosing large steps with probability p_{LS}. Specifically:

$$T(\mathbf{u}_i \rightarrow \mathbf{v}) = \begin{cases} \mathcal{U}() & \text{with probability } p_{\text{LS}} = 0.3 \\ \text{Perturb}(\mathbf{u}_i) & \text{else} \end{cases} \tag{2}$$

5.1 Markov Chain Exploration

The data space $\mathcal{D}$ can have arbitrarily high dimension, depending on how much variability exists in the scene. Our goal is to generate training samples that follow the distribution of sample importance, i.e., the impact of the sample on training. In MCMC terminology, our target function f and corresponding target distribution p should be defined so that the sampling process produces samples that maximize benefit for training.

The high dimensional space of $\mathcal{D}$, with pockets of importance, is ideal for a MCMC random walk exploration.

The hypercube of our data space $\mathcal{D}$ has a very similar structure to the primary sample space [Kelemen et al. 2002] and we take inspiration from the exploration choices of that method. The Metropolis-Hastings algorithm defines a proposal distribution $T(\mathbf{u}_i \rightarrow \mathbf{v})$ from a given state $\mathbf{u}_i$ to a proposed state $\mathbf{v}$. The target distribution p is defined such that new states should be proposed and accepted for the Markov Chain to have a stationary distribution (i.e., the distribution at convergence) proportional to the target function.

Our goal is to define a target distribution that will guide the training process to samples that accelerate training. Previous work has suggested different metrics of sample importance [Zhao and Zhang 2015]. Two common such metrics are the training loss or the norm of the gradients after a backward pass.

Our experiments showed that if only the loss is used, MCMC doesn't take into account where the network can improve the most.

However, the *product* of the loss and the norm works well, see Fig. 17. Since we use ADAM [Kingma and Ba 2015], instead of the norm of the gradients we use the norm of the total step to take into account the momentum and RMSprop [Tieleman and Hinton 2012].

The small step involves applying normally distributed perturbations to each component of $\mathbf{u}_i$. A visualization of the impact of these steps on the final rendering is shown in Fig. 5. Since the proposal distribution is symmetric, meaning $T(\mathbf{u} \rightarrow \mathbf{v}) = T(\mathbf{v} \rightarrow \mathbf{u})$, the acceptance probability of the proposed state similar to the Metropolis-Hastings algorithm is:

$$\alpha(\mathbf{u}_i \rightarrow \mathbf{v}) \;=\; min\left(1, \frac{p(\mathbf{v})}{p(\mathbf{u}_i)}\right) \tag{3}$$

The acceptance probability transforms the Markov Chain's stationary distribution to the target distribution. In our case we have a) an evolving target distribution that b) changes based on the samples we provide.

In our experiments, the acceptance probability of Eq. 3 does not converge to the target distribution fast enough, i.e., before it has changed. For this special case we propose instead a more aggressive acceptance policy:

$$\alpha(\mathbf{u}_i \rightarrow \mathbf{v}) \;=\; \begin{cases} 1 & \text{if } p(\mathbf{v}) > p(\mathbf{u}_i) \\ 0 & \text{else} \end{cases} \tag{4}$$

This acceptance probability has the desirable property that the more we remain in a state the more the target function – which is related to the error – decreases for this state. If we assume that the network can represent this state, then it will learn from it, meaning that the gradients and error will decrease, allowing a new proposed state to be accepted. If there are states that cannot be represented (e.g., pixel perfect reflections) the gradients will guide the MCMC towards states that still have room for improvement avoiding the issue of getting stuck.

In the initial phase of data generation, known in the literature as the burn-in phase, the Markov Chain the samples do not follow the target distribution. To alleviate this issue we use 16 Markov Chains in parallel, one for each patch rendered, leading to a shorter burn-in phase. This can be seen in Fig. 6 "MCMC Samples".

We evaluate our proposed acceptance policy and the sample distribution in a simple scenario shown in Figure 6 and by disabling

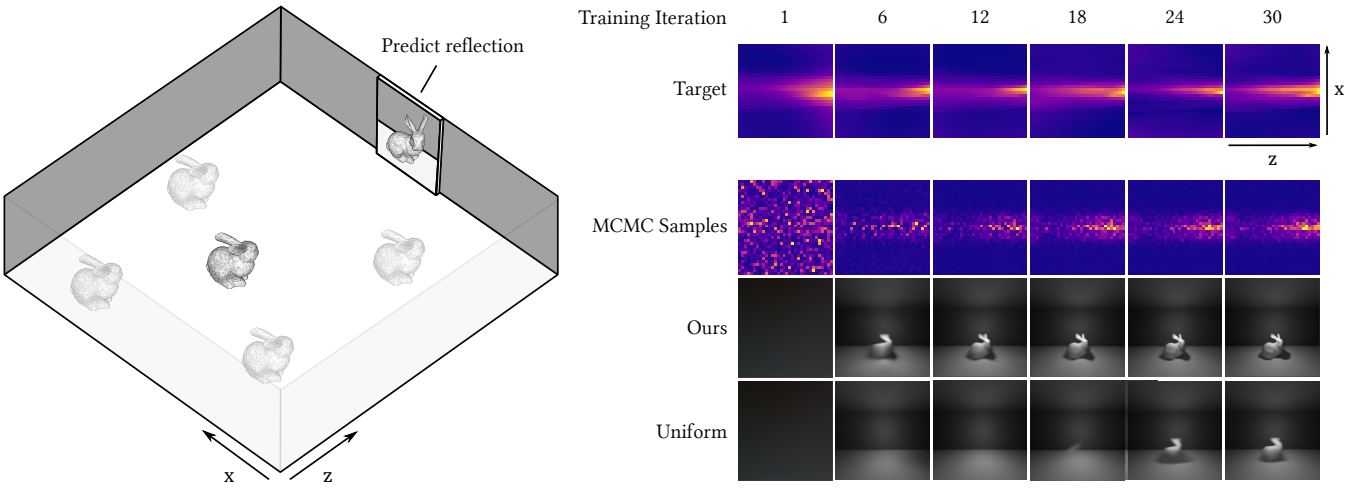

Fig. 6. We test our method in this simple example to verify the convergence of the samples generated by Active Exploration. In this scene the variable parameters are the X-Z placement of a Bunny figure in an empty room (left). We fix the viewpoint to always look into the mirror and ask our generator to predict the reflection. The Bunny appears in the reflection for only a specific range of X values. We plot the target function (loss times gradients) for different X-Z values of the Bunny position (the heatmap can be seen as a top down view of the room) through training iterations, on the right. We show that the 2D histogram of Bunny placement from our Ative Exploration, after a burn in period, starts following the distribution of the target function. As a result the reflection of the Bunny starts appearing much sooner compared to uniform sample generation.

sample reuse. Here the 2D variable parameter is the position of a Bunny figure in a room with a mirror at the center of its wall. We task the generator with predicting the reflection (viewpoint is fixed to always look into the mirror). For this case the static reflection of the walls is learned quite easily but the variable reflection needs the Bunny to be placed in view of the mirror. Our method correctly does so and leads to its reflection appearing much faster compared to Uniform sampling.

6 TRAINING AND SELF-TUNING SAMPLE REUSE

For training, we use the combination of L_1 and structural dissimilarity loss, as in Granskog et al. [2020]. Since rendering is still slow it is beneficial to reuse samples as much as possible. We next discuss our self-tuning sample reuse and resolution enhancement methods.

6.1 Self-tuning Sample Reuse

Traditional supervised deep learning typically uses a fix sized precomputed dataset and runs optimization steps many times on batches, running through the entire dataset several times. Each such run is referred to as an epoch, resulting in the reuse of each data point many times.

In our case, we are generating training samples on-the-fly, and thus we do not have the notion of epochs. However, sample generation is costly (typically 2.5 sec for the 16 32x32 patches), and it is thus important to reuse training samples as much as possible, to speed up training, and also prevent the network from forgetting over the course of training. We do this by introducing a new self-tuning sample reuse strategy based on the divergence between the loss of newly seen data points and those already seen, to achieve a balance between overfitting and training speed.

Inspired by these observations, we achieve this balance by tracking two different losses $Loss_{\text{new}}$ and $Loss_{\text{exist}}$, i.e., the loss of newly generated, unseen samples and the loss of the previously generated samples that were already used to train the network. Both are tracked using an exponential moving average to lessen the effect of the stochasticity of the optimization process. When $Loss_{\text{exist}}$ starts decreasing faster than $Loss_{\text{new}}$, thus diverging from it, our model is starting to over-fit (as new data is performing worse than previously generated data). In this case we need new samples to augment the size of our dataset. This can be done by decreasing p_s, i.e., reusing with a lower probability.

We start training on-the-fly, generating and storing a new sample for the first 100 samples. After this short initialization, for each new step we randomly decide to reuse a previously generated sample, or generate and store a new one. The decision is made based on a Bernoulli distribution with a self-tuning probability p_s over steps s, representing the probability of reusing a training sample.

We build probability p_s to satisfy two goals. First we would like p_s to be as high as possible, so that we save as much computation as possible. But if it is too high, or even equal to 1, we would over-fit to the already generated samples and stop exploring the space of parameters. Thus p_s should also be sufficiently low to avoid over-fitting. Over-fitting is usually measured by the difference of performance of a model between a training and validation dataset.

We propose a mechanism with a single parameter to control p_s:

$$p_s = \sigma(Loss_{\text{exist}} - Loss_{\text{new}} + \beta) \tag{5}$$

where σ represents the sigmoid function and β is the parameter controlling the reuse probability when both losses are equal. This formulation decreases the probability of reusing a sample when $Loss_{\text{exist}}$ is lower than $Loss_{\text{new}}$. Intuitively the above equation is

derived by associating the losses to negative log-likelihood of probability distributions parameterized by the ground truth. More details can be found in the supplemental materials. Since one component of the MCMC target function maximizes the loss (see Sec. 5.1) we use only large step samples to keep track of both $Loss_{\text{exist}}$ and $Loss_{\text{new}}$. In all our experiments β is set to 4.6, to have $p_s = 0.99$ when $Loss_{\text{new}} = Loss_{\text{exist}}$. When a sample is reused we build a batch of training images from the stored patches. We use the previous loss of the sample as a weight, i.e., setting the probability of selecting a patch proportional to its last recorded loss. We update the weight of a sample whenever it is reused, using the network loss on that sample in the current iteration. This allows hard samples to be reused more often and discards those for which the network performs well, leading to better adequacy between reuse and MCMC.

6.2 Resolution of Training Images

One of the main advantages of using a PixelGenerator for the generator architecture, as demonstrated by [Granskog et al. 2020], is its performance during inference on much higher resolutions than that used for training. Shading effects that depend heavily on G-buffers such as textured diffuse materials benefit from buffer upscaling, providing improved quality. That is less true for high frequency view dependent effects, such as reflections, that are are typically small, and band-limited by the resolution of training images. Our goal is to progressively reduce the area each training sample covers, allowing the model to gradually focus on such effects.

We adopt a multi-resolution approach to address this. We start training with 32x32 patches extracted from 128 by 128 pixel images with a 90° field of view. Note that the MCMC controls the respective patch position on the image plane and that we only render the patch pixels. We then progressively increase the resolution of the images used to select the 32x32 patches; we found that doing so by 4 pixels every 2000 iterations worked well, all the way up to 600 by 600 which is closer to our target resolution. This shrinks the area of the patch on the sensor and allows the network to observe finer details in hard regions, such as reflections, during training. This process is made possible by the MCMC exploration, due to its ability to locally explore the scene configuration through the small steps, and to adapt to this progressive change in resolution. On the other hand, adopting such a multi-resolution approach with uniform sampling of $\mathcal{D}$ results in worse overall results as it decreases the probability of observing a given point in the scene. This makes sampling even less efficient (see Fig. 7) resulting in lower perceived image quality.

We show results for several variable scenes; please see the supplemental videos to best appreciate the quality of our results in these dynamic scenes. We also present comparisons to previous work and analyze the various design decisions of our solution through ablations studies and quantitative evaluation.

We have implemented our system in Python interfaced to Mitsuba 2 [Nimier-David et al. 2019] which we use to render global illumination and G-buffers. We use between 200 and 24,000 samples per pixel for ground truth renderings, depending on the scene see Table 2.

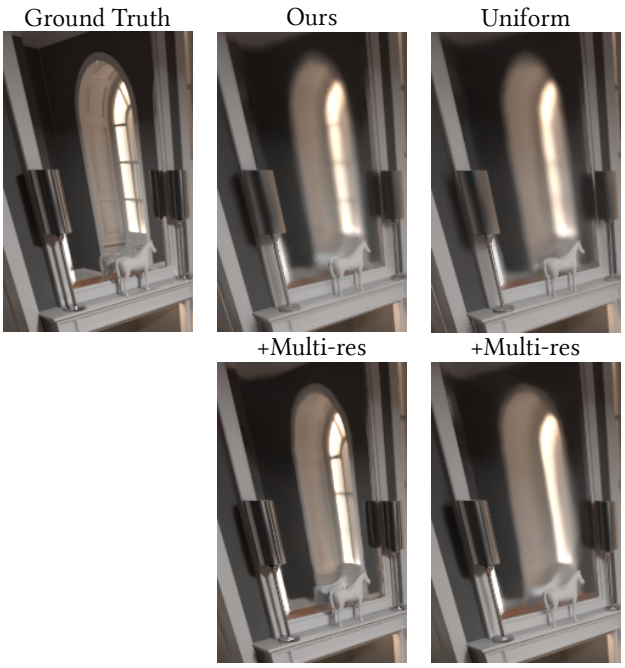

Fig. 7. Ablation study for adaptive resolution MCMC vs Uniform training. First row: the resolution is always 128x128. Second row: we progressively focus resolution, improving quality.

We allow transformation of geometry and lights, material editing and viewpoint changes, including discrete events (e.g., changing between different materials, objects appearing/disappearing).

Our prototype implementation runs at 4-6 fps at inference/rendering time (900x900 resolution on a NVIDIA 3090 GPU), allowing interactive exploration of dynamic global illumination in variable scenes with potential applications in architecture, design, games, etc.

Currently we only show results with a forward path tracer (the only integrator available in Mitsuba 2). However, our method is agnostic to the type of integrator and if we used a different renderer, we could train with bi-directional path tracing, Metropolis or any other method.

7 RESULTS, ANALYSIS AND COMPARISONS

7.1 Results

We present results of our method on several modified scenes from the Bitterli dataset [Bitterli 2016]; the viewpoint is variable in all 7 scenes except Sphere Caustic, Figures 1 and 8.

For the Bathroom scene, we added a showerdoor with variable roughness; additional variables are the intensity and position of the light source (total 8 dimensions). For the Living Room scene, we added blinds on the windows that can open and close; additional variables include the light intensity (7 dimensions). For the Bedroom scene, we have simulated variable sunlight with a distant source coming in through the window (6 dimensions). We present a modified version of the Veach Door scene, where the variable is the opening door (6 dimensions). We also have a modified Cornell Box, Sphere Caustic with variable wall colors and a large sphere that

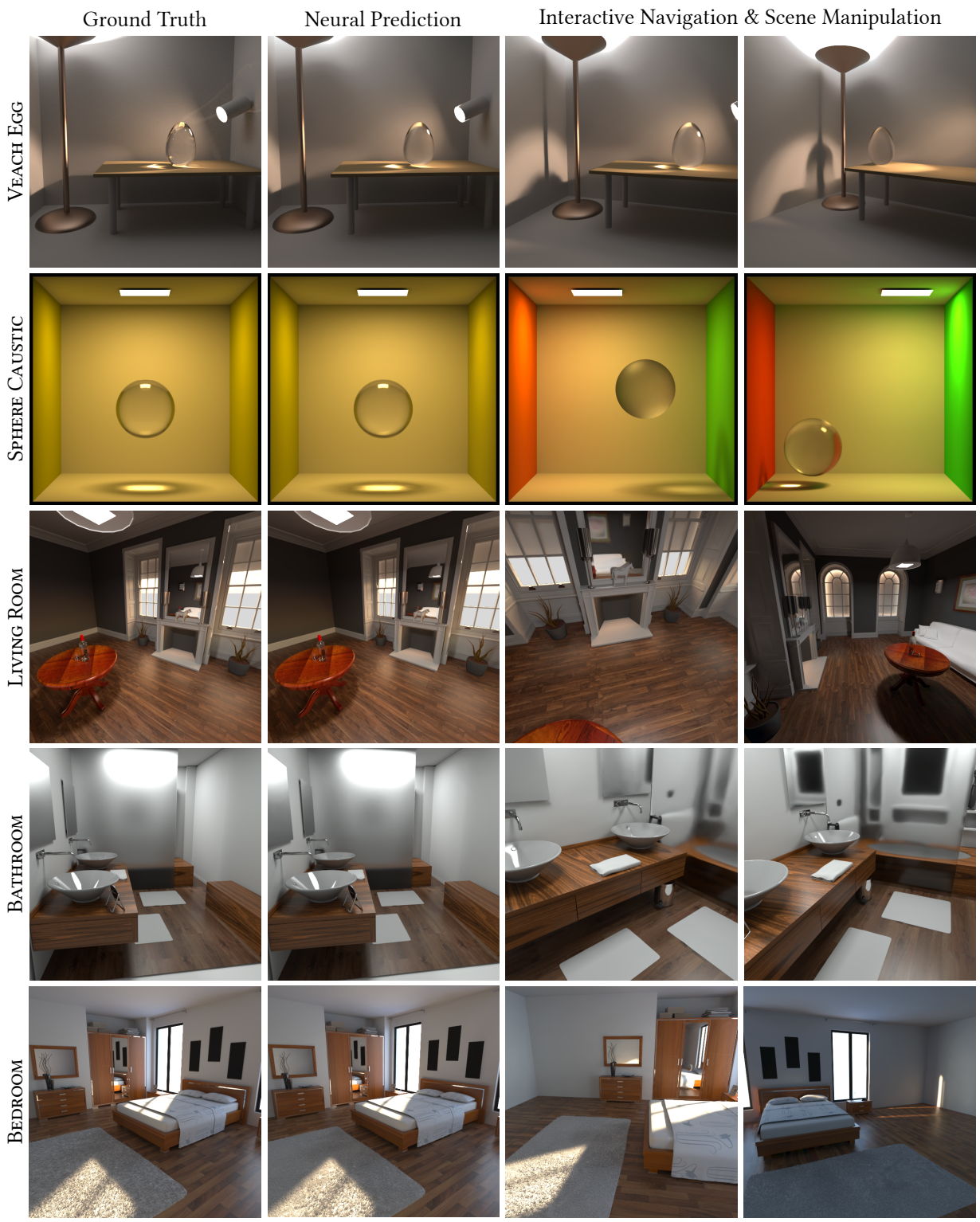

Fig. 8. Results of our method for 5 different scenes each with different variations. Note how we can capture view changes (all but row 2) reflections, approximate caustics, global illumination etc. all at interactive rates. The scene variables include: material albedos and roughness (Sphere Caustic, Bathroom), moving and rotating objects (Veach Door, Living Room) and time of day Bedroom. Please also see supplemental videos.

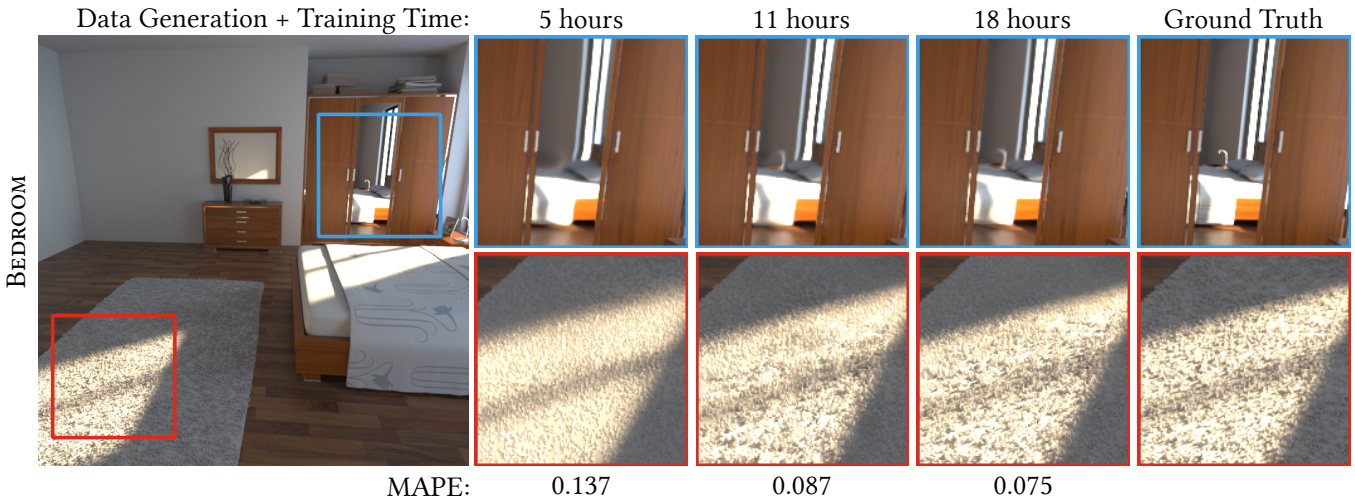

Fig. 9. Results of our method after increasing hours of training. Depending on the application if speed is valued over quality, our method yields plausible results after a few hours of training and data generation. For the best quality possible our method requires around 18 hours in the Bedroom scene.

can move in the scene and vary in roughness, for a fixed viewpoint (11 dimensions). The Spaceship scene contains 3 variable emitters, 2 on the ceiling and one in the cockpit and variable viewpoint (8 dimensions). Finally in the Veach Egg scene we can vary the position of the glass egg and the spotlight emitter (9 dimensions). We show several configurations of each scene in Fig. 8 and Fig. 1 and a path with variations in supplemental. We train the scenes for 5-18 hours on a single NVIDIA RTX 6000. If training speed is important, we obtain a reasonable first approximation after a few hours, but longer training is required if we want to be very close to ground truth (see Fig. 9).

Different application scenarios have different hardware resources. If the target platform is a modern desktop computer with a ray tracing GPU, specular interactions can be traced and not inferred. In this case our method needs less than an hour for acceptable results

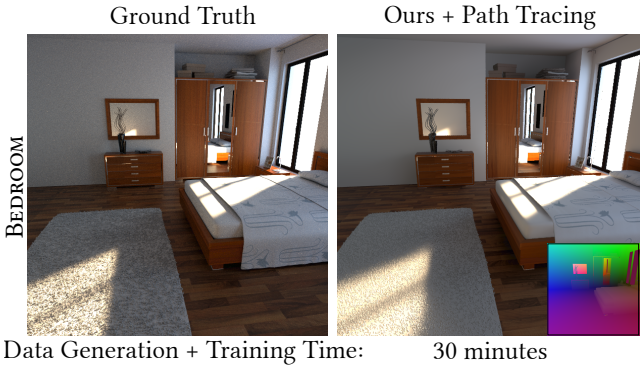

Fig. 10. When ray tracing hardware is available our method benefits by tracing all specular bounces during the buffer generation (positions buffer in inset). This means that with only 30 minutes of data generation and training our method learns the non specular shading for the Bedroom scene. The harder high frequency details on the carpet still need full training to appear.

more training is required to achieve the highest possible quality as seen in Fig. 10.

Scene		DSSIM	MAPE	MAE	LPIPS
Veach Egg	Ours	0.0141	0.079	0.20	0.0245
Sphere Caustic		0.0012	0.031	0.01	0.0023
Living Room		0.0068	0.048	0.04	0.0220
Bathroom		0.0029	0.033	0.03	0.0089
Bedroom		0.0149	0.074	0.05	0.0512

Table 1. Quantitative results using 4 metrics for the configuration shown in Figure 8.

Our solution shows good temporal stability (see supplemental and video). The results show that we can capture a wide variety of light transport effects: global illumination (Living Room with different blind positions; Fig. 8), soft shadows, glossy (or even partially specular) reflections (Living Room, transmission (Bathroom), caustics, (Sphere Caustic, Spaceship, Veach Egg) etc. A major strength of our approach is that we can render very hard light paths with good quality at interactive rates, e.g., the caustic in Spaceship (Fig. 1, or the shadow from the caustic in Veach Egg, Fig. 15, last row). The quantitative results in Tab. 1 show that we achieve low error rates in all scenes.

7.2 Comparisons

The most significant comparison we will present is to Uniform sampling, since this clearly reveals the advantage of our active exploration approach. We also compare to Compositional Neural Scene Representations (CNSR) [Granskog et al. 2020], since we share similar inputs and some goals. The comparison mainly shows the benefits of our Active Exploration, explicit scene representation, and sample reuse in terms of training and inference speed. Finally a compelling alternative to our method for real time rendering of

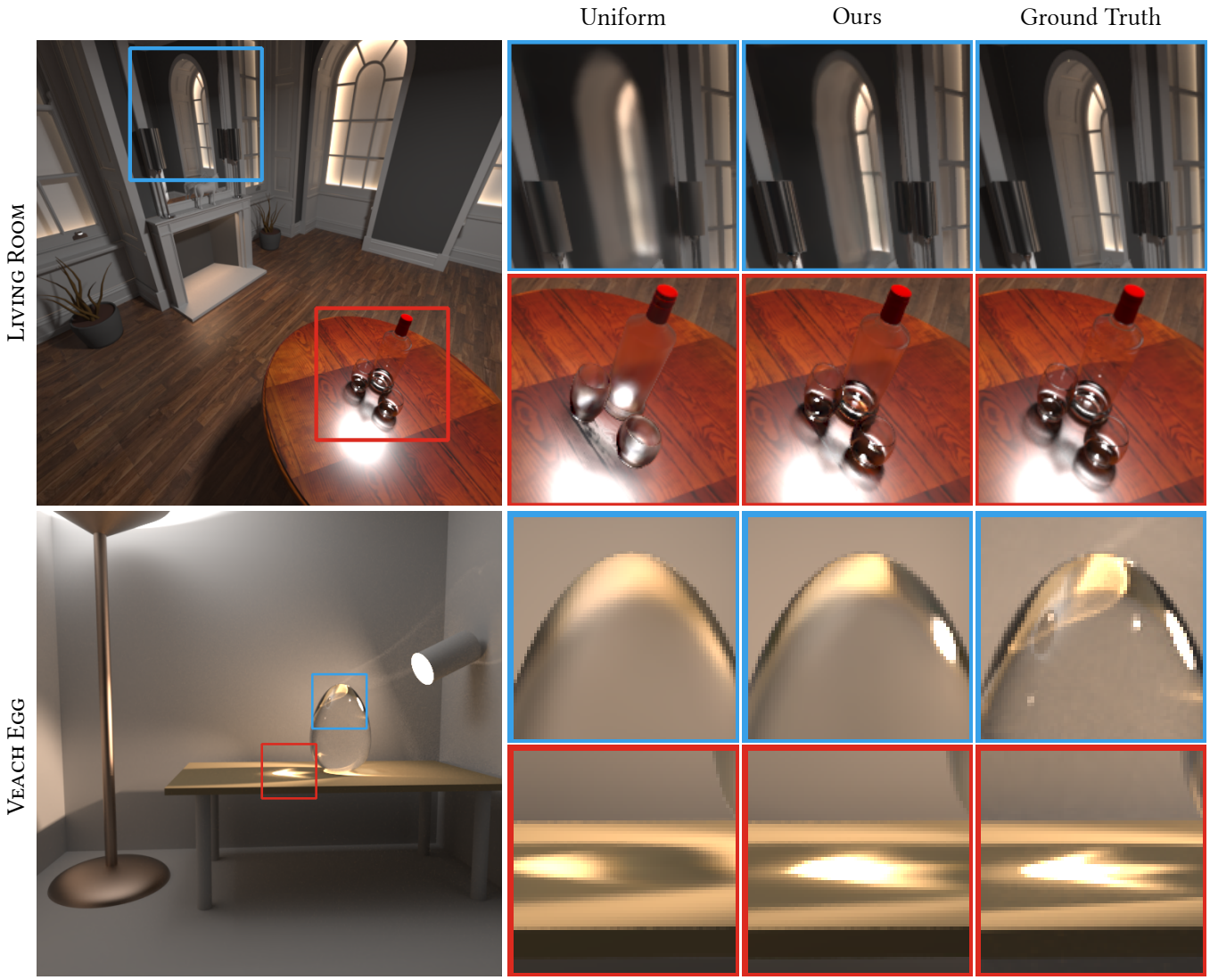

Fig. 11. We compare our active exploration MCMC method vs. Uniform sampling of the space $\mathcal{D}$ trained for the same time. We see that Uniform search running for the same time cannot produce sharp shadows, reflections and caustics.

Table 2. Samples per pixel used for each scene during training.

Scene	Sphere Caustic	Living Room	Bed room	Veach Door	Bath room	Space ship	Veach Egg
spp	200	400	400	600	800	1200	24000

Scene		DSSIM	MAPE	MAE	LPIPS
Living Room	Ours	**0.0141**	**0.079**	**0.20**	**0.0245**
	Uniform	0.0241	0.162	0.28	0.0652
Veach Egg	Ours	**0.0116**	**0.065**	**0.22**	**0.0477**
	Uniform	0.0147	0.076	0.25	0.0738

Table 3. Quantitative results using 4 metrics for the configuration shown in Figure 11.

dynamic scenes is Real Time Path Tracing plus denoising. We compare with the state of the art denoising method of [Işık et al. 2021], further illustrating that our method is one of most efficient solutions for interactive rendering of hard light transport configurations, that require a very high sampling rate to be captured by path-tracing.

7.2.1 Comparison to Uniform sampling. To evaluate the effect of MCMC active exploration our first comparison is to a simple uniform sampling baseline (in Fig. 11). To simulate uniform sampling, we replace our MCMC method with large steps only, that are always

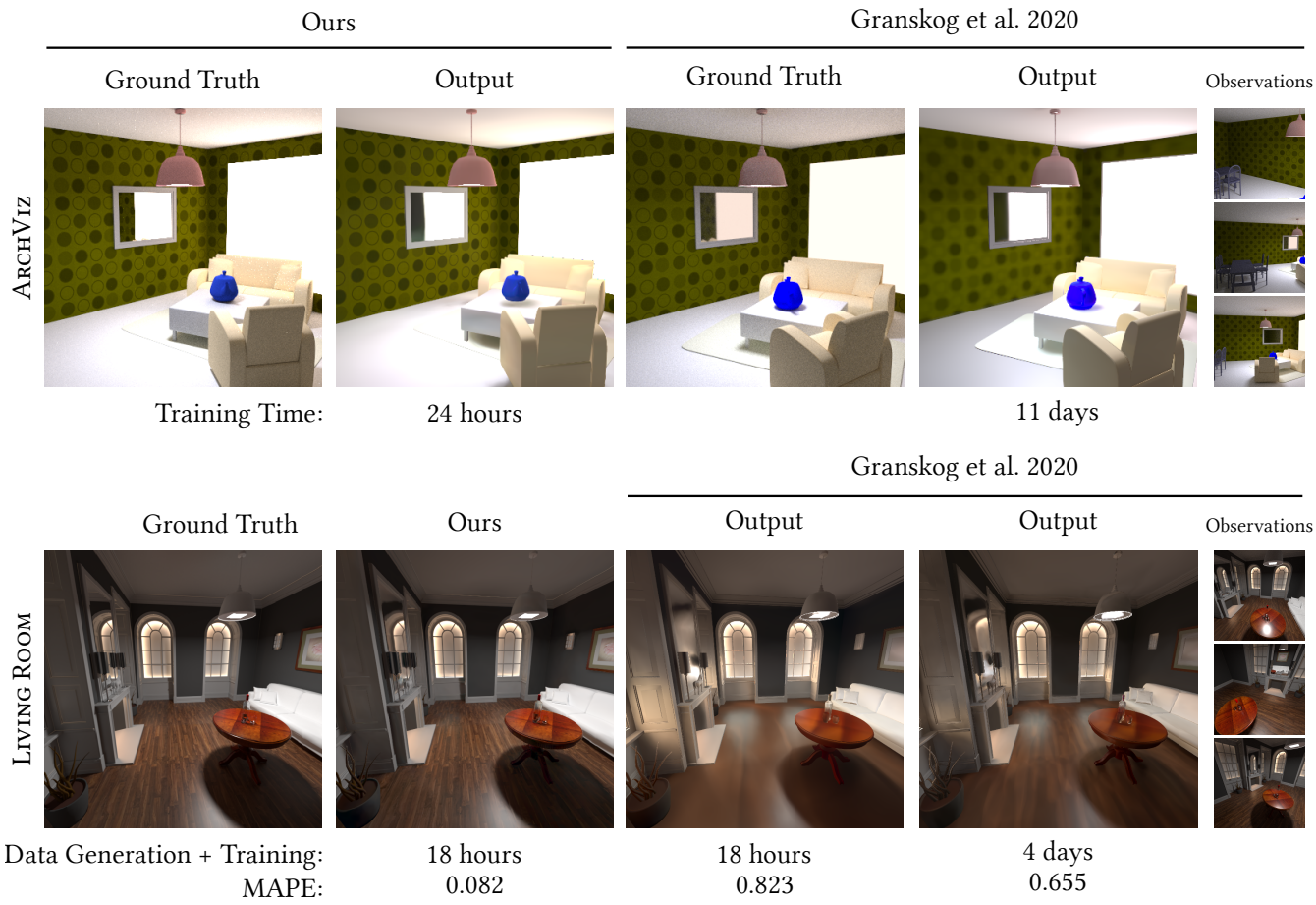

Fig. 12. Same quality (top) and same time (bottom) comparison with Granskog et al. [2020]. We show result of ArchViz for same quality as ours using the pretrained network provided by the authors in their rendering framework. We also show Living Room for same time as ours by training their method on our variable scene in our rendering framework. The 3 path traced observations required by Granskog et al. [2020] are shown on the right in both cases.

accepted; note that this baseline includes our sample reuse method, but not resolution adaptation since it gives worse results (Sec. 7.3). As we can see, for the same computation time, active exploration achieves sharper reflections, caustics and shadows, thanks to the guiding sampling it affords. We tried to obtain equal quality with the uniform sampling, however this naive approach converges to a low quality local minimum. The results shown in Fig. 11 were generated with the best quality this approach could achieve; after this point in training the loss does not decrease. The videos in supplemental showcase the improvement Active Exploration provides in all different configurations of the scenes, e.g., the caustics in Spaceship and Veach Egg, glossy effects in Bathroom, and reflections in Bedroom or Living Room. In all cases, our method provides sharper results, generally much closer to the ground truth. This is confirmed with quantitative analysis in Tab. 3.

We also show quantitative results in Fig. 13 using the Mean Absolute Percentage Error (MAPE), DSSIM [Loza et al. 2006], Mean Absolute Error (MAE) and LPIPS [Zhang et al. 2018] error metrics and a graph with the evolution of error over time. Since our main goal is to handle difficult lighting configurations, we select 10 frames from each path of each scene which correspond to such cases, and

	Bedroom DSSIM	Bedroom MAPE	Living Room DSSIM	Living Room MAPE	Veach Door DSSIM	Veach Door MAPE
Ours	**0.0097**	**0.0458**	**0.0180**	**0.0892**	**0.0117**	**0.0712**
Uniform	0.0160	0.0791	0.0212	0.1058	0.0135	0.0752

Fig. 13. Quantitative evaluation of our method and ablations compared to ground truth (graphs start at 2h of training.)

evaluate our method against ground truth; we show the selected frames for each scene in supplemental.

7.2.2 Comparison to CNSR. We compare our method to Compositional Neural Scene Representations [Granskog et al. 2020] (CNSR) using the variable ArchViz scene, the more complex of the two datasets used in CNSR; Our implementation of this scene has 71 dimensions. For best-effort same quality comparison we use a pretrained model provided by the authors. Note that the ArchViz dataset "consists of variations of a living room with a dining area" [Granskog et al. 2020]. Both the pretrained model of Granskog et al. [2020] and ours are trained on identical data involving variations of this scene. We recreated the ArchViz variable scene in our framework as closely as possible, using publicly available resources [Granskog et al. 2020]. The CNSR pretrained model is trained on a dataset of 9000 sample points. Each point includes 16 batches of 3 observations at 64x64 resolution and a query image at the same resolution, trained for 1M iterations.

The high complexity of this scenes' variations (constrained, specific configurations, e.g., the teapot appears at a specific position on the table etc.) challenges our base method; we show results using 256 features/layer and without resolution enhancement. This gives blurrier results (on a par with Granskog et al. in terms of quality) but avoids high frequency artifacts. Our method achieves same qualitative results with *36 hours* of *both training and rendering*. In comparison Granskog et al. [2020] needs *11 days of only training* (accounting for hardware differences), and an unspecified amount of rendering time to generate the data.

We also retrain CNSR on three of our scenes, providing same time comparisons on Living Room, Bedroom and Veach Door. For this we used the publicly available code provided by the authors to train on data generated by our framework. As in the case of the ArchViz scene we use 16 batches of 3 observations at 64x64 resolution and a query image at the same resolution to train their model.

The results of the same quality ArchViz and same time Living Room comparisons with Granskog et al. [2020] are shown in Fig. 12. Additional examples are shown in the supplementary material. Our method achieves much sharper results that are significantly closer to the ground truth.

We want to note that this comparison is provided only as an indication of the efficiency of our approach, since the goals of the two methods differ in several ways.

7.2.3 Comparison to ANF. We compare with the recent Affinity of Neural Features (ANF) denoising method [Işık et al. 2021] in Figure 15. For a fair comparison we take the pretrained model provided by the authors and fine tune it in each specific scene, using the authors original implementation. Since our method uses a different renderer than ANF (Mitsuba 2 vs PBRT v3), we give the same budget in terms of pixels generated during fine tuning. Also we fine tune the pretrained ANF model using sequences of 8 frames in random paths as in the original method. Fine-tuning improves temporal stability (please see videos), and sometimes improves visual quality (e.g., sharper results for Spaceship). Finally during inference we provide an 8 samples-per-pixel (spp) input image along with the all the buffers required (albedo, depth, etc.).

Our method demonstrates better temporal stability especially in parts of the scene where the noise in the input is higher such as the reflections in the Spaceship and Veach Egg scenes. ANF manages to successfully reconstruct parts of the scene where there is a big correlation between the input buffers and the final color, such as diffuse walls in Living Room, and parts where the light effect exists in the noisy input, highlights on the floor in Living Room. The limitation of ANF is clear in cases of complex light effects that do not appear in the noisy input due to the low spp and where the input buffers do not help, such as the red caustic in Spaceship, the bottle caustic in Living Room and the complex shadow of the glass egg Veach Egg.

These effects have significant impact on the observed realism of the scene. However, they are completely missing from the path-traced+denoising solution, despite these effects being present in the ground truth images used for fine-tuning (provided in supplemental). Quantitative results are shown in Tab. 4; our method outperforms Işık et. al. [2021] in Spaceship and Living Room. For Veach Egg, two metrics give our method lower score, even though we clearly capture indirect effects that are completely missing in Işık et. al.

Scene		DSSIM	MAPE	MAE	LPIPS
Spaceship	Ours	**0.0155**	**0.047**	**0.001**	**0.0176**
	Işık et al. 2021	0.0461	0.068	0.023	0.0693
	+ Finetuned	0.0483	0.067	0.023	0.0659
Living Room	Ours	**0.0074**	**0.051**	**0.040**	**0.0287**
	Işık et al. 2021	0.0164	0.096	0.052	0.0691
	+ Finetuned	0.0205	0.109	0.050	0.0722
Veach Egg	Ours	**0.0170**	0.082	**0.021**	0.0844
	Işık et al. 2021	0.0182	**0.071**	**0.021**	0.0765
	+ Finetuned	0.0194	0.081	0.027	**0.0758**

Table 4. Quantitative results using 4 metrics for the configuration shown in Figure 15.

This illustrates one of the major strengths of our approach: the only way to render such hard light transport in a path-tracing context is to dramatically increase the number of samples per pixel. In contrast our method encodes light transport in the neural network and uses the explicit scene representation vector to get information about such effects, such as the position of the glass egg or the cockpit light. As a result, we achieve interactive rendering with all effects present for the same training time.

7.3 Evaluation

We first study the effect of the number of variable dimensions, then present other ablations concerning different design choices of our method.

Study of number of variable dimensions. We investigate the impact of the number of scene variables on our results. We trained our method and the uniform approach described above on 5 increasingly variable variants of the Salon scene. The first variant

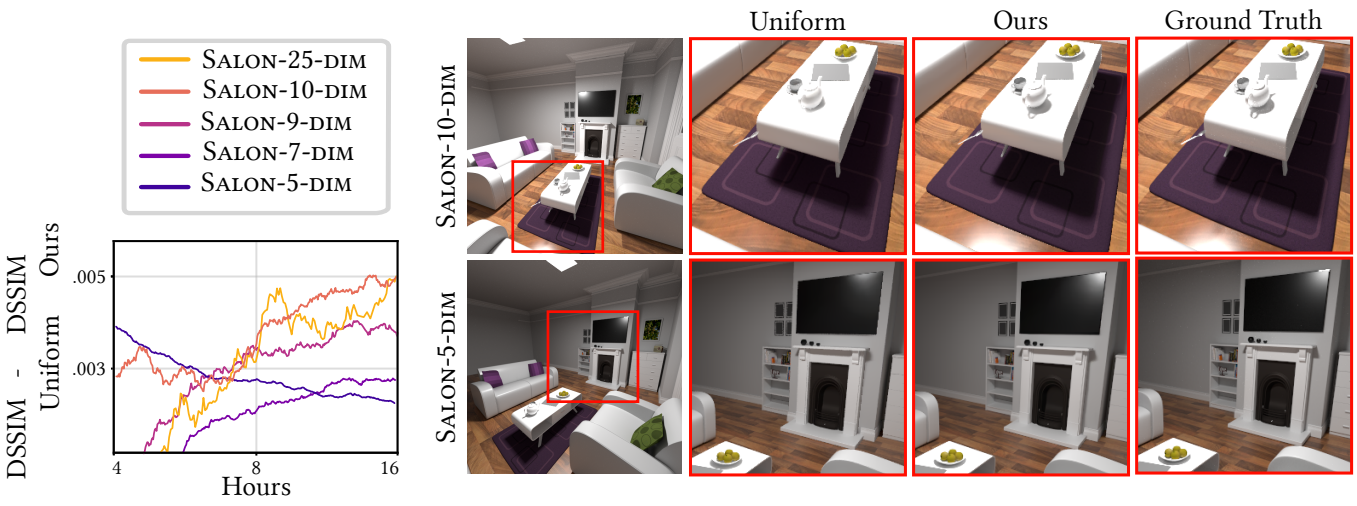

Fig. 14. Study of the number of dimensions on the effectiveness of our approach on the Salon scene. Left: we show the difference between the loss using the uniform approach and our method; the graph starts at 4h of training. In general, the benefit of our method increases with the number and complexity of the variable elements in the scene, but some elements are more important: despite going from 10 to 25 dimensions the difference of gain between Salon-10-dim and Salon-25-dim – which only involves albedo changes – is smaller than adding a single important dimension such as light position (difference from Salon-7-dim to Salon-9-dim. Right, for low dimensions (bottom row, Salon-5-dim) our method slightly improves the glossy highlight on the tv compared to uniform sampling; however, once the dimensions increase (top row, Salon-10-dim), we capture a lot of effects completely missed by the uniform, namely the tv and floor glossy highlight as well as the detailed shadows of the teapot on the table.

– Salon-5-dim – only varies viewpoint (5 dimensions), Salon-7-dim adds a movable set of furniture on the floor (7 dimensions). In Salon-9-dim the light source moves on the ceiling (9 dimensions). In Salon-10-dim the roughness of the wooden floor is also variable (10 dimensions). To demonstrate that some variables have a bigger impact on training time than others, e.g., light source position compared to changing albedo, we introduce Salon-25-dim which also varies the albedos of the furniture and walls (25 dimensions). In Fig. 14 we see that while for Salon-5-dim the difference in validation loss between our method and the uniform sampling is small, Salon-10-dim demonstrates that the benefit of our method increases with higher numbers of variable dimensions. For this case, uniform search almost completely misses the important highlight on the glossy wooden floor due to the very specific configuration of parameters that create it. As a result Active Exploration is crucial for scenes with many variable elements, such as the ones used in production.

Ablation: preconditioning on position. In Fig. 16 we show the difference in results between our full method and an ablation where position is concatenated with all other dimensions and fed directly to the network. We see clearly that the position preconditioning greatly improves overall performance. In the Living Room scene the albedo buffer for the table has high frequency variations due to the wood texture. Without the preconditioning it is hard for the PixelGenerator to learn to ignore this information when shading the caustic and the shadow of the bottle. Quantitative results in Tab. 5 confirm this choice.

Ablation: increasing resolution. We next study the effect of progressively increasing resolution during training (Sec. 6.2). In Fig. 7,

Scene		DSSIM	MAPE	MAE	LPIPS
Living Room	Ours	**0.0141**	**0.079**	**0.20**	**0.0245**
	w/o Preconditioning	0.0184	0.098	0.25	0.0393

Table 5. Preconditioning improves the quantitative performance (see also Figure 16).

we compare to an ablation where we do not increase resolution during training. We can see that the increase in resolution allows our active exploration to resolve high frequency effects such as reflections and shadows (lamp on the left) much more effectively. The corresponding quantitative results in Tab. 6 confirm the improvement in quality.

Scene		DSSIM	MAPE	MAE	LPIPS
Living Room	Ours	0.0201	0.135	0.23	0.0590
	+Mutli-res	**0.0141**	**0.079**	**0.20**	**0.0245**
	Uniform	0.0241	0.162	0.28	0.0652
	+Multi-res	0.0250	0.117	0.38	0.0573

Table 6. Quantitative results illustrating the effect of resolution on error.

Ablation: target function. In Fig. 17 and Tab. 7, we see that using only the loss for the target function degrades quality, since the training process gets stuck in local minima. The MCMC finds configurations that cannot be improved anymore, such as the mirror reflection, and does not accept other states where the network could still improve, such as the bottle caustic. As a result the latter is lacking detail.

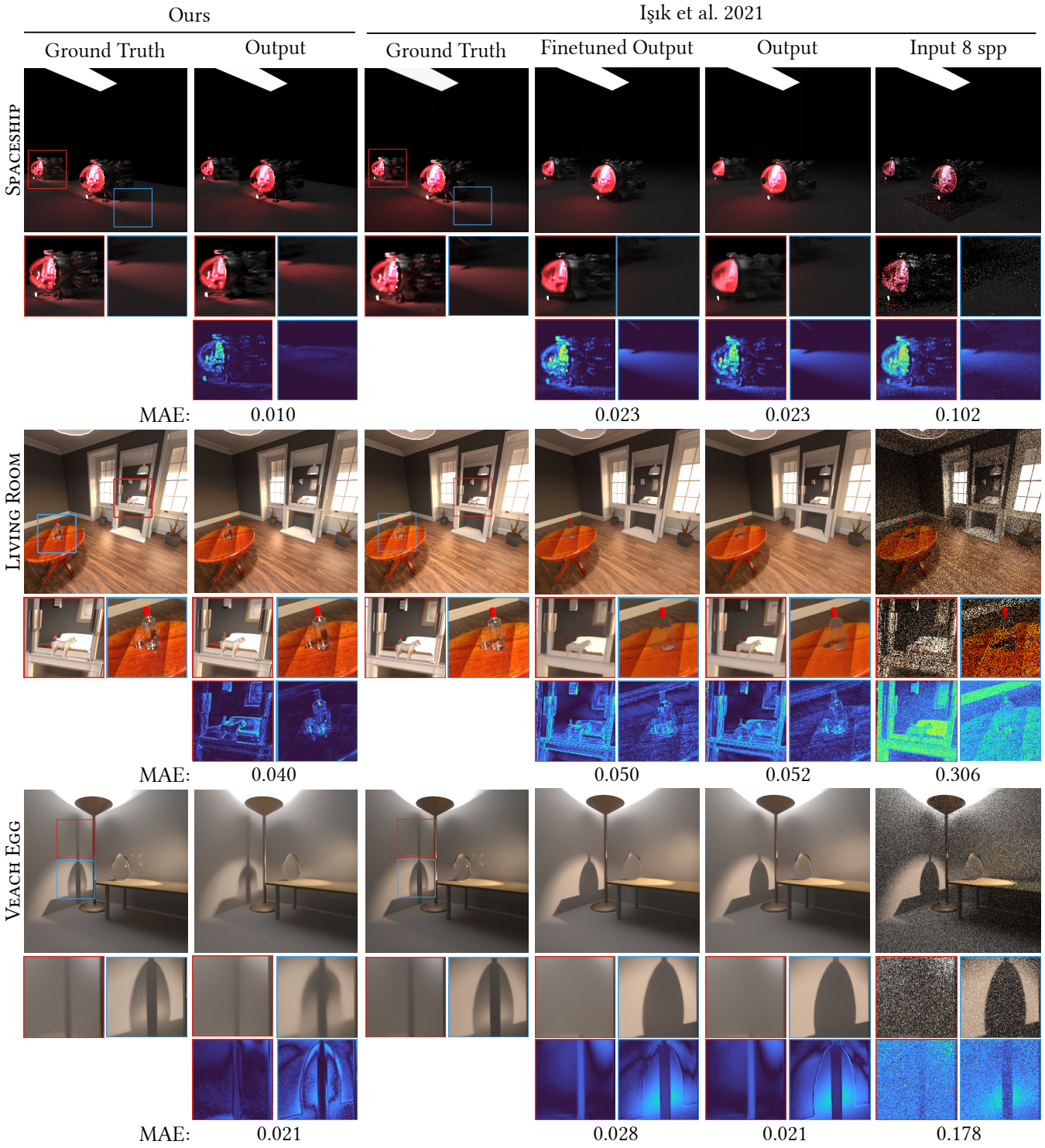

Fig. 15. Same time comparison with Işık et al. [2021], fine-tuned on our scenes. Note how our neural renderer captures hard light paths, e.g., caustics (Spaceship) or even shadows from caustics (Veach Egg) that are almost completely missing from the path-tracing + denoising solution.

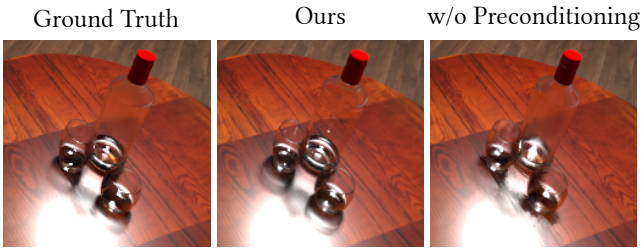

Fig. 16. Position Preconditioning allows the generator to ignore the high frequencies of the wood texture when it forms the shadows and caustic, resulting in better quality.

Scene		DSSIM	MAPE	MAE	LPIPS
Living Room	Ours	**0.0141**	**0.079**	**0.20**	**0.0245**
	Loss Based	0.0149	0.085	0.22	0.0306

Table 7. Quantitative results using 4 metrics illustration the benefit of our choice of target function (see in Figure 17).

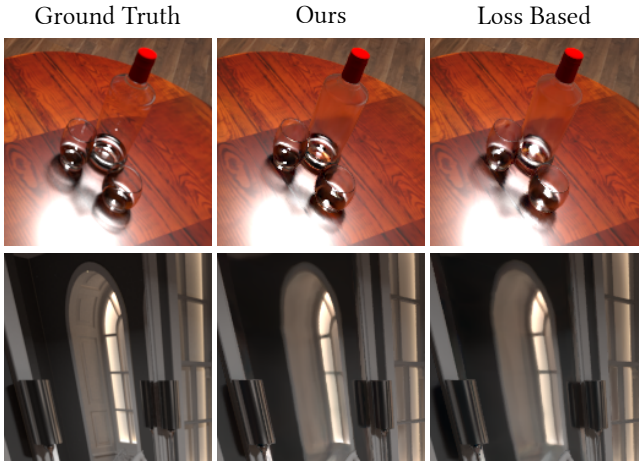

Fig. 17. Use of the loss alone for the target function results in blurrier results.

8 FUTURE WORK, LIMITATIONS AND CONCLUSION

Despite providing interactive global illumination in dynamic scenes, our method is not without limitations; we discuss these below together with some avenues for future work before concluding.

Rendering using our unoptimized Python implementation currently runs at 4-6 fps, including a 15ms overhead for generating G-buffers in Mitsuba – which could be performed with hardware acceleration – and an unoptimized inference step. We are confident that significant speedup can be achieved with further optimization. We chose to learn *all* light paths, including mirror reflections. While we achieve acceptable results in many cases, high-frequency effects may not be reproduced exactly. However, our approach can be used in a hybrid setting, using real-time ray-tracing for specular interactions as seen in Fig. 10, overcoming this issue. If the use of path tracing is not an option, Neural Textures such as the ones used

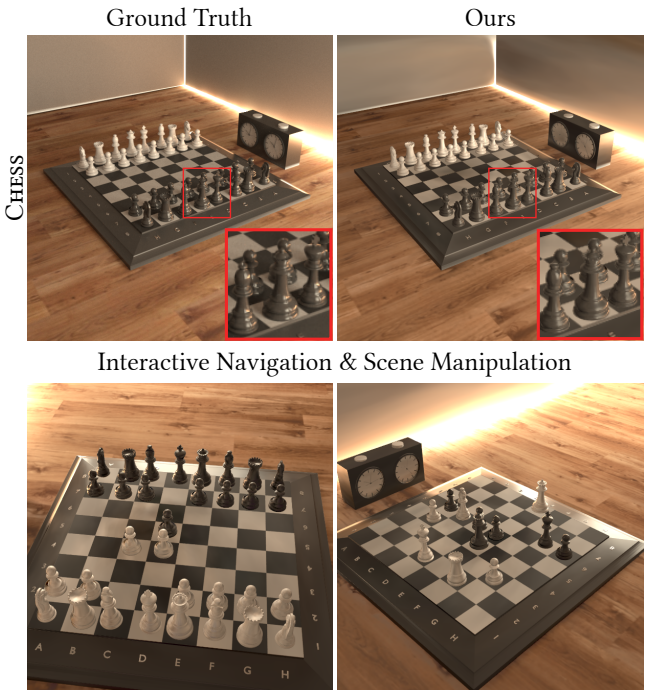

Fig. 18. The Chess scene tests the limits of our method with 128 variables. Each chess piece can be moved on the board, lifted and be captured. Our method still gives plausible results but is missing some shadows and highlights.

in [Thies et al. 2019] could improve reflections in cases where the G-Buffers do not provide any meaningful information.

Active Learning literature has explored many different metrics for deciding the value of each sample. In this work we explored two functions that can be efficiently computed in a single GPU training scenario but there are alternatives. In a multi GPU training scenario one option is to use a query by committee. Different copies of the model could be trained in parallel in each GPU and whenever a large step is performed all the models could be evaluated on the proposed state. Using the prediction variance of all the models' answers can be a good fit for a target function as it shows there is uncertainty on what the result should be. Additionally Bayesian Neural Networks with explicit access to uncertainty metrics could possibly be an option for our Active Exploration in the future.

One aspect we would like to explore in future work is how to take into account the importance/difficulty of each scene variable. From our tests different variables can have different impact on the scene's global illumination and can be harder/ easier to represent by the generator. In general, variables that create or control high frequencies, such as reflections and shadows, are much harder to learn than variables such as the color of emitters or objects. Explicit injection of this knowledge using some form of Importance Sampling could help reduce training times and improve quality. Another property of the variables that we do not handle explicitly is the difference in their ranges. Since we normalize each variable the network needs to learn to scale the normalized values accordingly to match their impact

on the final rendering. For instance for two rotatable objects that can be rotated 360° and 15° respectively the network in both cases will receive values between 0 and 1 even though the first object will create much higher frequency shadows in that range. Finding a way to adapt the MCMC mutations to such range differences per variable could increase the efficiency of Active Exploration.

The types of variables we demonstrated can have a big impact on the overall appearance of our scenes but they are simple to represent with a few floats (rotation, translation, roughness etc). In future work we would like to expand our method to variables that are difficult to represent such as the parametric deformations in [Sloan et al. 2005]. We believe that finding inventive ways to represent such variability (such as using the keyframe as a parameter) is a promising avenue of future research.

We still can require up to 18 hours of training time for a given scene, depending on the required quality and the number of variable parameters. As discussed earlier, the network architecture used can play a significant role in the quality of the results; it is possible that different architectures will further improve quality and thus training speed. Another possible extension could be to train with a set of variable parameters and allow fine-tuning of the network, e.g., allowing fast addition of a new object etc. Evidently, use of a faster path-tracer could also accelerate training.

In future work, we believe our Active Exploration approach has significant promise for any neural rendering method (e.g., [Baatz et al. 2021]) that trains on synthetic data, allowing potentially significant reduction in training time and improvements in quality.

One limitation by design for our method is that we cannot handle thousands of variables. The scene representation vector is repeated to match the size of the G-Buffers so that the generator, which operates on a per pixel basis, is aware of the global state of the scene. For example, given 5000 variables (such as a variable texture) we would need to create a tensor of size 128x128x5000 that would be unmanageable in terms of memory. In such cases there is a need to encode this information in a different way, possibly through an encoder neural network. Our method, as shown in Figure 18, can work with 128 variables with similar training times (18 hours) but the quality is lower than in simpler scenes (some missing highlights and shadows).

In conclusion, we introduced a resolution-aware Active Exploration method that guides the sampling of the training data space, and a self-tuning sample reuse method that enables interleaved on-the-fly data generation and training.

Our neural renderer, combined with our explicit scene instance parameterization vector, uses these contributions to capture hard light transport effects, allowing interactive exploration with full global illumination, including all light paths.

Using these elements we can render variable scenes after 5-18 hours of training, depending on scene and variation complexity and the quality required, including indirect lighting, shadows, transmission, glossy effects etc. Looking forward, we believe that our main contributions can be used beyond the precomputation scenario presented here: Active Exploration and self-tuning reuse could be used for future solutions that can provide data online, e.g., with real-time path tracing.

ACKNOWLEDGMENTS

This research was funded by ERC Advanced grant FUNGRAPH No 788065 (http://fungraph.inria.fr). The authors are grateful to the OPAL infrastructure from Université Côte d'Azur for providing resources and support. The authors would also like to thank the anonymous referees for their valuable comments and helpful suggestions, as well as Georgios Kopanas and Felix Hähnlein for fruitful discussions.

REFERENCES

2019. Offline Deep Importance Sampling for Monte Carlo Path Tracing. *Computer Graphics Forum (Proceedings of Pacific Graphics 2019)* 38, 7 (2019), 527–542.

Hendrik Baatz, Jonathan Granskog, Marios Papas, Fabrice Rousselle, and Jan Novák. 2021. NeRF-Tex: Neural Reflectance Field Textures. In *Eurographics Symposium on Rendering (DL-only track)*. The Eurographics Association.

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

Mojtaba Bemana, Karol Myszkowski, Hans-Peter Seidel, and Tobias Ritschel. 2020. X-Fields: Implicit Neural View-, Light- and Time-Image Interpolation. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia 2020)* 39, 6 (2020). https://doi.org/10.1145/3414685.3417827

Nir Benty, Kai-Hwa Yao, Petrik Clarberg, Lucy Chen, Simon Kallweit, Tim Foley, Matthew Oakes, Conor Lavelle, and Chris Wyman. 2020. The Falcor rendering framework, 03 2020.

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

Benedikt Bitterli and Wojciech Jarosz. 2019. Selectively metropolised Monte Carlo light transport simulation. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–10.

Benedikt Bitterli, Chris Wyman, Matt Pharr, Peter Shirley, Aaron Lefohn, and Wojciech Jarosz. 2020. Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 148–1.

Guillaume Bouchard, Théo Trouillon, Julien Perez, and Adrien Gaidon. 2015. Online learning to sample. *arXiv preprint arXiv:1506.09016* (2015).

J. Burgess. 2020. RTX on the NVIDIA Turing GPU. *IEEE Micro* 40, 2 (2020), 36–44.

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

SM Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S Morcos, Marta Garnelo, Avraham Ruderman, Andrei A Rusu, Ivo Danihelka, Karol Gregor, et al. 2018. Neural scene representation and rendering. *Science* 360, 6394 (2018), 1204–1210.

Michaël Gharbi, Tzu-Mao Li, Miika Aittala, Jaakko Lehtinen, and Frédo Durand. 2019. Sample-based Monte Carlo denoising using a kernel-splatting network. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–12.

Jonathan Granskog, Fabrice Rousselle, Marios Papas, and Jan Novák. 2020. Compositional neural scene representations for shading inference. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 135–1.

Gene Greger, Peter Shirley, Philip M Hubbard, and Donald P Greenberg. 1998. The irradiance volume. *IEEE Computer Graphics and Applications* 18, 2 (1998), 32–43.

Saeed Hadadan, Shuhong Chen, and Matthias Zwicker. 2021. Neural Radiosity. *arXiv preprint arXiv:2105.12319* (2021).

Elad Hazan, Tomer Koren, and Nati Srebro. 2011. Beating sgd: Learning svms in sublinear time. In *Advances in Neural Information Processing Systems*. 1233–1241.

Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. 2018. Deep Blending for Free-Viewpoint Image-Based Rendering. *ACM Transactions on Graphics (SIGGRAPH Asia Conference Proceedings)* 37, 6 (November 2018). http://www-sop.inria.fr/reves/Basilic/2018/HPPFDB18

Mustafa Işık, Krishna Mullia, Matthew Fisher, Jonathan Eisenmann, and Michaël Gharbi. 2021. Interactive Monte Carlo Denoising using Affinity of Neural Features. *ACM Transactions on Graphics (TOG)* 40, 4, Article 37 (2021), 13 pages. https://doi.org/10.1145/3450626.3459793

Csaba Kelemen, László Szirmay-Kalos, György Antal, and Ferenc Csonka. 2002. A simple and robust mutation strategy for the metropolis light transport algorithm. In *Computer Graphics Forum*, Vol. 21. Wiley Online Library, 531–540.

Alexander Keller, Luca Fascione, Marcos Fajardo, Iliyan Georgiev, Per Christensen, Johannes Hanika, Christian Eisenacher, and Gregory Nichols. 2015. The path tracing revolution in the movie industry. In *ACM SIGGRAPH 2015 Courses*. 1–7.

Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. 2018. Measuring catastrophic forgetting in neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

Hyeongwoo Kim, Michael Zollhöfer, Ayush Tewari, Justus Thies, Christian Richardt, and Christian Theobalt. 2018. Inversefacenet: Deep monocular inverse face rendering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4625–4634.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference for Learning Representations*.

Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila. 2018. Noise2noise: Learning image restoration without clean data. *arXiv preprint arXiv:1803.04189* (2018).

Chunyuan Li, Andrew Stevens, Changyou Chen, Yunchen Pu, Zhe Gan, and Lawrence Carin. 2016. Learning weight uncertainty with stochastic gradient mcmc for shape classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5666–5675.

Xiao Li, Yue Dong, Pieter Peers, and Xin Tong. 2017. Modeling surface appearance from a single photograph using self-augmented convolutional neural networks. *ACM Transactions on Graphics (ToG)* 36, 4 (2017), 1–11.

A. Loza, L. Mihaylova, N. Canagarajah, and D. Bull. 2006. Structural Similarity-Based Object Tracking in Video Sequences. In *2006 9th International Conference on Information Fusion*. 1–6. https://doi.org/10.1109/ICIF.2006.301574

Zander Majercik, Jean-Philippe Guertin, Derek Nowrouzezahrai, and Morgan McGuire. 2019. Dynamic diffuse global illumination with ray-traced irradiance fields. *Journal of Computer Graphics Techniques Vol* 8, 2 (2019).

Morgan McGuire, Mike Mara, Derek Nowrouzezahrai, and David Luebke. 2017. Real-time global illumination using precomputed light field probes. In *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. 1–11.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*.

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

Thomas Müller, Fabrice Rousselle, Jan Novák, and Alexander Keller. 2021. Real-time neural radiance caching for path tracing. *arXiv preprint arXiv:2106.12372* (2021).

O. Nalbach, E. Arabadzhiyska, D. Mehta, Seidel. H-P., and T. Ritschel. 2017. Deep Shading: Convolutional Neural Networks for Screen Space Shading. *Computer Graphics Forum (Proc. EGSR)* 36, 4 (2017).

Radford M Neal. 1996. *Bayesian learning for neural networks*. Springer Verlag.

Greg Nichols, Jeremy Shopf, and Chris Wyman. 2009. Hierarchical image-space radiosity for interactive global illumination. In *Computer Graphics Forum*, Vol. 28. Wiley Online Library, 1141–1149.

Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. 2019. Mitsuba 2: A retargetable forward and inverse renderer. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–17.

Julien Philip, Michaël Gharbi, Tinghui Zhou, Alexei Efros, and George Drettakis. 2019. Multi-view Relighting Using a Geometry-Aware Network. *ACM Transactions on Graphics (SIGGRAPH Conference Proceedings)* 38, 4 (July 2019). http://www-sop.inria.fr/reves/Basilic/2019/PGZED19

Peiran Ren, Jiaping Wang, Minmin Gong, Stephen Lin, Xin Tong, and Baining Guo. 2013. Global Illumination with Radiance Regression Functions. *ACM Trans. Graph.* 32, 4, Article 130 (July 2013), 12 pages.

Gernot Riegler and Vladlen Koltun. 2020. Free view synthesis. In *European Conference on Computer Vision*. Springer, 623–640.

Tobias Ritschel, Carsten Dachsbacher, Thorsten Grosch, and Jan Kautz. 2012. The state of the art in interactive global illumination. In *Computer Graphics Forum*, Vol. 31. Wiley Online Library, 160–188.

Tobias Ritschel, Thorsten Grosch, and Hans-Peter Seidel. 2009. Approximating dynamic global illumination in image space. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*. 75–82.

Simon Rodriguez, Thomas Leimkühler, Siddhant Prakash, Chris Wyman, Peter Shirley, and George Drettakis. 2020. Glossy Probe Reprojection for Interactive Global Illumination. *ACM Transactions on Graphics (SIGGRAPH Asia Conference Proceedings)* 39, 6 (December 2020). http://www-sop.inria.fr/reves/Basilic/2020/RLPWSD20

Ozan Sener and Silvio Savarese. 2017. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489* (2017).

Burr Settles. 2009. *Active learning literature survey*. Technical Report. University of Wisconsin-Madison Department of Computer Sciences.

Dario Seyb, Peter-Pike Sloan, Ari Silvennoinen, Michał Iwanicki, and Wojciech Jarosz. 2020. The design and evolution of the UberBake light baking system. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 150–1.

Ari Silvennoinen and Jaakko Lehtinen. 2017. Real-time global illumination by precomputed local reconstruction from sparse radiance probes. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 1–13.

Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. 2020. Implicit Neural Representations with Periodic Activation Functions. In *Proc. NeurIPS*.

Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. 2019. Scene Representation Networks: Continuous 3D-Structure-Aware Neural Scene Representations. In *Advances in Neural Information Processing Systems*.

Peter-Pike Sloan, Ben Luna, and John Snyder. 2005. Local, deformable precomputed radiance transfer. *ACM Transactions on Graphics (TOG)* 24, 3 (2005), 1216–1224.

Pratul P Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T Barron. 2021. NeRV: Neural Reflectance and Visibility Fields for Relighting and View Synthesis. *CVPR*.

Pratul P Srinivasan, Richard Tucker, Jonathan T Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. 2019. Pushing the boundaries of view extrapolation with multiplane images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 175–184.

Sebastian U Stich, Anant Raj, and Martin Jaggi. 2017. Safe adaptive importance sampling. In *Advances in Neural Information Processing Systems*. 4381–4391.

Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. 2020. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. *NeurIPS* (2020).

A. Tewari, O. Fried, J. Thies, V. Sitzmann, S. Lombardi, K. Sunkavalli, R. Martin-Brualla, T. Simon, J. Saragih, M. Nießner, R. Pandey, S. Fanello, G. Wetzstein, J.-Y. Zhu, C. Theobalt, M. Agrawala, E. Shechtman, D. B Goldman, and M. Zollhöfer. 2020. State of the Art on Neural Rendering. *Computer Graphics Forum (EG STAR 2020)* (2020).

Justus Thies, Michael Zollhöfer, and Matthias Nießner. 2019. Deferred neural rendering: Image synthesis using neural textures. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–12.

T. Tieleman and G. Hinton. 2012. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning.

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

Max Welling and Yee W Teh. 2011. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*. Citeseer, 681–688.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 586–595.

Ruqi Zhang, Chunyuan Li, Jianyi Zhang, Changyou Chen, and Andrew Gordon Wilson. 2019. Cyclical stochastic gradient MCMC for bayesian deep learning. *arXiv preprint arXiv:1902.03932* (2019).

Peilin Zhao and Tong Zhang. 2015. Stochastic optimization with importance sampling for regularized loss minimization. In *international conference on machine learning*. 1–9.

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

Junqiu Zhu, Yaoyi Bai, Zilin Xu, Steve Bako, Edgar Velázquez-Armendáriz, Lu Wang, Pradeep Sen, Miloš Hašan, and Ling-Qi Yan. 2021. Neural Complex Luminaires: Representation and Rendering. *ACM Trans. Graph.* 40, 4, Article 57 (July 2021), 12 pages. https://doi.org/10.1145/3450626.3459798

Jia-Jie Zhu and José Bento. 2017. Generative adversarial active learning. *arXiv preprint arXiv:1702.07956* (2017).