Learning to Importance Sample in Primary Sample Space

Quan Zheng [iD] and Matthias Zwicker [iD]

University of Maryland, College Park, USA
quan.zheng@outlook.com; zwicker@cs.umd.edu

Abstract

Importance sampling is one of the most widely used variance reduction strategies in Monte Carlo rendering. We propose a novel importance sampling technique that uses a neural network to learn how to sample from a desired density represented by a set of samples. Our approach considers an existing Monte Carlo rendering algorithm as a black box. During a scene-dependent training phase, we learn to generate samples with a desired density in the primary sample space of the renderer using maximum likelihood estimation. We leverage a recent neural network architecture that was designed to represent real-valued non-volume preserving ("Real NVP") transformations in high dimensional spaces. We use Real NVP to non-linearly warp primary sample space and obtain desired densities. In addition, Real NVP efficiently computes the determinant of the Jacobian of the warp, which is required to implement the change of integration variables implied by the warp. A main advantage of our approach is that it is agnostic of underlying light transport effects, and can be combined with an existing rendering technique by treating it as a black box. We show that our approach leads to effective variance reduction in several practical scenarios.

CCS Concepts

• *Computing methodologies* → *Ray tracing; Neural networks; Importance sampling; Global illumination;*

1. Introduction

Importance sampling has been recognized as a key technique for variance reduction right from the inception of Monte Carlo rendering algorithms [Kaj86]. Today, importance sampling of BRDFs, environment maps, direct illumination from many light sources, or visibility are standard features in Monte Carlo path tracing systems. A number of advanced techniques have also been developed to jointly importance sample several of these factors. Many of these approaches rely on an analytical analysis of scene properties, such as the surface appearance models and BRDFs used in the scenes.

In this paper, we propose a technique that treats an existing Monte Carlo renderer as a black box and learns how to importance sample entire paths in primary sample space. Our approach first acquires a set of training samples for a given scene using the existing renderer. Based on these samples, we learn to generate a desired scene-dependent target density in the primary sample space (PSS) of the renderer. In the subsequent rendering step, instead of feeding the renderer with uniform PSS samples, we provide samples drawn from the learned target density. By specifying a suitable target density, we achieve effective variance reduction compared to using the existing renderer with the usual uniform primary sample space.

The key component of our approach is a recent neural network architecture that was designed to represent real-valued non-volume preserving ("Real NVP") transformations in high dimensional spaces. This approach learns a one-to-one, non-linear warp between two high-dimensional spaces. In addition, the computa-

tion of the warp is structured such that the forward warp, its inverse, and the determinant of its Jacobian can all be computed effectively. We leverage these properties to learn a warp from a uniform density to a desired non-uniform target density in primary sample space, and then generate well-distributed PSS samples for Monte Carlo rendering tasks. Accounting for efficiency issues with high-dimensional spaces, we adopt practical simplifications to make learning and sampling the target density tractable in limited dimensions. The advantages of our approach are that it treats a renderer as a black box and is agnostic to specific light transport effects, hence it can be combined with many existing algorithms.

Conceptually, our approach has similarities to several previous strategies. Primary sample space Metropolis (PSS-MLT) light transport can also importance sample any desired target density by operating in primary sample space while treating an existing renderer as a black box [KSKAC02]. PSS-MLT can be inefficient, however, because it often needs to reject many proposed paths to achieve the desired density. In contrast, our approach never rejects samples during rendering. Several previous techniques acquire a set of initial samples to build up low-dimensional ad hoc data structures that approximate the desired density, and then can be used to perform importance sampling during subsequent rendering passes. The advantage of our approach is that the renderer is treated as a black box, and the approach does not rely on additional data structures such as octrees, spatial samplings of the scene, or explicit density models. Instead, the information learned from an initial set of

samples is captured by the Real NVP neural network. In summary, the main contribtions of this paper are:

- A novel formulation of importance sampling of entire light paths as a non-linear warp in primary sample space.
- A novel technique to learn the primary sample space warp using a suitable neural network architecture.
- A demonstration that this approach can effectively reduce variance of Monte Carlo path tracers in several scenarios.

2. Related Work

2.1. Importance Sampling in Monte Carlo Rendering

Already when introducing the rendering equation, Kajiya [Kaj86] discussed importance sampling as a technique to reduce variance in Monte Carlo rendering. Importance sampling aims to obtain samples with a probability density that is proportional to a desired target density function, and by designing target densities similar to the integrand, variance can be reduced. In Monte Carlo path tracing, the target density function can be defined either as a product of local densities in an incremental manner, or directly as a density in global path space. Accordingly, two categories of importance sampling techniques have been studied in past decades.

Incremental sampling. When light paths are constructed in an incremental way, it is natural to perform importance sampling locally in each step. For example, BRDF sampling and light sampling techniques construct a path segment by separately mimicking the local distribution of cosine weighted BRDF and incident illumination, respectively. To further reduce the estimator's variance, Veach and Guibas [VG95] introduced multiple importance sampling to combine the advantages of individual path sampling techniques. We refer to standard texts for an overview of the extensive literature [PJH16]. In general, importance sampling techniques can be categorized into two broad groups: *a priori* methods that construct target densities and sampling techniques using analytical approximations of the integrand (for example, Heitz and d'Eon [Hd14] among many others), and *a posteriori* methods that fit target densities based on empirically acquired samples of the integrand [VKŠ*14, MGN17, DK17]. The technique by Clarberg [CJAMJ05] is an *a priori* method that performs importance sampling of products of environment maps, BRDFs, and visibility by representing them using wavelets. They achieve importance sampling by evaluating wavelet products on the fly and constructing a hierarchical warp accordingly, which has some similarities to our warp-based importance sampling. Our approach, however, operates in an *a posteriori* manner and it does not rely on a specific representation of the integrand, such as using wavelet products. Our approach is also more general and can be applied to any light transport effect. Recently, a Bayesian method has been proposed to sample direct illumination that can be considered a combination of *a priori* and *a posteriori* strategies [VKK18]. Our approach considers entire light paths instead of performing incremental sampling.

Global sampling. Instead of importance sampling individual path segments, Metropolis light transport [VG97] treats a complete path as a single sample in a global path space that is also used to define the target density. A new path is sampled from a Markov

process by mutating an existing path according to a scalar contribution function. Various parameterizations of path space have been proposed to efficiently explore it using Markov chain methods [KSKAC02, JM12, KHD14, LLR*15], and techniques have been proposed to combine different parameterizations to further improve the sampling process [HKD14,OKH*17,BJNJ17]. Our approach is similar to global path sampling techniques since we also importance sample entire paths. In particular, we sample paths in primary sample space as proposed by Kelemen et al. [KSKAC02], which is simply a multi-dimensional unit hypercube. As in Kelemen's approach, we also build on existing path construction techniques that map primary sample space parameters into geometric paths, and that we treat as a black box. In contrast, our approach does not rely on Markov chain sampling, however. Instead, we obtain a non-linear, one-to-one mapping of primary sample space onto itself that produces the desired target density.

2.2. Deep Learning to Sample Complex Data Distributions

Recently, various neural network architectures have been proposed to learn generative models of high-dimensional data, such as images and videos. A generative model transforms random samples from a latent space into samples of some observable data, for example images, such that the distribution of the generated data matches the distribution of the observed data. Successful techniques include generative adversarial networks (GANs) [GPAM*14] and variational autoencoders (VAEs) [KW14]. Our idea is to use such a learned mapping from a latent space to a data space to perform importance sampling of light paths. A key requirement for this is that we need to be able to efficiently compute the Jacobian of this mapping, such that we can use it to perform a change of integration variables. However, the computation of the Jacobian involved in deep neural network is costly in techniques such as GANs or VAEs. Instead, we leverage a recent architecture called "Real NVP" [DSB16] that was specifically designed as an invertible mapping whose Jacobian can be computed easily. Concurrent work [MMR*18] also leverages "Real NVP" to construct an invertible mapping for use in Monte Carlo rendering. While they propose a slightly different representation of such a bijective mapping, in essence the concept is equivalent to ours.

2.3. Deep Learning in Monte Carlo Rendering

Deep learning has been successfully applied to denoise Monte Carlo rendering [BVM*17, CKS*17], which is common in production today. Our approach is orthogonal to these techniques and can be combined with any denoiser. Neural networks have also been used as a regression architecture to interpolate radiance values from sparse samples [RWG*13, KMM*17]. These methods have no guarantees to converge to ground truth solutions, however, since they directly predict radiance based on a model learned from a limited amount of data. In contrast, we learn how to sample, and our sampling densities enable unbiased importance sampling. While we do not have a theoretical proof that our approach reduces variance, we demonstrate significant variance reduction in practical scenarios.

3. Background

The path integral formulation [VG97] of the rendering equation [Kaj86] expresses the value I_j of each pixel j as an integral over the space of all light paths,

$$I_j = \int_\Omega f_j(x) \mathrm{d}\mu(x), \qquad (1)$$

where x is a light path, $\mathrm{d}\mu$ is a measure on the space of paths, and f_j is the measurement contribution function. In addition, Ω is the space of all light paths, consisting of the union of light paths of all lengths k, $\Omega = \cup_k \Omega_k$.

An important observation is that there are many ways to parameterize light paths, and each parameterization has its own measure $\mathrm{d}\mu$. Considering only light paths of a certain length k, however, it is always possible to define a mapping Φ_k from a canonical parameterization over the $2(k+1)$-dimensional unit hypercube[†] to Ω_k, that is, $\Phi_k : [0,1]^{2(k+1)} \to \Omega_k$. This is also called primary sample space [KSKAC02], and indeed each Monte Carlo rendering algorithm implicitly evaluates Φ_k when constructing light paths. Using Φ_k to perform a change of integration variables, we can rewrite the path integral formulation as

$$I_{j,k} = \int_{\Omega_k} f_j(x) \mathrm{d}\mu(x) = \int_{[0,1]^{2(k+1)}} f_j(\Phi_k(y)) \left| \frac{\partial \Phi_k(y)}{\partial y} \right| \mathrm{d}y, \qquad (2)$$

where $I_{j,k}$ denotes the contribution of all paths of length k to pixel j, $y \in [0,1]^{2(k+1)}$, and the notation $|\cdot|$ is shorthand for the determinant of the Jacobian matrix. Figure 1(a) illustrates the notation in Equation 2.

In this view, incremental importance sampling techniques discussed in Section 2.1 imply certain mappings Φ_k from primary sample space to geometric paths, and the effect of importance sampling is absorbed in the mapping Φ_k. Usually, the integral in Equation 2 is estimated using uniform sampling in primary sample space,

$$I_{j,k} \approx \frac{1}{N} \sum_{i=1}^{N} \frac{f_j(\Phi_k(y_i))}{\left| \frac{\partial \Phi_k(y_i)}{\partial y_i} \right|^{-1}}, \qquad (3)$$

where the $y_i \in [0,1]^{2(k+1)}$ are uniform random samples. The goal of importance sampling is to construct suitable mappings Φ_k with determinants that are inversely proportional to f_j as much as possible[‡], which will reduce variance of the estimate in Equation 3. Incremental importance sampling approaches, however, rely only on local information about f_j, hence they are unable to account for non-local effects.

[†] This assumes that path vertices are on surfaces, and each of them is parameterized using two numbers. Depending on the details of the renderer, however, more parameters may be involved in practice, for example to select BRDF components.

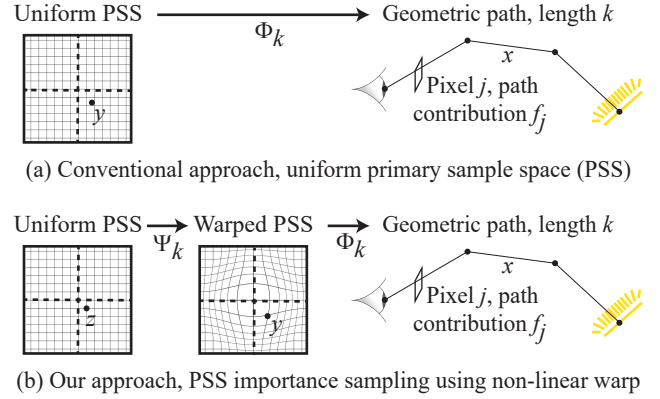[‡] The determinant of the Jacobian is the inverse of the corresponding sampling density.

(a) Conventional approach, uniform primary sample space (PSS)

(b) Our approach, PSS importance sampling using non-linear warp

Figure 1: *Comparison of the conventional approach and ours: (a) Usually, primary sample space (PSS) is sampled uniformly. Each point y in primary sample space (PSS) of dimension $2(k+1)$ corresponds to a geometric path x of length k via a mapping Φ_k. Conventional importance sampling is accounted for by the determinant of the Jacobian of Φ_k. (b) Our approach introduces a non-linear warp Ψ_k in primary sample space to further reduce variance, and we learn this mapping using a neural network.*

4. Primary Sample Space (PSS) Warping

As illustrated in Figure 1(b), the key idea of our approach is to introduce an additional mapping of primary sample space onto itself, acting as a non-uniform warp that leads to a non-uniform PSS density. We design this density to further reduce the variance of an existing renderer that evaluates Equation 3, which we treat as a black box. We learn the warp in a scene-dependent training phase, and then use it to draw as many samples for rendering as desired.

In the following, we describe how we construct the warp in the training phase. We start by introducing the details of our problem statement in Section 4.1, and explain in Section 4.2 how we achieve the objective of the PSS warp using an *a posteriori* approach. This involves first drawing a number of samples from Equation 3 and resampling them to obtain the desired target density. Then we use a maximum likelihood estimation of the warp to match the target density. In Section 4.3 we describe how we represent the warp in practice using the Real NVP architecture and solve the maximum likelihood estimation via gradient descent. We describe the details of the neural networks involved in Real NVP in Section 4.4, and describe how we use the warp during rendering in Section 4.5.

4.1. Problem Formulation

Let us denote the PSS warp that we will learn as

$$\Psi_k : [0,1]^{2(k+1)} \to [0,1]^{2(k+1)}. \qquad (4)$$

Assuming this is bijective and differentiable, we can use it to introduce another change of integration variables $y = \Psi_k(z)$ in Equation 2,

$$I_{j,k} = \int_{[0,1]^{2(k+1)}} f_j(\Phi_k(\Psi_k(z))) \left| \frac{\partial \Psi_k(z)}{\partial z} \right| \left| \frac{\partial \Phi_k(\Psi_k(z))}{\partial \Psi_k(z)} \right| \mathrm{d}z. \qquad (5)$$
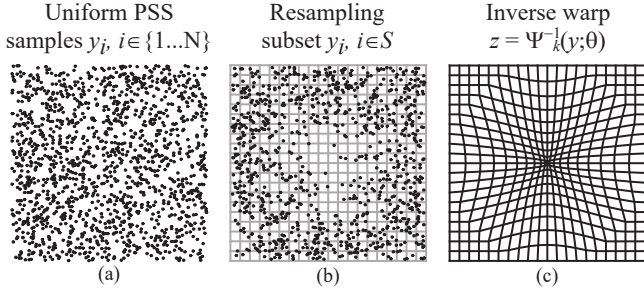
Figure 2: *We illustrate how we learn the inverse warp $\Psi_k^{-1}(y;\theta)$ to match a target density. (a) We first obtain uniform samples in PSS. (b) We resample them to obtain samples from the target density. (c) The inverse warp $\Psi_k^{-1}(y;\theta)$ is computed to maximize the likelihood of the target samples under the warp.*

This leads to the Monte Carlo estimator

$$I_{j,k} \approx \frac{1}{N} \sum_{i=1}^{N} \frac{f_j(\Phi_k(\Psi_k(z_i)))}{\left| \frac{\partial \Psi_k(z_i)}{\partial z_i} \right|^{-1} \left| \frac{\partial \Phi_k(\Psi_k(z_i))}{\partial \Psi_k(z_i)} \right|^{-1}}, \qquad (6)$$

where the $z_i \in [0,1]^{2(k+1)}$ are uniform random samples. To reduce variance of this, we want to design Ψ_k such that the integrand in Equation 5 is as close as possible to a constant. Note that the determinant of the Jacobian of our primary sample space warp, $|\partial \Psi_k(z)/\partial z|$ is the inverse of the desired non-uniform sampling density in a warped primary sample space, which serves as one part of the inputs to the black-box renderer.

Practical Simplifications. Our description so far implies that we would need to learn a mapping Ψ_k for each pixel. This is impractical, however, and we will firstly make the simplification to replace the measurement contribution function f_j with the path throughput f, which is related to the measurement contribution by omitting the pixel filter (also called the importance function). While the path length k is unlimited in theory, in practice, only paths with limited length are processed. Additionally, longer paths generally contribute less to the final image than shorter paths, thus the actual advantages of importance sampling are reduced after the first few bounces. Hence, our second simplification is to only learn a PSS warp for certain limited path lengths m, and go on tracing paths with uniform random PSS coordinates for later bounces.

4.2. Maximum Likelihood Estimation of the PSS Warp

As illustrated in Figure 2, we follow an *a posteriori* approach to learn the PSS warp by first drawing a set of samples from a desired target density in primary sample space. Then we estimate the warp under a maximum likelihood objective for these observed samples.

Sampling the Target Density. As mentioned above, in this paper we use the path throughput f to define our target density as

$$p_Y(y) = \frac{f(\Phi_k(y))}{\left| \frac{\partial \Phi_k(y)}{\partial y} \right|^{-1}}, \qquad (7)$$

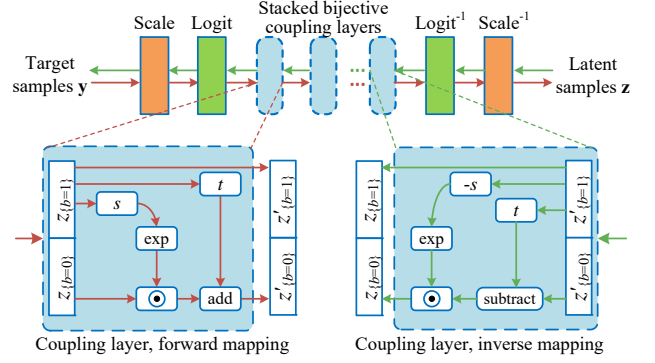although any other target density could be used. We obtain a set

Figure 3: *Computational structure of our neural importance sampling model based on Real NVP. The core element of this architecture consists of a set of stacked, invertible coupling layers. The functions s and t are implemented using neural networks as shown in Figure 5, and stacking several coupling layers makes it possible to represent complex, bijective mappings.*

of samples $\mathbb{S} = \{y_i \mid i \in N^+, y_i \backsim p_Y\}$ from this distribution using a resampling process as proposed by Talbot et al. [TCE05]. This involves first rendering a set $\mathbb{T}$ of candidate samples using a uniform PSS density as usual, and storing the PSS parameters and path throughputs for all samples. Then the subset $\mathbb{S}$ is extracted from $\mathbb{T}$ such that $\mathbb{S}$ follows the desired target density. The ratio of the sizes of $\mathbb{S}$ and $\mathbb{T}$ is determined by a parameter α, that is, $|\mathbb{T}| = \alpha|\mathbb{S}|$.

Maximum Likelihood Estimation. We learn the inverse warp $z = \Psi_k^{-1}(y)$ using maximum likelihood estimation. Assume the warp is parameterized using parameters θ, written as $z = \Psi_k^{-1}(y;\theta)$, $\theta \in \Theta$. Here, Θ represents the parameter space of a family of acceptable distributions. We first observe that the density $p_Y(\theta;y)$ and the warp Ψ_k^{-1} are related by the change of variable formula

$$p_Y(\theta;y) = \left| \frac{\partial \Psi_k^{-1}(y;\theta)}{\partial y} \right|. \qquad (8)$$

Note that this implies that latent variable $z = \Psi_k^{-1}(y)$ is distributed uniformly. Considering $p_Y(\theta;y)$ a function of θ, it is also called a likelihood function. The maximum likelihood estimation approach tries to find the optimum parameters $\theta \in \Theta$ to maximize the likelihood, that the warp produces the data samples $y_i \backsim p_Y$ from uniform samples z_i. More precisely, maximizing the log-likelihood of the data samples $y_i, i \in \{1, \dots, |\mathbb{S}|\}$ can be written as

$$\theta^* = \arg\max_{\theta \in \Theta} \sum_i \log \left(\left| \frac{\partial \Psi^{-1}(y_i;\theta)}{\partial y_i} \right| \right). \qquad (9)$$

4.3. PSS Warping using Real NVP

We are leveraging a recent neural network architecture dubbed as "Real NVP" transformation [DSB16] to represent our PSS warp Ψ and its inverse Ψ^{-1}. Real NVP provides desired properties that fit into our problem: it is guaranteed to be bijective, it is easy to invert, the determinant of its Jacobian can be evaluated efficiently,
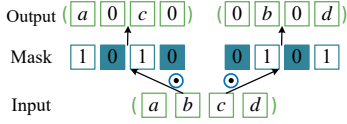
Figure 4: *Element-wise masking of a 4D input vector. Masks are designed as a binary "checkerboard" mode, where elements of the mask are set to alternating 0 and 1.*

Figure 5: *The neural network architecture to compute the s and t functions in a bijective coupling layer consists of N residual blocks (RB), containing fully connected layers (FC), batch normalization layers (BN), and ReLU activation units. Additionally, we have fully connected, batch normalization, and activation layers before and after the residual blocks. All fully connected layers contain the same number of neurons.*

and maximum likelihood estimation can be performed via gradient descent.

In a nutshell, Real NVP transformations consist of multiple stacked (concatenated) so-called affine coupling layers as shown in Figure 3, top. The mapping computed by each coupling layer is designed to have all the properties mentioned above, and the concatenation of multiple coupling layers can represent complex mappings. Assume each coupling layer computes a mapping from a *D*-dimensional space onto itself. A key idea is to split the input vector *z* into two disjoint parts, which we represent by a binary mask *b* of size *D*. An example of applying element-wise masking to a 4D vector is shown in Figure 4. Let $\{b = 1\}$ be the set of indices where the mask has value 1, and similar for $\{b = 0\}$. Each coupling layer forwards the first part of the input $z_{\{b=1\}}$ directly to its output. In addition, the second part of the output $z_{\{b=0\}}$ consists of an affine mapping that is constructed using functions $s(z_{\{b=1\}})$ and $t(z_{\{b=1\}})$ of the first part of the input. As shown at the bottom left of Figure 3, a coupling layer computes

$$z'_{\{b=1\}} = z_{\{b=1\}}$$
$$z'_{\{b=0\}} = z_{\{b=0\}} \odot \exp(s(z_{\{b=1\}})) + t(z_{\{b=1\}}), \quad (10)$$

where *s* and *t* are functions from $\mathbb{R}^{|\{b=1\}|} \to \mathbb{R}^{|\{b=0\}|}$, and $\odot$ is the element-wise product. Crucially, such a coupling layer is trivial to invert as shown at the bottom right of Figure 3,

$$z_{\{b=1\}} = z'_{\{b=1\}}$$
$$z_{\{b=0\}} = \left(z'_{\{b=0\}} - t(z_{\{b=1\}})\right) \odot \exp(-s(z_{\{b=1\}})), \quad (11)$$

and its Jacobian is a triangular matrix whose determinant is also trivial to obtain as $\exp\left(\sum_j s(z_{\{b=1\}})_j\right)$ [DSB16]. Neither operation requires inverting the Jacobians of *s* and *t* nor computing their inverses. Hence they can be arbitrarily complex, and we implement them using deep neural networks. We describe our network architecture to implement *s* and *t* in more detail below. Finally, we apply a linear scaling and a logit mapping at the start, and their inverse at the end of the model. Linear scaling layers are employed to ensure that values are within valid ranges. We found that this improves training convergence of our model.

In summary, we implement our PSS warp Ψ and its inverse using Real NVP with multiple stacked affine coupling layers. Since each bijective coupling layer only warps part of the input dimensions, we concatenate several coupling layers to warp all dimensions non-linearly. In our experiments, we use eight coupling layers and masks where either the even or odd bits are set to zero or one, respectively. We also experimented with other masks, but did not observe any significant differences. The warp parameters θ in
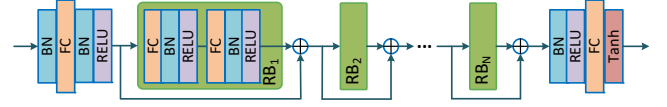
Equation 9 correspond to the trainable weights of the neural networks that define the *s* and *t* functions in all coupling layers (each layer has its own *s* and *t* functions, that is, neural networks). We describe our networks in more detail in the next section. We perform maximum likelihood estimation of Ψ^{-1} using gradient descent and standard backpropagation techniques for neural networks.

4.4. Neural Network Architecture for Coupling Layers

Figure 5 shows the neural network architecture that we use to compute both the *s* and *t* functions in Equations 10 and 11. Each function in each coupling layer has its own set of trainable network weights. The network consists of several blocks with residual connections [HZRS16]. Each block contains two fully connected layers (FC) with batch normalization (BN) and rectified linear unit (ReLU) activation $\sigma(x) = max(0, x)$, and a block can be bypassed via a residual connection. In addition, there are more such layers before and after the residual blocks. The warping capability of the model is directly affected by the number of coupling layers. More coupling layers enable the model to learn complex mappings. Meanwhile, using more coupling layers and residual blocks will lead to relatively long training time. We discuss the exact network parameters in Section 6.

4.5. Generating Rendering Samples

For rendering, we use the forward mapping Ψ, which can easily be derived from Ψ^{-1} as shown above. Note that Ψ reuses trained parameters of Ψ^{-1}. We generate an arbitrary number of samples distributed approximately according to the target density as represented by the learned warp, as illustrated in Figure 6.

In this paper, path tracing (PT) with next event estimation is utilized as the underlying rendering algorithm to show how it can be improved via our method. The construction of paths will consume PSS coordinates. Based on the practical simplifications in Section 4.1, we opt to learn to importance sample dimensions of the first *m* bounces (e.g. $m = 1, 2, 3, \ldots$). After *m* bounces, we continue to trace the path using uniform random PSS coordinates.

5. Implementation Details

In this section, we describe datasets used for training, validation and test. We also provide details of neural network training.
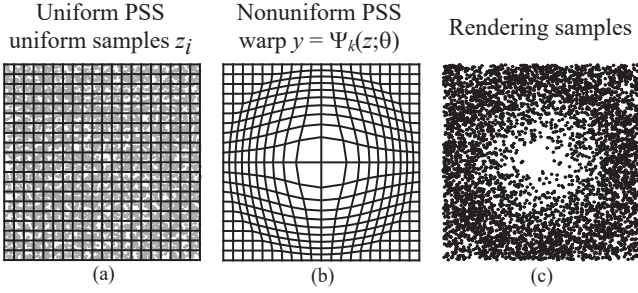
Uniform PSS
uniform samples z_i

Nonuniform PSS
warp $y = \Psi_k(z;\theta)$

Rendering samples

(a)   (b)   (c)

Figure 6: *During rendering, we use the forward warp Ψ to draw an arbitrary number of samples distributed approximately according to the target density as shown in this 2D illustration. (a) Uniform samples in PSS. (b) Warped PSS using Ψ. (c) Non-uniform samples in PSS, which we provide as input to a black-box renderer.*

Training Datasets. We set the number of examples (that is, samples y_i in Equation 9 in a training data set) to k times a certain image resolution, equivalent to k examples per pixel (*epp*) on average. For brevity, we will denote a dataset as *epp-k*. When generating training data we use small image resolutions. As shown in Table 2, we chose 160×88 for the first four scenes, and 100×100 for the TORUS scene. Note that final images can be rendered at any desired higher resolution. We empirically found that good resampling results are obtained when α is at least 6, and all experiments follow this heuristic. During training, 80% examples of a training dataset are randomly selected as training data and the rest are used for validation. Simultaneously, we provide an accompanying test dataset to test the performance of the neural networks. None of the test data are used in the training and validation process.

Model Initialization. For faster convergence during scene-dependent training, we pre-train our neural network to achieve an identity warp. We use the resulting network weights as initialization for scene-dependent training, instead of the usual random or Xavier initialization [GB10]. We have observed that this leads to faster scene-dependent training convergence in practice.

Training. We train neural networks corresponding to inverse warp Ψ^{-1} in an end-to-end fashion using the TensorFlow [AAB*15] framework. The networks are optimized using Adam [KB14] with a learning rate 10^{-4} and decay rates $\beta_1 = 0.9, \beta_2 = 0.99$. Training examples are fed into the neural network in mini-batches size of 2000, and the order of examples in a mini-batch is randomly shuffled in each iteration.

6. Results and Analysis

All our experiments are conducted on a workstation with an octa-core 3.60 GHz i7-7700 CPU and Nvidia GeForce GTX 1070 GPU. We implement our method based on PBRT [PJH16], and execute it in a hybrid way using both GPU and CPU. We perform neural network training and evaluation on the GPU and the renderer runs on the CPU. We firstly validate the neural network designs in Section 6.1 and 6.2, where experiments are in terms of 4D PSS learning for the COUNTRY KITCHEN scene. Then we apply our approach to five scenes with a range of challenging settings.
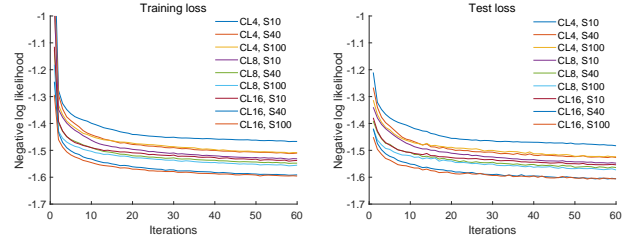

Figure 7: *Training and test loss of different neural network architectures. CL-i, S-j denotes i coupling layers and j neurons per hidden layer. The residual block count is set to 2 and training datasets use epp-16.*
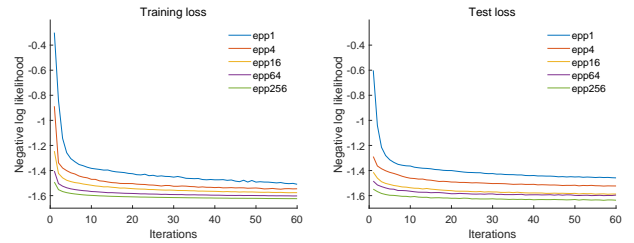

Figure 8: *Comparisons of training loss and test loss with respect to the size of training datasets. Sizes ranges from epp-1 to epp-256.*

6.1. Neural Network Architecture Validation

The capacity of our neural network to represent complex mappings is related to the number of coupling layers and the size of a hidden layer. To get an efficient sampling model, we tend to use a lightweight architecture with conservative numbers of coupling layers. In Figure 7, we compare different combinations of coupling layer count (CL) and hidden layer size (S). Each training session runs for 60 epochs. As can be seen, more coupling layers lead to lower loss values. Given a fixed number of coupling layers, increasing the number of neurons of hidden layers further reduce loss values. The deep and wide architecture provides the lowest loss values, while it incurs a long training time of several hours. We empirically find that eight coupling layers and 40 neurons per hidden layer can successfully learn 4D to 8D target densities in our experiments and keep sampling cost small.

6.2. Training Data Size and Density Error

Training data provides an implicit description of the target probability density function. To examine the effects of different numbers of training examples, we compare training and test losses with respect to training datasets of five different sizes. During model training, we use a low image resolution of 160×88 pixels. The dataset with the smallest size has one example for each pixel on average.

Figure 8 plots the training and test losses for each dataset. As shown, a lower negative log likelihood can be obtained with an increasing size of datasets, and the improvements brought by using more training data gradually reduce beyond *epp* 16.

In addition, Figure 9 shows relative L_1 errors of density values for both training and test dataset. The relative density errors is com-
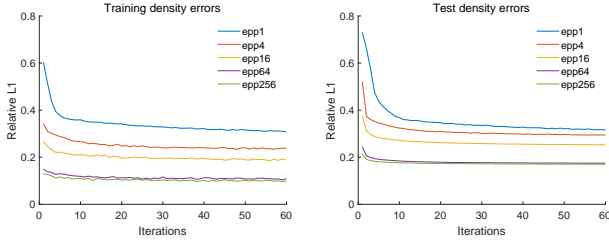
Figure 9: *Relative L_1 density error for training and test processes, with respect to different sizes of data sets. Each training session runs for 60 epochs.*

puted with

$$\varepsilon = \frac{1}{N} \sum_{i=1}^{N} \frac{|P(D_i) - T_i|}{max(P(D_i), T_i)}. \tag{12}$$

Here, $P(D_i)$ stands for the density deduced by neural networks, T_i represents the target density value of D_i and $|\cdot|$ indicates the absolute value. Target densities are readily obtained up to a normalizing constant in the resampling process. Using more training data helps to reduce relative density errors, but the improvement of density accuracy gets smaller with increasing size of datasets.

6.3. Rendering Results and Further Analysis

In Figure 11, we apply our method to render a range of challenging scenes. COUNTRY KITCHEN and WHITE ROOM contain complex visibility settings, and they are dominated by indirect illumination. SALLE DE BAIN, CLASSROOM and TORUS feature glossy and specular light transport. Our method treats the underlying Monte Carlo ray tracing method (path tracing) as a black box, and globally warps light transport paths in the PSS.

We compare our method with baseline path tracing (PT without PSS coordinates warping) and a recent kd-tree based PSS warping method [GBBE18]. Since our method and other methods differ in the rendering mode of GPU and CPU computations, we show equal sample budget comparisons. Timings for rendering and the ratios of zero-radiance paths are presented in Table 1. Our method can learn the distribution of path contributions and distributes samples according to the distribution, thus it effectively reduces the ratio of zero-radiance paths, which carry no radiance. Time cost of training neural networks is tabulated in Table 2. We apply two metrics to measure quality of images: mean squared errors (MSE) and Structural Similarity Index (SSIM) [WBSS04]. Note that we present the values $1 - SSIM$. Hence lower values indicate higher image quality. Finally, all reference images are rendered with at least 32768 samples per pixel (*spp*) to get noise-free ground truth.

As described in Section 4.5, we apply PSS warping to the path prefixes of one, two or three bounce(s), corresponding to dimensionalities of 4D, 6D and 8D in the primary sample space. In our experiments, next event estimation which is part of the black box, is enabled for path tracing. For the kd-tree based method [GBBE18], we use its default settings.

As can be observed in Figure 11, our method in 4D and 6D, corresponding to warping the first one or two bounce(s), generally

provides lower errors than the baselines. A similar finding is reported in [GBBE18], where they only warp the initial two bounces. Interestingly, warping more than six dimensions leads to slightly higher errors. This is mainly because the degrees of freedom of a PSS vector increase with the number of dimensions. It leads to overfitting issues that could be avoided by providing more training data, at the cost of additional training time. Additionally, since paths with more vertices typically make smaller contributions to the image than paths with fewer vertices, improved importance sampling of vertices further along a path brings decreasing benefits. A failure case of our method is the CLASSROOM scene, where our method does not reduce errors. This scene is equipped with many small-scale geometries, such as desks and chairs with glossy legs, which can undermine the correspondence between well-distributed PSS vectors and light transport paths.

In Figure 12, we demonstrate the ability of our method to render distribution effects, such as motion blur (POOL BALL scene) and defocus blur (CHESS scene). Motion blur is controlled by time samples and shutter opening intervals, and defocus blur is affected by lens samples and aperture radius. Therefore, we focus on warping only the first 3D or 4D of a PSS vector, corresponding to components of camera samples, and use uniform random data for the remaining dimensions. Compared to the baseline approach without PSS warping and kd-tree based warping, our method achieves lower numerical errors and better image quality.
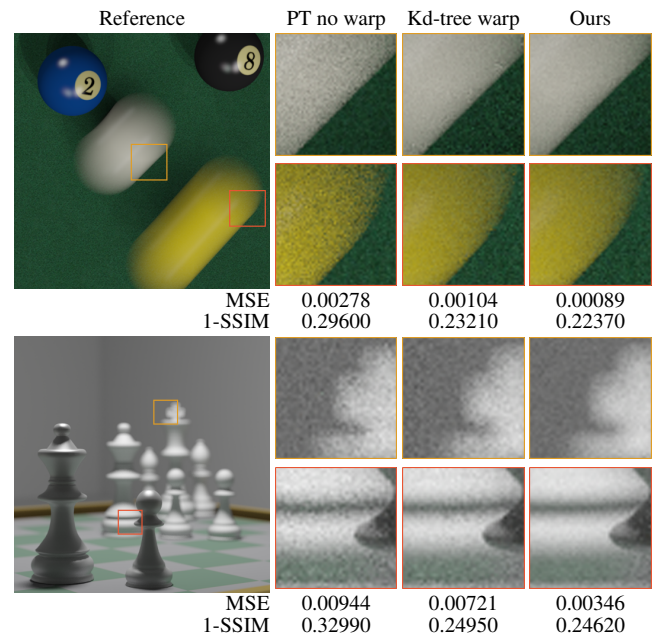

	MSE	0.00278	0.00104	0.00089
	1-SSIM	0.29600	0.23210	0.22370

	MSE	0.00944	0.00721	0.00346
	1-SSIM	0.32990	0.24950	0.24620

Figure 12: *Comparisons of distribution effects: motion blur (Top: POOL BALL) and depth of field (Bottom: CHESS). Training data set here uses epp-16.*

To obtain an interpretable profile of a warped distribution, we further visualize distributions of uniform random input samples and warped samples of the POOL BALL scene in Figure 13. In this case, 3D PSS warping provides us with an intuitively meaningful sampling distribution. As can be seen, our method is able to warp
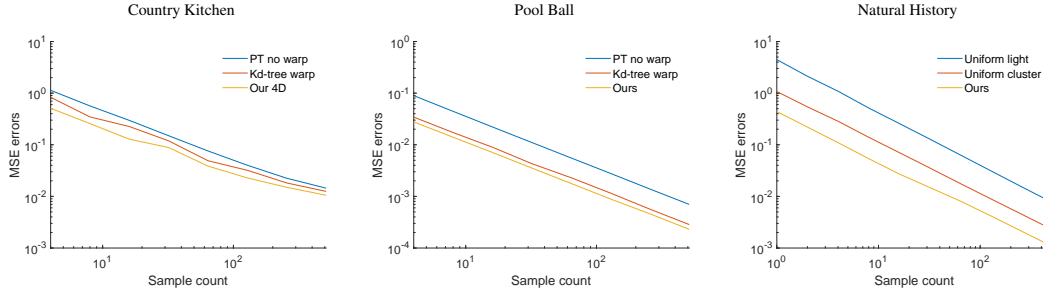
Figure 10: *MSE convergence. For the first two scenes, we compare our method against PT without PSS warp and kd-tree based warp. For the* NATURAL HISTORY *scene, we compare our method against uniformly sampling one light and uniformly sampling one cluster.*

Table 1: *Comparisons of time cost of rendering of different techniques and the ratios of zero-radiance paths (Black ratio). For all method, images are rendered with 16 spp using 2 threads. Timings for our method include both the sample generation and rendering process.*

Scene	PT no warp		kd-tree warp		Our 4D		Our 6D		Our 8D	
	Timing	Blk. ratio	Timing	Blk. ratio	Timing	Blk. ratio	Timing	Blk. ratio	Timing	Blk. ratio
Country Kitchen	65s	27.16%	136s	18.39%	170.8s	9.26%	206.3s	9.54%	238.6s	9.62%
Salle De Bain	54s	8.94%	121s	24.29%	177.4s	4.70%	186.2s	4.45%	221.5s	6.92%
White Room	69s	18.17%	162s	11.91%	184.3s	8.61%	231.1s	7.79%	272.2s	9.30%
Classroom	61s	69.79%	98s	62.74%	210.8s	59.60%	240.4s	60.29%	274.6s	60.67%
Torus	14s	19.86%	42s	42.65%	154.5s	13.28%	168.1s	20.60%	208.2s	14.45%

Table 2: *Time cost of training neural networks for 4D, 6D and 8D data in minutes (m). For each scene, the training example count is set to epp times the image resolution. Neural networks are trained for 60 epochs on an Nvidia GeForce GTX 1070 GPU.*

Scene	Example#	4D	6D	8D
Country Kitchen	$16 \times 160 \times 88$	19.4m	20.2m	20.8m
Salle De Bain	$16 \times 160 \times 88$	11.6m	12.1m	12.3m
White Room	$16 \times 160 \times 88$	15.0m	15.6m	15.9m
Classroom	$16 \times 160 \times 88$	17.0m	17.2m	17.5m
Torus	$16 \times 100 \times 100$	9.2m	9.4m	10.7m

samples to increase the density in the area corresponding to motion blur, while covering the rest of the domain with lower density. In contrast, the sample distribution of kd-tree warping tends to be blocky.

In addition, our method can be applied to render complex many-light direct illumination. The NATURAL HISTORY scene in Figure 14 contains 92 area light sources and one environment light map, posing a challenge to traditional light selection and accumulation techniques. We firstly build a light tree to organize all light sources and split the scene's space into multiple disjoint regions. Then we compute light clusters (a.k.a., a lightcut [WFA*05]) for each region. Subsequently, we importance sample a 3D PSS distribution to get coordinates for selecting components with high contribution. We compare our method with the baseline method of uniformly sampling each individual light and uniformly sampling each cluster of a lightcut. The baseline method is unaware of the visibil-

ity difficulty, thus most shadow rays are blocked and computation efforts are wasted. Our method can learn the shape of the manifold of light path samples in the PSS and generate new samples on the manifold, thus it achieves a significant error reduction.

Figure 10 illustrates MSE errors with respect to the average *spp*. We select one scene from each of the previous three application categories: PSS path warping, distribution effects, and many-light rendering. As can be seen, our method generally provides a consistently lower error and converges without introducing bias.

7. Discussion and Limitations

Computation cost. A downside of our approach is that it requires a training phase that takes on the order of minutes for typical warp dimensions up to eight. In addition, the accuracy of the density produced by our warp depends on the amount of training data, and using more training data increases training time. During rendering, the evaluation of neural networks to generate samples also incurs computational overhead. Fortunately, limiting the number of warping dimensions leads to a useful trade-off between training time and variance reduction.

Network weights reuse and retraining. Being a data-driven approach, our training correlates with the current camera view of a scene. Provided a change of camera view, we can reuse weights of a trained neural network of the previous camera view to initialize retraining. The reuse of network weights implicitly exploits the data coherence of different views of a scene, and it significantly accelerates retraining compared to training from scratch; see Section 1 of the supplementary material for experimental results.
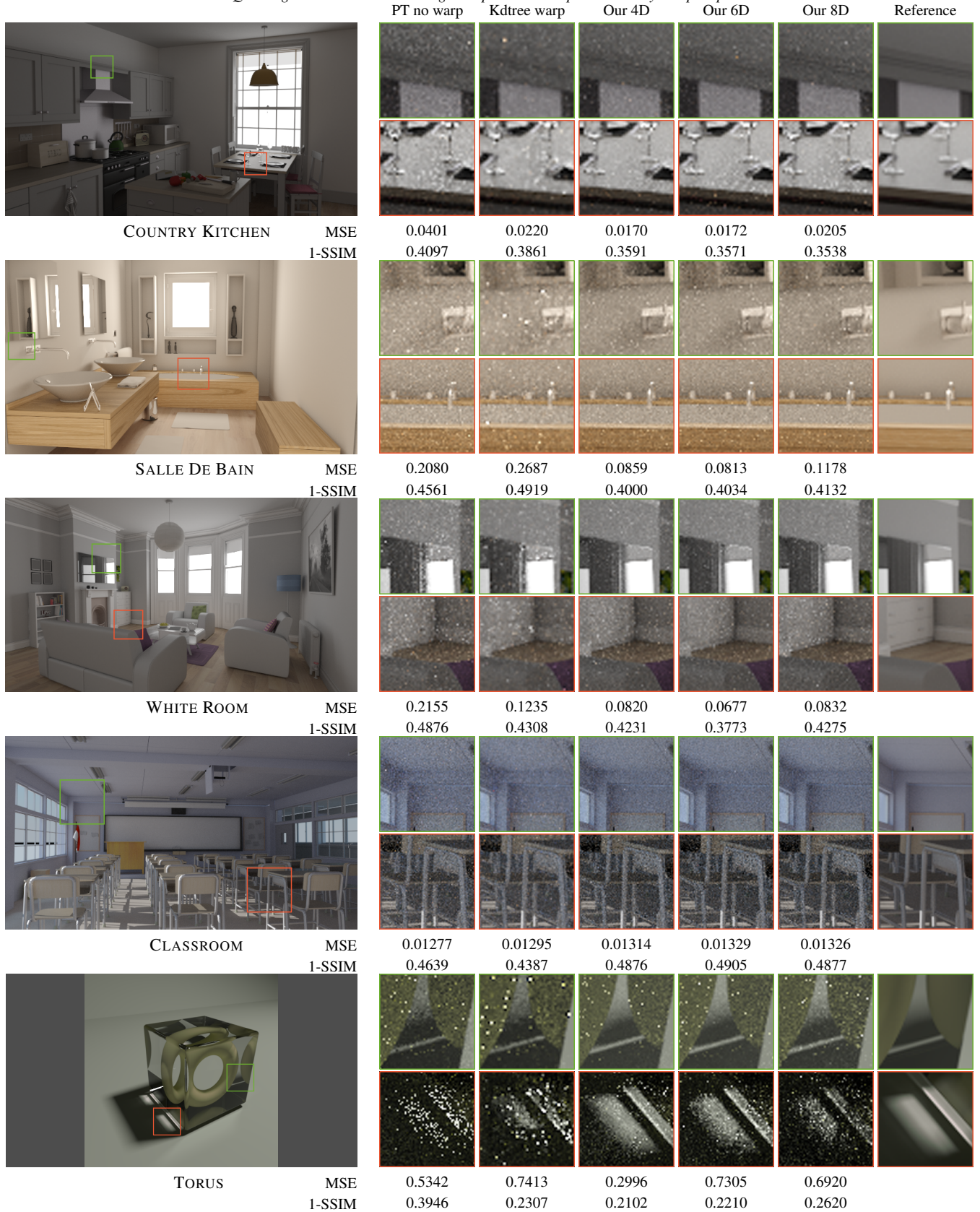
	PT no warp	Kdtree warp	Our 4D	Our 6D	Our 8D	Reference
COUNTRY KITCHEN MSE	0.0401	0.0220	0.0170	0.0172	0.0205	
1-SSIM	0.4097	0.3861	0.3591	0.3571	0.3538	
SALLE DE BAIN MSE	0.2080	0.2687	0.0859	0.0813	0.1178	
1-SSIM	0.4561	0.4919	0.4000	0.4034	0.4132	
WHITE ROOM MSE	0.2155	0.1235	0.0820	0.0677	0.0832	
1-SSIM	0.4876	0.4308	0.4231	0.3773	0.4275	
CLASSROOM MSE	0.01277	0.01295	0.01314	0.01329	0.01326	
1-SSIM	0.4639	0.4387	0.4876	0.4905	0.4877	
TORUS MSE	0.5342	0.7413	0.2996	0.7305	0.6920	
1-SSIM	0.3946	0.2307	0.2102	0.2210	0.2620	

Figure 11: *Equal sample count (128 spp) comparisons of path tracing without PSS warping, kd-tree based PSS warping [GBBE18], and our approach. We warp the first one, two, and three path bounce(s), corresponding to 4D, 6D and 8D in PSS. The training datasets use epp-16. Due to the small training data, warping in 6D and 8D does not provide benefits over 4D warping in most scenes.*
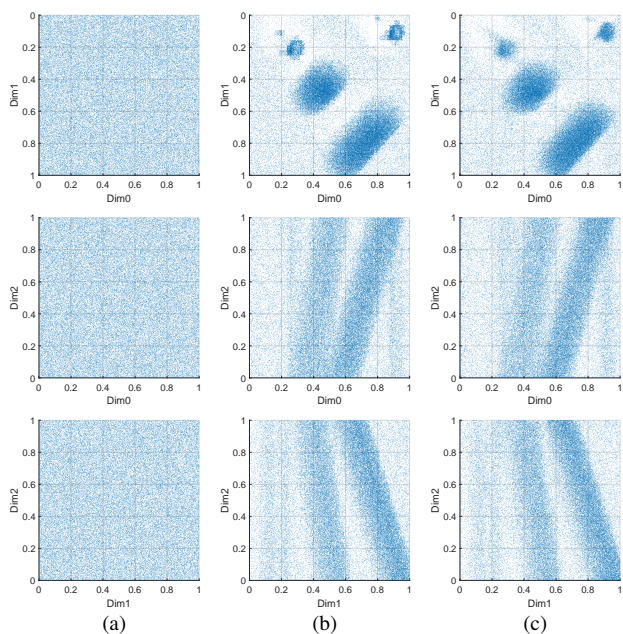
Figure 13: *Visualizations of sample distributions of the* POOL BALL *scene's 3D warping. We compare distributions of 80,000 uniform random input samples (a), warped samples of kd-tree based approach (b) and our method (c). Our method distributes more samples to domain corresponding to motion blur and better cover the whole sampling domain, while the sample distribution of kd-tree warping tends to be blocky.*

Higher dimensional warps. We also experimented with higher-dimensional warps up to 12D (see the supplementary document), and were able to obtain error reductions compared to baseline path tracing up to 12D. This requires more training data and time, however, and we generally did not observe any further error reduction compared to 4D or 6D warps. Hence taking advantage of higher dimensional warps in a practical scenario is an interesting avenue for future research.

Efficiency of resampling. The resampling process includes a step to draw initial examples to form a candidate set and another step to resample examples from the set. Both steps have linear complexity in terms of the number of examples. Since the original distribution is represented by discrete examples in the candidate set, an accurate representation requires a relatively large candidate set and more processing time. Our further experiments (see Section 3 of the supplementary material) investigate the candidate set sizes in terms of a trade-off between efficiency and quality of resampling.

Small illumination features. Small-scale caustics with fine structures and glints, caused by specular or highly glossy light transport, are generally hard to render for many methods. Path tracing based on uniform PSS sampling may "miss" these small features, since it is inefficient to sample light paths for such features. In the supplementary material, we compare our method against baseline path tracing to show that our approach can efficiently sample small illumination features given enough training data.
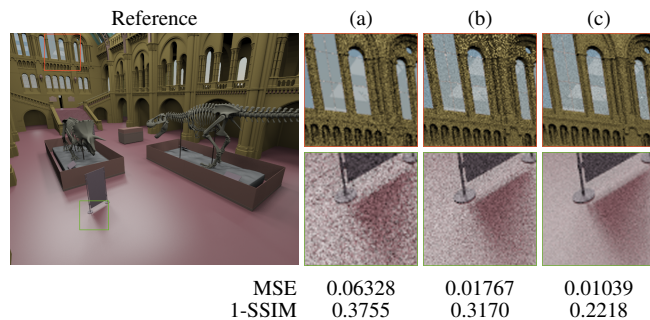

	Reference	(a)	(b)	(c)
MSE		0.06328	0.01767	0.01039
1-SSIM		0.3755	0.3170	0.2218

Figure 14: *Visual comparisons for complex many-light rendering. We compare uniformly sampling one light (a), uniformly sampling one cluster (b) with our method (c), under the the same sample budget of 64 spp. The training dataset uses epp-16.*

Comparison to caching based approaches. In addition to neural network approaches, previous tabulation based methods applied caching structures like histogram [DK17], octree, quadtree [MGN17] or local Gaussian-mixture models [VKŠ*14] to learn local, directional sampling densities in an *a posterior* manner. These structures are embedded in the path space and they conduct low-dimensional warping in local regions. In contrast, our method works in a PSS space to learn a sampling distribution, and it treats the underlying rendering algorithm as a black box. This makes it more general to be applied to multiple tasks of importance sampling, such as light path sampling, distribution effects' multi-dimensional sampling, and many-light sampling as shown in the previous section.

8. Conclusions and Future Work

We introduced a novel approach to learn importance sampling of entire light paths in primary sample space using a suitable deep neural network architecture. We leverage the neural network to perform a non-linear warp in primary sample space, achieving a desired target density that can further reduce the variance of an existing rendering algorithm, which is treated as a black box by our method. Our experiments demonstrated that this approach can effectively reduce variance in practical scenarios without introducing bias. A main advantage of our approach is that it is agnostic of specific light transport effects in any scene, and the underlying renderer. Therefore, it is easy to implement on top of existing systems. For future work, reducing the computation costs of our approach is an important and meaningful direction. Another interesting avenue for future research is to extend our maximum likelihood approach to a Bayesian framework with a prior, which could allow more effective scene-dependent training using fewer samples.

References

[AAB*15] ABADI M., AGARWAL A., BARHAM P., BREVDO E., ET AL.: TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. URL: https://www.tensorflow.org/. 6

[BJNJ17] BITTERLI B., JAKOB W., NOVÁK J., JAROSZ W.: Reversible jump metropolis light transport using inverse mappings. *ACM Transactions on Graphics (TOG) 37*, 1 (2017), 1. 2

[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 Trans. Graph. 36*, 4 (July 2017), 97:1–97:14. 2

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

[CKS*17] CHAITANYA C. R. A., KAPLANYAN A., SCHIED C., SALVI M., LEFOHN A., NOWROUZEZAHRAI D., AILA T.: Interactive Reconstruction of Monte Carlo Image Sequences using a Recurrent Denoising Autoencoder. *ACM Transactions on Graphics* (Aug 2017). 2

[DK17] DAHM K., KELLER A.: Learning light transport the reinforced way. *arXiv preprint arXiv:1701.07403* (2017). 2, 10

[DSB16] DINH L., SOHL-DICKSTEIN J., BENGIO S.: Density estimation using real NVP. *CoRR abs/1605.08803* (2016). 2, 4, 5

[GB10] GLOROT X., BENGIO Y.: Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (2010), pp. 249–256. 6

[GBBE18] GUO J., BAUSZAT P., BIKKER J., EISEMANN E.: Primary sample space path guiding. In *Eurographics Symposium on Rendering - EI & I* (July 2018), Jakob W., Hachisuka T., (Eds.), Eurographics, The Eurographics Association, pp. 73–82. DOI = 10.2312/sre.20181174. 7, 9

[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 27*, Ghahramani Z., Welling M., Cortes C., Lawrence N. D., Weinberger K. Q., (Eds.). Curran Associates, Inc., 2014, pp. 2672–2680. 2

[Hd14] HEITZ E., D'EON E.: Importance sampling microfacet-based bsdfs using the distribution of visible normals. *Computer Graphics Forum 33*, 4 (2014), 103–112. 2

[HKD14] HACHISUKA T., KAPLANYAN A. S., DACHSBACHER C.: Multiplexed metropolis light transport. *ACM Transactions on Graphics (TOG) 33*, 4 (2014), 100. 2

[HZRS16] HE K., ZHANG X., REN S., SUN J.: Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2016), pp. 770–778. 5

[JM12] JAKOB W., MARSCHNER S.: Manifold exploration: a markov chain monte carlo technique for rendering scenes with difficult specular transport. *ACM Transactions on Graphics (TOG) 31*, 4 (2012), 58. 2

[Kaj86] KAJIYA J. T.: The rendering equation. In *ACM Siggraph Computer Graphics* (1986), vol. 20, ACM, pp. 143–150. 1, 2, 3

[KB14] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014). 6

[KHD14] KAPLANYAN A. S., HANIKA J., DACHSBACHER C.: The natural-constraint representation of the path space for efficient light transport simulation. *ACM Transactions on Graphics (TOG) 33*, 4 (2014), 102. 2

[KMM*17] KALLWEIT S., MÜLLER T., MCWILLIAMS B., GROSS M., NOVÁK J.: Deep scattering: Rendering atmospheric clouds with radiance-predicting neural networks. *ACM Trans. Graph. 36*, 6 (Nov. 2017), 231:1–231:11. 2

[KSKAC02] KELEMEN C., SZIRMAY-KALOS L., ANTAL G., CSONKA F.: A simple and robust mutation strategy for the metropolis light transport algorithm. In *Computer Graphics Forum* (2002), vol. 21, Wiley Online Library, pp. 531–540. 1, 2, 3

[KW14] KINGMA D. P., WELLING M.: Auto-encoding variational bayes. In *ICLR* (2014). 2

[LLR*15] LI T.-M., LEHTINEN J., RAMAMOORTHI R., JAKOB W., DURAND F.: Anisotropic gaussian mutations for metropolis light transport through hessian-hamiltonian dynamics. *ACM Transactions on Graphics (TOG) 34*, 6 (2015), 209. 2

[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, 10

[MMR*18] MÜLLER T., MCWILLIAMS B., ROUSSELLE F., GROSS M., NOV'AK J.: Neural importance sampling. *arXiv preprint arXiv:1808.03856* (2018). 2

[OKH*17] OTSU H., KAPLANYAN A. S., HANIKA J., DACHSBACHER C., HACHISUKA T.: Fusing state spaces for markov chain monte carlo rendering. *ACM Transactions on Graphics (TOG) 36*, 4 (2017), 74. 2

[PJH16] PHARR M., JAKOB W., HUMPHREYS G.: *Physically Based Rendering: From Theory to Implementation*, 3 ed. Morgan Kaufmann, 2016. 2, 6

[RWG*13] REN P., WANG J., GONG M., LIN S., TONG X., GUO B.: Global illumination with radiance regression functions. *ACM Trans. Graph. 32*, 4 (July 2013), 130:1–130:12. 2

[TCE05] TALBOT J. F., CLINE D., EGBERT P.: Importance resampling for global illumination. In *Proceedings of the Sixteenth Eurographics Conference on Rendering Techniques* (Aire-la-Ville, Switzerland, Switzerland, 2005), EGSR '05, Eurographics Association, pp. 139–146. 4

[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), ACM, pp. 419–428. 2

[VG97] VEACH E., GUIBAS L. J.: Metropolis light transport. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (1997), ACM Press/Addison-Wesley Publishing Co., pp. 65–76. 2, 3

[VKK18] VÉVODA P., KONDAPANENI I., KŘIVÁNEK J.: Bayesian online regression for adaptive direct illumination sampling. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2018) 37*, 4 (2018). 2

[VKŠ*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 (2014), 101. 2, 10

[WBSS04] WANG Z., BOVIK A. C., SHEIKH H. R., SIMONCELLI E. P.: Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing 13*, 4 (2004), 600–612. 7

[WFA*05] WALTER B., FERNANDEZ S., ARBREE A., BALA K., DONIKIAN M., GREENBERG D. P.: Lightcuts: A scalable approach to illumination. *ACM Trans. Graph. 24*, 3 (July 2005), 1098–1107. 8