Eurographics Symposium on Rendering 2023
T. Ritschel and A. Weidlich
(Guest Editors)

NEnv: Neural Environment Maps for Global Illumination

Carlos Rodriguez-Pardo*[1,2] and Javier Fabre*[1,2] and Elena Garces[1,2] and Jorge Lopez-Moreno[1,2]

[1]SEDDI, Madrid, Spain
[2]Universidad Rey Juan Carlos, Madrid, Spain
* Denotes equal contribution

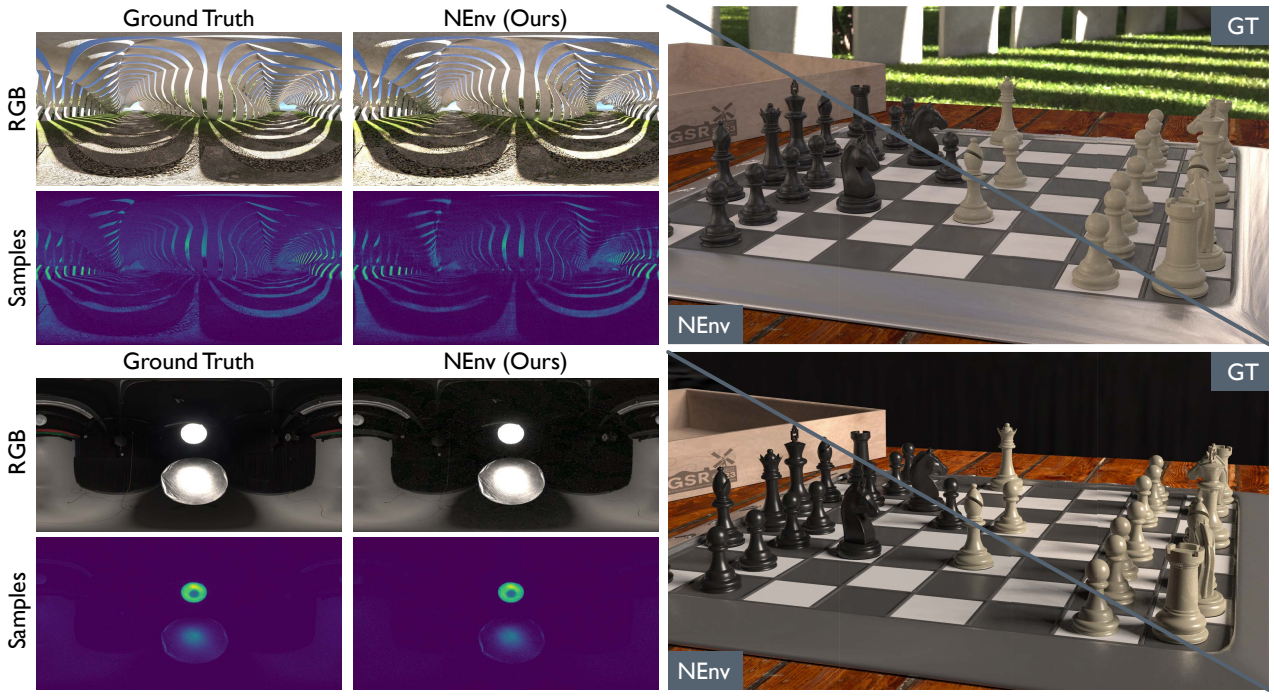

Figure 1: *We introduce NEnv, an invertible and fully differentiable neural method which achieves high-quality reconstructions for environment maps and their probability distributions. NEnv is up to two orders of magnitude faster to sample from than analytical alternatives, providing fast and accurate lighting representations for global illumination using Multiple Importance Sampling. Our models can accurately represent both indoor and outdoor illumination, achieving higher generality than previous work on environment map approximations.*

Abstract

Environment maps are commonly used to represent and compute far-field illumination in virtual scenes. However, they are expensive to evaluate and sample from, limiting their applicability to real-time rendering. Previous methods have focused on compression through spherical-domain approximations, or on learning priors for natural, day-light illumination. These hinder both accuracy and generality, and do not provide the probability information required for importance-sampling Monte Carlo integration. We propose NEnv, a deep-learning fully-differentiable method, capable of compressing and learning to sample from a single environment map. NEnv is composed of two different neural networks: A normalizing flow, able to map samples from uniform distributions to the probability density of the illumination, also providing their corresponding probabilities; and an implicit neural representation which compresses the environment map into an efficient differentiable function. The computation time of environment samples with NEnv is two orders of magnitude less than with traditional methods. NEnv makes no assumptions regarding the content (i.e. natural illumination), thus achieving higher generality than previous learning-based approaches. We share our implementation and a diverse dataset of trained neural environment maps, which can be easily integrated into existing rendering engines.

CCS Concepts

• ***Computing methodologies*** → ***Neural networks;*** *Image-based rendering; Image representations;*

1. Introduction

Environment Maps are widely used in rendering to represent far-field illumination around a point with a single texture, usually a High Dynamic Range image (HDRi). There are two typical scenarios for using these maps: as infinite spherical light sources in offline rendering methods (such as path tracing), or as local probes that encode near-field irradiance to approximate global illumination in real-time applications.

In the first case, many algorithms such as Multiple Importance Sampling (MIS) [VG95] require the ability to sample lights in the scene, which can be problematic when dealing with environment maps. Since the source data comes from an image, there is no analytic Cumulative Distribution Function (CDF) than can be used for sampling. Instead, tabulated methods can be used, but these require a pre-computation step that consumes a considerable amount of memory, as well as a search step to find the sample probability.

In real-time scenarios, irradiance sampling is becoming more relevant with the increasing capabilities of GPUs and sample reservoir techniques for global illumination [BWP*20, OLK*21], yet the usual techniques to sample environment maps do not easily benefit from parallel GPU environments. Also, in real-time, analytical approximations (such as Spherical Harmonics or Spherical Gaussians) are often used to overcome this issue, however they are not able to capture final-scale details present in most environment maps.

This work presents a novel method for sampling, PDF evaluation, and compression of high-resolution environment maps, able to encode any type of illumination with high-fidelity. Our method builds on recent advances on neural representations, that we found particularly suitable for this kind of problems where it is important to encode both the high-frequency and low-frequency patterns. Key to our solution is to use a normalizing flow [DBMP19a] to encode an *invertible* representation of the environment map PDF, reducing both sampling and evaluation time by two orders of magnitude compared with analytic solutions. Thanks to being invertible, our representation is differentiable and compatible with importance-sampling techniques. We also propose a method based on implicit neural representations to compress the environment map reducing the memory footprint of the original image by one order of magnitude with minimal loss. We demonstrate that our approach works for a wide range of scenes and is more accurate and faster than previous methods. We provide a dataset of ground truth environment maps and trained models, along with an open source implementation at our project website.

2. Related Work

2.1. Environment Map Representations

In offline rendering, environment maps are usually encoded in high resolution ($> 4K$) HDR images (Radiance HDR [LS98], OpenEXR [KBH03]), requiring tabulated approximations for sampling and evaluation. Techniques such as Piecewise–Constant 2D Distributions [PJH16] or Hierarchical Warping [CJAMJ] are used for this purpose, but they are not suited for RT applications due to limited time and memory budget, being a costly and difficult-to-parallelize operation, even for offline rendering. This cost can be reduced, for instance, by taking into account scene information, such as occluders [ARBJ03] to avoid poor-quality samples.

Heitz [Hei20] proposed a method to invert non–analytically invertible CDFs, which can be applied to some distribution functions (e.g.: BSDFs) to obtain an analytical sampling function. However, environment maps are not a good candidate for triangle-cut parametrization, since their PDF and CDF come from a discrete tabulated source (HDR image).

To avoid using high resolution images, RT methods rely on environment map prefiltering [KVHS00], or analytical approximations, such as Spherical Harmonics (SH) [See66], and Spherical Gaussians (SG) [XSD*13]. These methods allow for analytical evaluation of an environment map, as well as analytical sampling (directly in the case of SG, using Hierarchical Sampling for SH [JCJ09], or with lookup tables in [SK13]) and easy interpolation (useful when not enough samples can be requested). Still, they do not accurately represent complex HDR images commonly used when representing high–detailed light setups by using environment maps, and blend important light features in the original images, sometimes generating artifacts. More recently, neural representations have been explored for this purpose. For instance, Gardner et al. [GHGS*19] propose a conditional neural field representation, based on a variational auto-decoder (RENI), which leverages natural image priors to efficiently encode a full HDR environment image into a small-dimension latent space vector (10 to 300 dimensions). Being low-dimensional, it shares the same accuracy limitations than its predecessors, and does not consider sampling, nor evaluation in its design. In the following we describe the most relevant neural approaches for sampling and 2D image representations.

2.2. Neural Sampling and Representations

Learned Sampling and Normalizing Flows Normalizing Flows (NFs) have been proposed as powerful models for learned sampling for rendering applications. The seminal work of Müller et al. [MMR*19] proposed *Neural Importance Sampling*, using *Piecewise-Linear* and *Piecewise-Quadratic Coupling Flows* for learned sampling for Monte Carlo rendering. This method showed the first successful use of these neural models for Path-Tracing, but their results showed a prohibitive overhead in runtime cost over previous methods. On a later work, Müller et al. proposed *Neural Control Variates* [MRKN20], an *Autorregresive Flow* which allows for more efficient unbiased integration. Beyond Monte Carlo integration, Flows have been used for BRDF representations in inverse rendering [CNN22]. Relatedly, Sztrajman et al. [SRRW21] propose a method for neural sampling of BRDFs. However, instead of relying on NFs, they propose an encoder network which maps from *Neural BRDF* to fitted Blinn-Phong parameters for which importance sampling is known. Learned importance sampling has also been studied for complex luminaires rendering [ZBX*21]. Besides these tasks, NFs have been utilized in computer graphics and vision for image [KD18, WZY22] and video [KBE*19] generation, compression [HDGS20], super-resolution [LDVGT20], domain translation [GCS*20], and uncertainty quantification [WLM*22, SFMP20]. We build upon *Neural Importance Sampling* [MMR*19] and propose a lightweight model for sampling and PDF evaluation of environment maps, as we de-

scribe in Section 4. We show that the proposed architecture and training produces a single network which can be integrated in any rendering pipeline, providing a significant speedup in image-based lighting with importance sampling.

Implicit Neural Representations (INRs) have emerged in recent years as powerful alternative to traditional representations for natural signals. They allow for differentiable, continuous and expressive functions which can represent many types of data, such as images [SMB*20, MGB*21], video [GSKH21], or *Neural Fields* for rendering [MST*21]. INRs typically build upon standard Multi Layer Perceptrons (MLPs) but benefit from working on the frequency domain to better handle higher frequencies, which are present on natural signals. These can be introduced to the models using fixed *Positional Encodings* [MST*21, BMT*21], *Fourier Features* [TSM*20, KAL*21] or *Sinusoidal Activations* [GES22, MGB*21, SMB*20], among other approaches. These general-purpose models have been utilized extensively in inverse and neural rendering, for representing geometry [TLY*21, YBHK21], illumination [GES22, GMX22, AHZ*22], material reflectance [BGP*22, FWH*22, SRRW21, KWM*22, Kuz21, YZL*22] or entire scenes [MST*21, FKYT*22, TCY*22, VHM*22, SRF*21]. We refer the reader to the seminal surveys in *Neural Fields* [XTS*22, TTM*22] for more comprehensive reviews of these representations for neural rendering and computer vision. Besides their effectiveness for rendering, INRs have also been used for image [SPY*22] and video [RCKP22] compression. We build upon these findings to design an efficient yet expressive INR for environment maps, as described in Section 5.

3. Overview

Our method takes as input a single HDRi environment map of any resolution, which is an image-based 360º representation of the illumination of a scene $\mathcal{I} \in \mathbb{R}^{H\times W\times 3}$. First, we perform a series of pre-processing steps (Section 3.1) and compute the ground truth probability distribution function,(Section 3.2). Then, we train a normalizing flow $\mathcal{F}$ for joint sampling and PDF evaluation (Section 4), and an implicit neural representation $\mathcal{C}$ for environment map compression (Section 5). We illustrate NEnv in Figure 2.

$\mathcal{F}$ is an invertible neural network which can generate directional samples $(\theta,\phi) = \mathcal{F}(z)$ by mapping from a known and simple noise distribution $z \in \mathbb{R}^2, z \sim p_z(z)$, as well as evaluating the probability density of any direction $p(\theta,\phi)$. $\mathcal{F}$ provides an efficient and differentiable approximation of the PDF encoded in $\mathcal{I}$, which is particularly useful for Monte Carlo integration in rendering systems using Multiple Importance Sampling (MIS). We model $\mathcal{C}$ as a sinusoidal Multilayer Perceptron (MLP) which maps directions (θ,ϕ) to RGB values: $RGB = \mathcal{C}(\theta,\phi)$; compressing $\mathcal{I}$ into a differentiable, continuous function. We train $\mathcal{F}$ and $\mathcal{C}$ individually for each input $\mathcal{I}$, and design their architectures and training configurations to achieve high compression rates, efficient evaluation, and minimal loss in rendering quality. Once trained, both $\mathcal{F}$ and $\mathcal{C}$ can be integrated seamlessly into rendering engines, considerably reducing the time and memory consumption per sample compared to traditional global illumination approaches.

Implementation details are included in Section 6 and in the supplementary material. To validate our method, we perform extensive ablation studies, and perform comparisons with previous work on environment map approximations (Section 8), both in environment map reconstruction and its impact on final renders.

3.1. Map Pre-processing

Some environment maps have large and intense light sources close to the horizontal borders of the image, separating the same light source into extreme values of $\phi \approx 0$ and $\phi \approx 2\pi$. While this is not a problem when rendering with traditional approaches, we observe these cases become very challenging for our normalizing flows, creating artifacts and tails in the encoded PDFs, as we show in Figure 3. We solve this by rotating the environment map by $\pi - \phi_{max}$ angles, so that its most intense light source is in $\phi = \pi$, thus avoiding strong discontinuities at the borders:

$$\phi_{max} = \arg\max_{\phi} \sum_{\theta} \mathcal{I}(\theta,\phi) \tag{1}$$

While there are specific parameterizations for normalizing flows on spheres [RPR*20], we empirically observe that this rotation algorithm effectively solves the discontinuities issues we observed on real-world environment maps without requiring any modifications in our model design. As shown in Figure 3, despite the differences on the encoded PDF, the renders converge to the same image.

3.2. Computing Ground Truth Tabular PDFs

As mentioned, there are two widely–used approaches for sampling directions according to an environment map distribution:

- Computing a *Piecewise-Constant* [PJH16] 2D Distribution by sampling a marginal PDF to select a row/column of $\mathcal{I}$, then sampling a conditional PDF to choose an individual pixel.
- A Hierarchical Warping [CJAMJ, Pha19] algorithm based on MIP–mapping over $\mathcal{I}$ to iteratively warp 2D samples from an uniform space until they match the target distribution.

Pharr [Pha19] compared these methods, stating that despite obtaining different spatial warpings, both produce similar error in renders. Interestingly, the official implementation for Mitsuba3 [JSR*22] relies on the later approach, while the available implementation for the 4th edition of PBRT [PJH20] uses the former. These are both widely–used engines in the literature. Due to its comparably better efficiency (see Section 8.3), which is important during model training, we choose to use the Piecewise-Constant method to compute our ground truth PDFs and samples.

To obtain the PDF for a given environment map $\mathcal{I}$, we first define $f(\theta,\phi)$, by mapping directions to the luminance values stored in the image $\mathcal{I}$. These values are proportional to the probability of selecting a particular pixel that we want to compute, but require additional processing and normalization. We start by multiplying $\mathcal{I}$ by $\sin(\theta)$, given the θ corresponding to each row, thus eliminating the distortion caused by mapping the image to the unit sphere [PJH16]. Using $f(\theta,\phi)$ as source data, we then compute one conditional density distribution $f_{conditional_\theta}(\phi)$ for each ϕ, so we can later obtain a single marginal distribution by integrating all conditional distributions (rows) to build 1D Piecewise distributions. We use this

Figure 2: *Overview of* NEnv. *We propose an invertible generative neural network $\mathcal{F}$ (i.e. a* normalizing flow*) which simultaneously learns to sample directions (θ,ϕ) from an environment map, as well as evaluating the probability of a given direction. This model, illustrated on the left, can be integrated seamlessly into path tracer render engines, allowing for efficient multiple importance sampling. We additionally use a implicit neural representation $\mathcal{C}$ (second column) which learns to map from directions to linear RGB values, allowing for faster evaluation.*

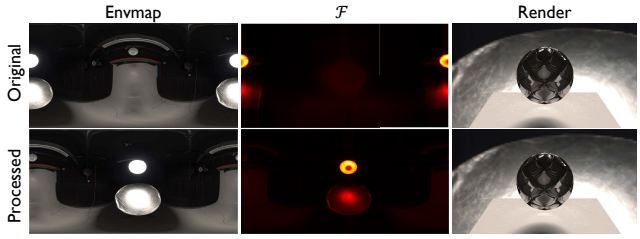

Figure 3: *Environment map, the PDF encoded by NEnv, and renders for an unprocessed (top) and rotated (bottom) environment map. For visualization purposes, we rotate the illuminations so both scenes match. Our simple rotation preprocessing avoids discontinuities in the borders of the learned reconstructions, enhancing the performance of our models.*

marginal distribution to sample a row, and in turn, sample the corresponding conditional distribution stored for such row to select a column, hence getting a single pixel defined by θ,ϕ coordinates.

Finally, to compute the probability density of choosing a given pixel $p_{\mathcal{I}}(\theta,\phi)$, we obtain the CDF for the conditional distribution at row ϕ for value θ, and normalize it to the marginal distribution integral:

$$p_{\mathcal{I}}(\theta,\phi) = \frac{f_{conditional_\theta}(\phi)}{\sum_{i=0}^{n_\theta} f_{marginal}(i)} \tag{2}$$

where $f_{conditional_\theta}$ represents the data used to compute the conditional PDF for row ϕ, and $f_{marginal}$ is the data used to compute the marginal distribution over the n_θ rows of the Y axis.

4. Learned Sampling and PDF evaluation

We aim to find a learning-based representation useful to sample directions from an environment map following its distribution, as well as evaluating the probability density (PDF) for any given direction $p(\theta,\phi)$. While this could be approached using two independent neural networks, there is no guarantee that a sample generated by a network is drawn with the probability estimated by the other network, which is a requirement for unbiased Monte Carlo integration, thus very often producing visual artifacts (*e.g.* fireflies). Neural networks by default are typically non-invertible and can become prohibitively expensive once model sizes grows, either by stacking more layers or adding more neurons. We instead leverage *normalizing flows*, defined by specifying a bijective function $\mathcal{F}$ or its inverse $\mathcal{F}^{-1}$. This enables both sampling and evaluation simultaneously and coherently with a single invertible neural network.

4.1. Formalization

We define our normalizing flow as a differentiable transformation $\mathcal{F}$ which maps from a random vector $z \in \mathbb{R}^2$, sampled from a 2-dimensional uniform distribution $p_z(z)$, to samples $(\theta,\phi), \theta \in [0,\pi)), \phi \in [0,2\pi)$:

$$(\theta,\phi) = \mathcal{F}(z) \text{ where } z \sim p_z(z) \tag{3}$$

The probability density of a direction $p_{\mathcal{F}}(\theta,\phi) \in \mathbb{R}$ under the normalizing flow $\mathcal{F}$ is obtained through a change of variables. This can be computed analytically because $\mathcal{F}$ is designed as an invertible function. Formally:

$$p_{\mathcal{F}}(\theta,\phi) = p_z(\mathcal{F}^{-1}(\theta,\phi)) \left| \det \frac{\delta \mathcal{F}^{-1}}{\delta(\theta,\phi)} \right| \tag{4}$$

$\mathcal{F}$ locally transforms the density of original noise distribution $p_z(z)$. The intensity of this change if measured by the determinant of the Jacobian of the transformation, as in the right hand side of the equation above. $p_z(z)$ is chosen to be a bidimensional uniform distribution bounded between 0 and 1: $p_z(z) = \mathcal{U}^2(0,1)$.

As shown, $\mathcal{F}$ provides the two calculations required in MIS for Monte Carlo integration. However, $\mathcal{F}$ needs to meet two competing requirements: it has to be as accurate as possible, *i.e.*, close to the real PDF, while being computationally efficient, both in evaluation and sampling. These competing requirements define our design choices, which we motivate next.

4.2. Design of $\mathcal{F}$

The design of $\mathcal{F}$ follows three principles: First, to achieve accurate renderings, we need it to be expressive enough to represent any complex probability density distribution. Second, we need an analytic form to compute its inverse $\mathcal{F}^{-1}$. Finally, for efficiency,

computing $\mathcal{F}$, $\mathcal{F}^{-1}$, and its Jacobian determinant needs to be as fast as possible.

Coupling Flows [DKB14, DSDB16, KD18] achieve the efficiency requirements by using a single feed-forward pass and preserving a reduced level of computation, required to evaluate the Jacobian determinant which is a lower triangular matrix for this type of flows. However, because all computation is done on a single pass to the models, coupling flows are typically less expressive than *Autoregressive Flows* [MRKN20, KSJ*16, PPM17, HKLC18, PSM19,KMLH21], which in turn, are less efficient during test time. We refer the reader to recent surveys [KPB20, PNR*21] for more comprehensive analyses of these topics.

To balance between efficiency and accuracy, we design $\mathcal{F}$ as a *Coupling Flow* and incorporate expressiveness through the coupling layer design. To this end, we evaluate several design choices available in the literature in the context of our problem. Recent work include *Affine Coupling* layers [DSDB16], *Piecewise Linear* and *Piecewise Quadratic* [MMR*19], *Cubic-Spline Flows* [DBMP19b] and *Rational-Quadratic Spline Flows* [DBMP19a]. The type of coupling controls the complexity of $\mathcal{F}$, which can further be tuned by modifying the number of *coupling layers*, the *width* and *depth* of the neural network that define each coupling, and the number of *bins* which define the complexity of the polynomial function inside each coupling layer. In addition, as in any neural network, it is possible to tweak the internal normalization layers, activation functions, optimizer type and configuration, regularization choices, etc. We found that *Rational-Quadratic Spline Flows* [DBMP19a] with a reduced amount of small coupling layers are expressive to a sufficient degree for every environment map that we tested, while being very efficient during evaluation, achieving orders of magnitude less computational cost than tabulated sampling. Because $\mathcal{F}$ is efficient and fully differentiable, it could potentially be incorporated into inverse rendering problems. We specify our model design choices in Section 6 and evaluate them in Section 8.1.

4.3. Training $\mathcal{F}$

We train our normalizing flow to minimize the aggregated negative log-likelihood of a set of samples, $\mathcal{B} = \{(\theta,\phi)^{(n)}\}_{n=1}^{N}$, that are drawn dynamically in batches of size N from the input environment map $\mathcal{I}$ as described in Section 3.2. During each training step, we minimize $\frac{1}{N}\sum_b^N \mathcal{L}_{nll}(\theta,\phi)^{(b)}$ where:

$$\begin{aligned}\mathcal{L}_{nll}(\theta,\phi) &= -\log(p_{\mathcal{F}}(\theta,\phi)) \\ &= -\log\left(p_z(\mathcal{F}^{-1}(\theta,\phi))\left|\det\frac{\delta\mathcal{F}^{-1}}{\delta(\theta,\phi)}\right|\right)\end{aligned} \tag{5}$$

This is is done by learning to map from the distribution of samples to the noise distribution $p_z(z)$. Given a sufficient number of training batches, $\mathcal{F}$ learns to accurately sample from the distribution of samples encoded in the input environment map $\mathcal{I}$. Because $\mathcal{F}$ is invertible and due to our efficient model design, the probability of a sample (θ,ϕ) can easily be computed by its inverse $\mathcal{F}^{-1}$. Note that, as discussed in [MMR*19], this loss effectively minimizes the KL-divergence of target and encoded distributions.

During training, we observe that most of the computation time is used to generate the ground truth batches of samples $\mathcal{B}$, which is the very process that we are interested in improving with $\mathcal{F}$. We also notice that achieving a training procedure that works for every environment map is challenging due to gradient instabilities. We introduce a series of modifications to the training procedure to allow for stable training dynamics and avoid NANs for any input environment map, which we specify in Section 6.

5. Environment Map Compression

In addition to learning to sample and evaluate the PDF, we train a compression neural network $\mathcal{C}$ which learns to map from directions to RGB values: $\mathcal{C}(\theta,\phi) = RGB, \theta \in [0,\pi), \phi \in [0,2\pi), RGB \in \mathbb{R}^{+3}$. This serves two purposes: First, it provides a memory and time efficient representation of the environment map, thus further improving the render times and computational cost. Second, it transforms the tabulated representation into a continuous and differentiable function for which gradients can be computed, which may prove useful for inverse rendering scenarios. For the model design of $\mathcal{C}$, we build upon previous work on implicit neural representations. In particular, neural network architectures that work on the frequency domain have shown increased performance with respect to MLP for modelling natural signals [TSM*20, SMB*20], both in reconstruction quality and parameter efficiency. We thus model $\mathcal{C}$ as a shallow *sinusoidal MLP*, a *SIREN* [SMB*20]. This architectural design has been explored in recent work on environment map approximation, such as *RENI* [GES22]. However, as we do not aim to learn any prior over natural illumination, we can remove some constraints introduced in *RENI*, most notably the *rotation equivariance* requirement. Further, we do not use any latent vector to condition the output of our model, nor we require *Vector Neurons* for our representation. With our simplifications, we achieve much higher reconstruction quality with the same parameter count. We train $\mathcal{C}$ on linear RGB space, with a pixel-wise reconstruction loss. Specifically, we use an $\mathcal{L}_1$ loss, as it produces sharper and more accurate reconstructions than higher order norms, such as $\mathcal{L}_2$ [IZZE17, RPG21]. We also observe strong gradient instabilities when training with $\mathcal{L}_2$ on linear RGB. As $\mathcal{C}$ expects inputs in radians, but the input image has ranges of $H \times W$ pixels, our full loss becomes:

$$\mathcal{L}_{recon} = \frac{1}{H \times W}\sum_{i}^{H}\sum_{j}^{W} \|\mathcal{C}(\frac{i \times \pi}{H}, \frac{j \times 2\pi}{W}), \mathcal{I}(i,j)\|_1 \tag{6}$$

We use mini-batching to train our compression models $\mathcal{C}$, using uniformly sampled directions θ,ϕ. Previous work [GES22, SRRW21, RPDEPHG23, LBFS21] introduce adaptive sampling, cosine weighting or specular peak attenuation to similar loss functions for environment map or BRDF reconstruction. However, for our particular problem, we empirically observed that these additions tend to contribute negatively either to the training dynamics or to the final quality of the reconstruction.

6. Implementation Details

Preprocessing Before the rotation algorithm described In Section 3.1, we resize every input HDRi environment map to a resolution of $(2000, 4000)$ pixels using area interpolation. This is an optional step, but it helps us standardize our experiments so that input resolution does not change training times. Note that evaluation

times of our NEnv models depend on the model sizes and parameterizations, not on the resolution they were trained on.

Model Design and Training Our models sizes, loss functions, coupling layer design, normalization, optimizer and training configurations were selected using a combination of manual tuning and Bayesian hyperparameter optimization using *Weights and Biases* [Bie20] for a variety of representative environment maps.

We train the models using PyTorch [PGM*19] 1.11 and Torchvision [MR10]. Our flow and compression networks are trained independently, and individually for each input environment map. We empirically observe that mixed precision training [MNA*18] introduces strong training instabilities for this problem, so we train the models using *float32* precision. However, our models can be evaluated using half precision, which significantly increases their efficiency during test time. We provide evaluation code with this submission, training code will be provided upon acceptance. $\mathcal{F}$ and $\mathcal{C}$ each use around 3 MBs each, while a (2000, 4000) resolution HDRi environment map uses around 25 MBs.

Normalizing Flow Our normalizing flows $\mathcal{F}$ build upon the official implementation in [DBMP19a]. We use Batch Normalization [IS15] in the residual layers of our flows. To maximize evaluation efficiency, our flows have a limited number of trainable parameters. In particular, we use only 2 coupling layers, with 2 hidden layers, each with 256 hidden units and 256 *bins*. We use spline flows as our coupling layer type [DBMP19a]. Our initial learning rate is of $5e-4$, which is halved every 2500 iterations. We use Adam [KB14] as our optimizer. To stabilize training, we use gradient norm clipping [ZHSJ20] ($maxnorm = 1$). This is crucial for well-behaved models and efficient convergence. We train our models with a batch size of 100000 for 15000 epochs, which takes around 2 hours on an Nvidia RTX 3060 GPU. Note that fewer iterations (eg 5000) are typically sufficient for adequate results, while longer training helps to resolve additional details. These training times are comparable to previous work on neural reflectance encoding [Kuz21, ZRW*23, GES22].

Compression Network We use a small sinusoidal MLP as our environment map compression network $\mathcal{C}$. We use the official implementation in [SMB*20]. We train our models with a batch size of 500000 of randomly sampled directions and linear RGB values, for 10000 epochs, which takes around 0.5 hours on an Nvidia RTX 3060 GPU. Our initial learning rate is of $5e-4$, which is halved every 2000 iterations. We use Adam [KB14] as our optimizer. To stabilize training, we use gradient norm clipping ($maxnorm = 50$). Our compression networks have 3 layers with 512 hidden units each. We initialize their weights as described in [SMB*20].

Rendering We use our own path tracer (PT) rendering pipeline as baseline engine. This PT uses an implementation inspired by *Wavefront* [VA11, LKA13], powered by *Embree* [WWB*14]. We distribute render work in tiles of (25,25) pixels that perform path–tracing in usual Wavefront steps (ray generation, intersection, shading, connection), each tile using a separate CPU thread. Note that tile size could be changed, as our selected tile dimensions are fixed due to hardware limitations and amount of currently thread active simultaneously (Intel Core i7-7700K CPU @ 4.20GHz, 8 threads).

Under this architecture, we perform our environment map sampling step once per tile, before every pixel in each tile performs its connection step. This way, we can either use a CPU approach [CJAMJ] per pixel or integrate a single call to $\mathcal{F}$ to obtain all samples that are required in the current tile. We use *LibTorch* to use our PyTorch implementation from our C++ code. Intersections are performed in a single call using a ray packet call from Embree. Integrating $\mathcal{C}$ is more challenging, since we require ray data. However, we can store all this information simultaneously: We prepare all rays for intersection, and perform a single call to $\mathcal{C}$. This way, we obtain all values at once. Note that these evaluations will be discarded if the next path–tracing bounce does not go out of bounds.

7. Dataset

Our goal is to propose a method that works for any type of global illumination which can be represented using environment maps. To this end, we gather a dataset of 32 high resolution HDRi environment maps from publicly available sources, notably HDRMaps and Poly Haven, to thoroughly evaluate our models. Our dataset contains natural day light, including sunny midday illumination, cloudy diffuse images, sunset and dusk. Further, we also test our method on indoor scenes, from studio lighting, to concert hall or cathedral illuminations. We purposefully include very challenging cases with multiple colored light sources, which helps understand the limitations and capabilities of previous work and our models. We will make this dataset, our trained models and an interactive website for visualization, publicly available upon acceptance.

8. Evaluation

In this section, we evaluate our models, both quantitatively and qualitatively, We first measure their performance in terms of reconstructing the probability density and the RGB values of the input environment maps, and in computational cost. Finally, we measure their integrated quality in final rendered images compared to a variety of baselines. Unless stated otherwise, we use the full dataset described in Section 7 for our analyses.

To make comparisons fair, we use the following configuration: For RENI [GES22], we train a new model from scratch with the exact same architecture design and trainable parameters as our $\mathcal{C}$, and train it for 3000 epochs. For Spherical Harmonics (SH) [See66, RH01], we use 1024 coefficients, and for Spherical Gaussians (SG) [XSD*13], 16×32 dimensions. These configurations have a larger amount of parameters than what is typically used for these methods, especially in real time rendering. However, our goal is to maximize the reconstruction quality for each method and environment map, even if it results in an increased memory footprint. For the three methods we evaluate, the output resolution is set to (256, 512) pixels. We follow the official implementation of [GES22] for all these comparisons.

8.1. PDF Fit Accuracy

To validate our design choices for the coupling layers of our normalizing flow $\mathcal{F}$, we train different versions of $\mathcal{F}$ for every environ-

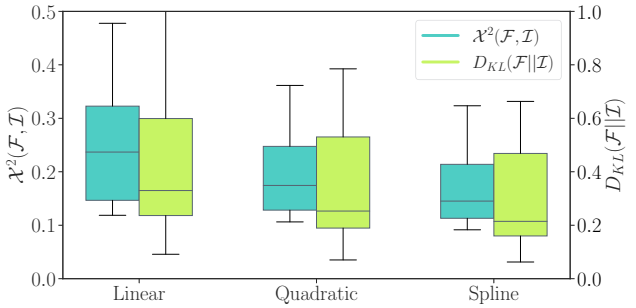

Figure 4: *PDF $\mathcal{X}^2$ and KL Divergence [KL51] of normalizing flows using different coupling layer types, including Linear and Quadratic Couplings from [MMR*19] and Spline Layers [DBMP19a].*

ment map in our dataset, exclusively changing the type of polynomial function in the coupling layer. We measure the $\mathcal{X}^2$ distance, as well as the KL divergence $\mathcal{D}_{KL}(\mathcal{F}||\mathcal{I})$ [KL51] between the probabilities encoded by our normalizing flow $p_{\mathcal{F}}$ and the ground truth counterparts $p_{\mathcal{I}}$ as follows:

$$\mathcal{D}_{KL}(\mathcal{F}||\mathcal{I}) = \sum_{\theta_i=0}^{\pi}\sum_{\phi_j=0}^{2\pi} p_{\mathcal{F}}(\theta_i,\phi_j) log\left(\frac{p_{\mathcal{F}}(\theta_i,\phi_j)}{p_{\mathcal{I}}(\theta_i,\phi_j)}\right) \quad (7)$$

This metric, also used in [MMR*19], measures the distance between the learned probability distribution and the real PDF encoded in the environment map. In Figure 4, we show the aggregated divergences for our dataset for three different coupling layer types. As shown, the *Piecewise Linear* coupling proposed in [MMR*19] is outperformed by its more expressive *Piecewise Quadratic* variation, while the *Rational-Quadratic Spline Flows* proposed in [DBMP19a] achieve the lowest divergence overall, albeit by a relatively small margin. We also provide a qualitative evaluation in Figure 5, where we show that linear and quadratic coupling layers encode less detailed probability distributions. These differences are particularly relevant for inputs with multiple distant light sources.

We also evaluate whether a larger model can yield significant improvements in accuracy. In Figure 6, we show the results of a large flow ($\mathcal{F}$-XL), with twice as many layers and neurons per layer, a total of approximately 7 times more trainable parameters and model size in memory. Even though it indeed reduces the divergence from $\mathcal{D}_{KL}(\mathcal{F}||\mathcal{I}) = 0.308$ to $\mathcal{D}_{KL}(\mathcal{F}\text{-XL}||\mathcal{I}) = 0.281$, this model takes approximately one day to train and makes sampling 3.43 times slower. At low *spp*, the larger model only achieves a small gain in render accuracy (from 0.059 to 0.048 in FLIP [ANAM*20]), suggesting that larger models exhibit diminishing returns in terms of quality. Given enough samples, both models converge to the same image. We believe our chosen model sizes adequately balance accuracy and speed. However, for practitioners that would like better accuracy at the cost of computational performance, we suggest using more coupling layers and more bins, as they tend to be the most effective design decisions beyond the coupling type.

Method	SH [RH01]	SG [WRG*09]	RENI [GES22]	NEnv $\mathcal{C}$ (Ours)
PSNR ↑	**13.68**±4.88	20.54±5.04	18.92±3.24	**30.65**±5.59
SSIM [WBSS04] ↑	**0.276**±0.20	0.559±0.16	0.557±0.16	**0.873**±0.08
LPIPS [ZIE*18] ↓	**0.687**±0.07	0.637±0.12	0.657±0.12	**0.155**±0.08
FLIP [ANAM*20] ↓	**0.505**±0.21	0.248±0.11	0.278±0.09	**0.062**±0.03

Table 1: *Average (± std.) environment map reconstruction error for different methods, with pixel-wise and perceptual metrics. We use a color code to highlight **best** and **worst** cases.*

8.2. Environment Map Reconstruction

We now aim to measure the accuracy of the environment maps encoded by $\mathcal{C}$ and previous work on this problem. In Table 1, we show the average error on a variety of metrics for every environment map in our dataset. We use pixel-wise (PSNR) and perceptual metrics (SSIM [WBSS04], LPIPS-VGG [ZIE*18], FLIP [ANAM*20]) to thoroughly understand the differences between each method. We compute these metrics using *PIQ* [KZPD22] and the implementation in [ANAM*20], on images of (512, 256) pixels. We apply a $\gamma = 2.2$ to tonemap them before computing the distances, and clip values larger than 1. As shown, our network $\mathcal{C}$ clearly outperforms every other method, with SH struggling significantly, while RENI and SG achieve comparably better performances. We also provide a qualitative comparison in Figure 7, for a subset of our dataset, where it can be seen that $\mathcal{C}$ provides high-quality reconstructions. RENI is optimized for outdoor, natural lighting, and it struggles with indoor illumination, providing overly smooth outputs. SG provides sharper results, while SH yields visually unpleasant results for every case: it is well known that the cyclic nature of SH tends to produce ringing artifacts, especially for high-contrast illumination environments and a large number of coefficients. We study the impact of these reconstructions on final renders in Section 8.4. Our representation $\mathcal{C}$ has the additional advantage that, thanks to its reconstruction quality, it can be used directly as background for samples which do not hit any geometry on the scene, while other alternatives require additional storage for higher quality versions, or even the original image.

8.3. Computational Cost

In Figure 8, we show the average time per sample obtained by the previous work and our normalizing flow $\mathcal{F}$. On average, *Hierarchical Warping* [CJAMJ, Pha19] obtains $6.48e^{-2} \pm 1.84e^{-3}$ milliseconds (ms) per sample, with the *Piecewise–Constant* method [PJH16] achieving $1.12e^{-2} \pm 3.24e^{-4}$ ms, while $\mathcal{F}$ obtains $6.83e^{-4} \pm 3.69e^{-6}$. To make these comparison more favorable to each method, the timings for *Hierarchical Warping* [CJAMJ, Pha19] and *Piecewise–Constant* [PJH20] are measured on CPU, since both algorithms rely on binary search which is difficult to parallelize on GPU, while the timings for NEnv are measured on GPU. Note that binary search is still needed to find the right bins in $\mathcal{F}$, but we use a very limited amount of such bins (256), which results in large computational improvements compared to analytical methods on high resolution (4K) input maps. Additional speed up is achieved through GPU parallel batching, and due to the very limited depth and width of the proposed flows, making them very efficient during inference. For $\mathcal{F}$, we use a sampling batch size

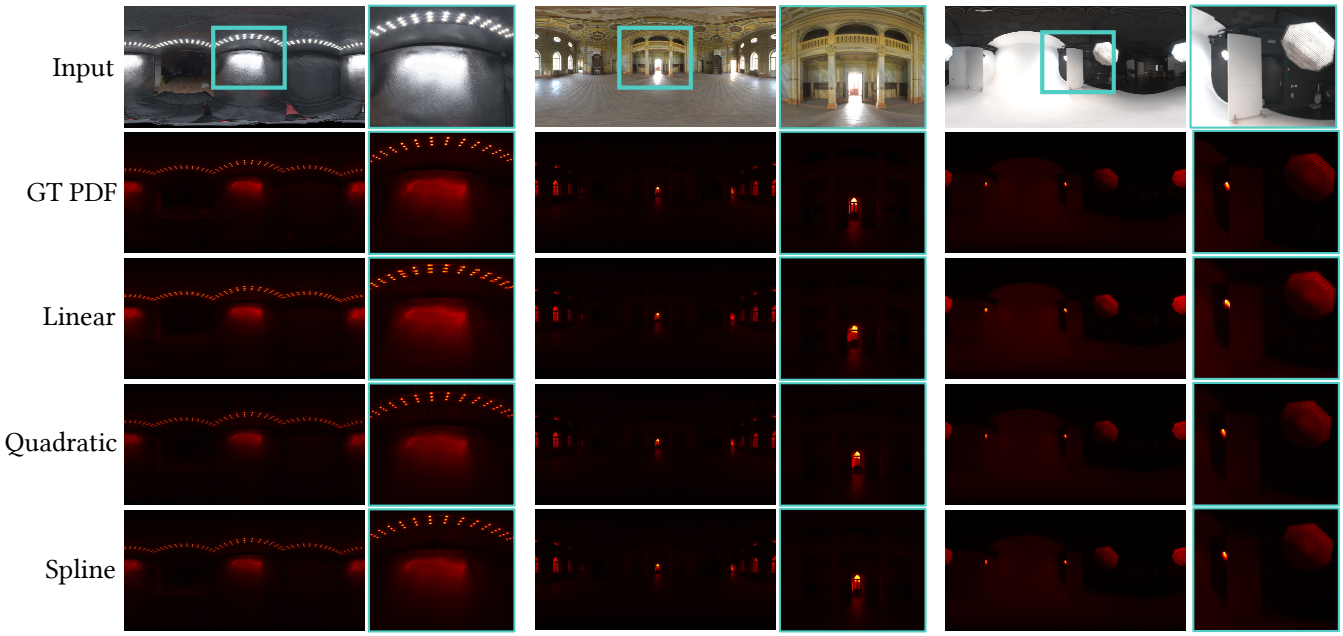

Figure 5: *A qualitative comparison between the type of coupling layer used in our normalizing flows. On the top two rows, we show the input environment maps and their ground truth PDFs. Using the linear and quadratic couplings defined in [MMR*19] we achieve somewhat accurate encodings. With spline flows [DBMP19a], we achieve sharper and more accurate probability distributions. We use a $\gamma = 2.2$ to tonemap the RGB and PDF maps to help visualization. We highlight relevant regions using blue insets. Best viewed in color on a screen.*

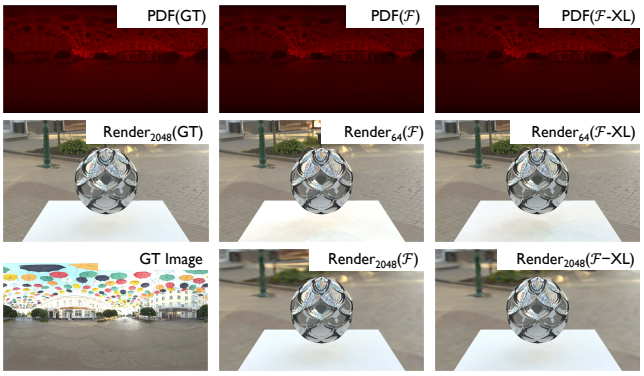

Figure 6: *A visualization of the impact of the size of $\mathcal{F}$. NEnv-XL is a larger version of $\mathcal{F}$, with twice as many layers, bins and neurons per layer. This yields on small improvements on render and reconstruction quality, at the cost of sampling efficiency. We show renders$_{spp}$ at different samples per pixel.*

of 200000, which is the maximal size we can fit in our GPU, and we also include the times required to upload and download the samples from GPU, as in a real rendering scenario. We measure these times on the same hardware to make results comparable.

$\mathcal{F}$ achieves, on average, 16.83 times more samples per second than the *Piecewise-Constant* method and 94.85 than *Hierarchical Warping*, with a maximum difference of 106.06. Furthermore, the timings for the *Piecewise-Constant* and *Hierarchical Warping* methods vary up to 15% and 12.5% respectively depending on the content of the environment map, while the variance of $\mathcal{F}$ is negligible. Not only we achieve up to two orders of magnitude faster sampling than previous work, our method also significantly more consistent in terms of timings. We found that *Hierarchical Warping* and the *Piecewise-Constant* methods typically struggle more with environment maps with multiple, distant light sources, while they are more efficient for natural daylight illumination. In terms of RGB evaluation, a tabulated version of the environment map requires an average of $8.43e^{-4} \pm 1.44e^{-5}$ ms per direction, while our model $\mathcal{C}$ achieves 2.97 times faster evaluations, at an average of $2.83e^{-4} \pm 7.85e^{-6}$ ms. Note that we use *TorchScript* with *Pytorch 1.11* to measure the times for $\mathcal{C}$ and $\mathcal{F}$. Newer versions of these libraries, which allow for compiled models, will likely help achieve faster evaluations without changing the model architectures. Globally, across our dataset, we achieve an average render time reduction of $5.8 \pm 4.8\%$, with a maximum of 15.3%.

8.4. Rendering Comparisons

The goal of NEnv is to generate accurate yet fast environment map representations that do not make compromises in terms of rendering quality. We use the *Piecewise-Constant* PDFs as our ground truth (GT) and perform qualitative and quantitative analyses to show the reconstruction quality of NEnv with respect to previous work. We provide a fine grained analysis of NEnv, in which we evaluate each of the models separately and a full version of the model which uses both $\mathcal{F}$ and $\mathcal{C}$.

We use our path tracing engine configured as explained in Sec-

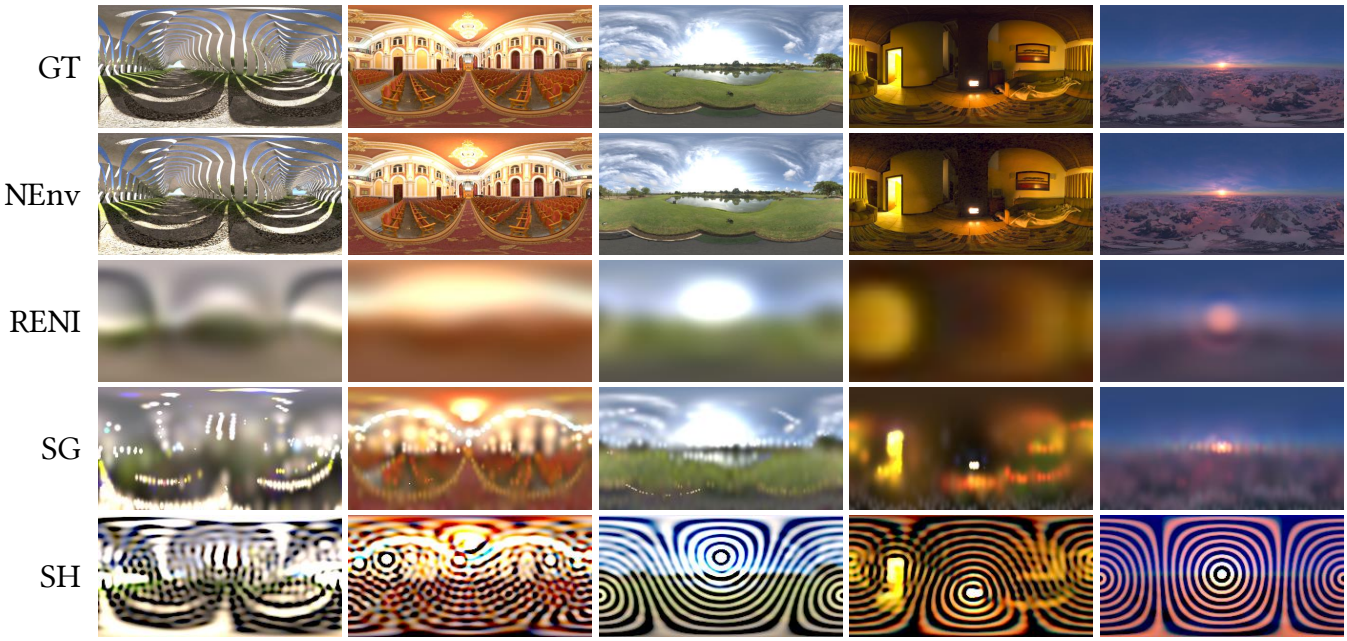

Figure 7: *Environment map encoded by our method and previous work, including RENI [GES22], Spherical Gaussian (SG) [WRG*09] and Spherical Harmonics (SH) [RH01]. We use a $\gamma = 2.2$ to tonemap the RGB maps to help visualization.*

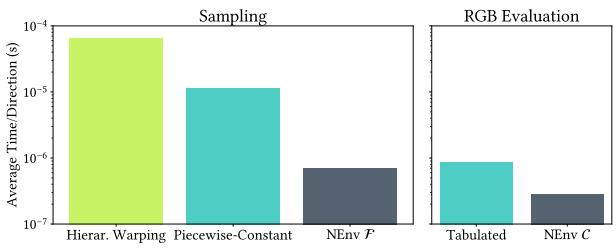

Figure 8: *On the left, average seconds per sample for the baseline sampling algorithms and our model $\mathcal{F}$. On the right, evaluation times for a tabulated baseline and our $\mathcal{C}$. Note that we use a logarithmic scale for this plot.*

tion 6 to render the same scene for each environment map. Each scene is rendered using a simple pinhole camera (no Depth of Field) and the environment maps as the only source of light, so we can reduce as much as possible externals sources of Monte Carlo noise at low sampling. We use MIS, so both sampling and PDF functions for our maps impact the resulting image. All images are rendered with a resolution of (1080, 1920) pixels, 32 samples, maximum depth of 16 and path throughput weight Russian Roulette [PJH16].

In Figure 9, we show a qualitative comparison between different methods on the same scene, illuminated with different environment maps. We use an object from [SHHD17], which helps in highlighting the differences between far field illumination representations. As shown, our method, in any of their configurations, generates images which closely match the ground truth, while previous work struggles significantly, particularly on indoor illumination. We include more scenes on the supplementary video.

	PSNR ↑	SSIM [WBSS04] ↑	LPIPS [ZIE*18] ↓	FLIP [ANAM*20] ↓
SH [RH01]	21.77±5.35	**0.933**±0.05	**0.160**±0.03	0.150±0.06
SG [WRG*09]	21.62±6.34	0.940±0.05	0.153±0.03	**0.154**±0.07
RENI [GES22]	**21.53**±5.62	0.936±0.04	0.159±0.03	0.151±0.05
NEnv ($\mathcal{F}$)	37.97±3.73	0.996±0.00	0.024±0.01	**0.032**±0.02
NEnv ($\mathcal{C}$)	**41.05**±8.89	**0.997**±0.01	**0.016**±0.01	0.036±0.02
NEnv (Both)	35.16±4.97	0.994±0.01	0.033±0.01	0.049±0.02

Table 2: *Average (± std.) render reconstruction error for different methods, with pixel-wise and perceptual metrics. We use a color code to highlight **best** and **worst** cases.*

Finally, in Table 2, we show a quantitative comparison of the average error obtained by these methods, for our entire dataset. For this analysis, we only measure the error on the object and floor on the renders in Figure 9, masking out the background. We observe that, even when combining our sampling and compression networks, our reconstruction quality is much higher than any of the methods in the previous work. Interestingly, the differences in environment map reconstruction quality we measured in Table 1, which were very unfavorable to SH, are not exactly correlated to rendering error, suggesting that SH, while being a pixel-wise inaccurate image representation, it tends to be precise on the more relevant information of the image, *i.e.*, where the key light sources lie. In terms of rendering, neither of the previous methods that we tested behaved consistently better than any other across every metric.

9. Conclusions

In this work, we have presented a neural rendering method for joint compression, evaluation and sampling of environment maps for global illumination. We have validated our models quantitatively using a diverse dataset of environment maps. Our proposed lightweight normalizing flows provides accurate yet efficient

Figure 9: *Render comparisons of different configurations of our work with the ground truth environment map and previous work, rendered at 2048* spp. *Best viewed in color on a screen. Please zoom in for details.*

sampling and pdf evaluation, achieving significantly faster computational times than analytical approaches with negligible loss in quality. Further, our sinusoidal compression networks achieve high-quality environment map approximations, surpassing previous work in both quality and generality. Every component in our models is differentiable, and they can be seamlessly integrated in engines like Mitsuba [NDVZJ19,JSR*22] or PRDPT [FR22] for inverse rendering applications. All in all, NEnv can represent global illumination with higher generality, quality and computational efficiency than previous work, with fully differentiable components. To enhance reproducibility and encourage the use of our neural illumination framework, we will open source our code for training and evaluation, and provide a dataset of trained *NEnv* models for a variety of publicly available environment maps.

Our work could be extended in several ways. We require two networks, one for sampling and pdf evaluation, and another for compression. While this dual approach allows us to maximize their individual efficiency, a future research avenue is to build a single model which can solve the three tasks at the same time, to simplify our representation and possibly help with parameter reuse across models. However, fusing a sinusoidal MLP with a spline-based normalizing flow is not straightforward and requires a significant research effort which lies outside of the scope of this work. Recent work on architectural design [MGB*21] may provide cues on how this could be achieved. Further, we require to train new models for each input environment map. Building upon recent work [GES22, RBRD22, GMX22, SRRW21, HGC*20], we could build a prior over environment maps, which should help reduce training times and solve inverse illumination problems, as in [LSR*20, ALKN19, HHM22, YS19, WPFK21, YS21, SMT*20, GHGS*19, WSG*23]. Achieving this without losing computational efficiency is a challenging future research problem. Besides, recent work on neural rendering [MESK22, CXG*22, WZL*22, AHZ*22, LPL*22] has shown that with carefully designed CUDA kernels or architectural representations, it is possible to achieve more efficient training and evaluation times. Integrating these ideas into our framework could help further increase its computational efficiency.

Finally, we believe that the ideas we present in this work can be used for other path tracing and physically based rendering problems, like aggregate scattering [BNH*16] or BxDF representa-

tions [SRRW21, CNN22, ZLW*21, CLZ*20]. We hope our work inspires future work on invertible neural generative models for physically-based and neural rendering.

Acknowledgments We would like to thank Luis Romero for his help designing the scenes we rendered. Elena Garces was partially supported by a Juan de la Cierva - Incorporacion Fellowship (IJC2020-044192-I). This publication is part of the project V+Real, PID2021-122392OB-I00 funded by MCIN/AEI/10.13039/501100011033/FEDER, UE.

References

[AHZ*22] ATTAL B., HUANG J.-B., ZOLLHÖFER M., KOPF J., KIM C.: Learning neural light fields with ray-space embedding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 19819–19829. 3, 10

[ALKN19] AZINOVIC D., LI T.-M., KAPLANYAN A., NIESSNER M.: Inverse path tracing for joint material and lighting estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 2447–2456. 10

[ANAM*20] ANDERSSON P., NILSSON J., AKENINE-MÖLLER T., OSKARSSON M., ÅSTRÖM K., FAIRCHILD M. D.: Flip: A difference evaluator for alternating images. *Proc. ACM Comput. Graph. Interact. Tech. 3*, 2 (2020), 15–1. 7, 9

[ARBJ03] AGARWAL S., RAMAMOORTHI R., BELONGIE S., JENSEN H. W.: Structured importance sampling of environment maps. *ACM Transactions on Graphics (TOG) 22*, 3 (2003), 605–612. 2

[BGP*22] BAATZ H., GRANSKOG J., PAPAS M., ROUSSELLE F., NOVÁK J.: Nerf-tex: Neural reflectance field textures. In *Computer Graphics Forum* (2022), vol. 41, Wiley Online Library, pp. 287–301. 3

[Bie20] BIEWALD L.: Experiment tracking with weights and biases, 2020. Software available from wandb.com. 6

[BMT*21] BARRON J. T., MILDENHALL B., TANCIK M., HEDMAN P., MARTIN-BRUALLA R., SRINIVASAN P. P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021). 3

[BNH*16] BLUMER A., NOVÁK J., HABEL R., NOWROUZEZAHRAI D., JAROSZ W.: Reduced aggregate scattering operators for path tracing. *Computer Graphics Forum (Proceedings of Pacific Graphics) 35*, 7 (2016), 461–473. 10

[BWP*20] BITTERLI B., WYMAN C., PHARR M., SHIRLEY P., LEFOHN A., JAROSZ W.: Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 39*, 4 (2020). 2

[CJAMJ] CLARBERG P., JAROSZ W., AKENINE-MÖLLER T., JENSEN H. W.: Wavelet importance sampling: Efficiently evaluating products of complex functions. 2, 3, 6, 7

[CLZ*20] CHE C., LUAN F., ZHAO S., BALA K., GKIOULEKAS I.: Towards learning-based inverse subsurface scattering. In *2020 IEEE International Conference on Computational Photography (ICCP)* (2020), IEEE, pp. 1–12. 11

[CNN22] CHEN Z., NOBUHARA S., NISHINO K.: Invertible neural brdf for object inverse rendering. *IEEE Transactions on Pattern Analysis and Machine Intelligence 44*, 12 (2022), 9380–9395. 2, 11

[CXG*22] CHEN A., XU Z., GEIGER A., YU J., SU H.: Tensorf: Tensorial radiance fields. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXII* (2022), Springer, pp. 333–350. 10

[DBMP19a] DURKAN C., BEKASOV A., MURRAY I., PAPAMAKARIOS G.: Neural spline flows. In *Advances in Neural Information Processing Systems* (2019), vol. 32. 2, 5, 6, 7, 8

[DBMP19b] DURKAN C., BEKASOVS A., MURRAY I., PAPAMAKARIOS G.: Cubic-spline flows. In *Workshop on Invertible Neural Nets and Normalizing Flows: ICML 2019* (2019). 5

[DKB14] DINH L., KRUEGER D., BENGIO Y.: Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516* (2014). 5

[DSDB16] DINH L., SOHL-DICKSTEIN J., BENGIO S.: Density estimation using real nvp. *arXiv preprint arXiv:1605.08803* (2016). 5

[FKYT*22] FRIDOVICH-KEIL S., YU A., TANCIK M., CHEN Q., RECHT B., KANAZAWA A.: Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 5501–5510. 3

[FR22] FISCHER M., RITSCHEL T.: Plateau-reduced differentiable path tracing. *arXiv preprint arXiv:2211.17263* (2022). 10

[FWH*22] FAN J., WANG B., HAŠAN M., YANG J., YAN L.-Q.: Neural layered brdfs. In *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* (2022). 3

[GCS*20] GROVER A., CHUTE C., SHU R., CAO Z., ERMON S.: Alignflow: Cycle consistent learning from multiple domains via normalizing flows. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2020), vol. 34, pp. 4028–4035. 2

[GES22] GARDNER J., EGGER B., SMITH W. A. P.: Rotation-equivariant conditional spherical neural fields for learning a natural illumination prior. In *Advances in Neural Information Processing Systems* (2022). 3, 5, 6, 7, 9, 10

[GHGS*19] GARDNER M.-A., HOLD-GEOFFROY Y., SUNKAVALLI K., GAGNÉ C., LALONDE J.-F.: Deep parametric indoor lighting estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 7175–7183. 2, 10

[GMX22] GAO D., MU H., XU K.: Neural global illumination: Interactive indirect illumination prediction under dynamic area lights. *IEEE Transactions on Visualization and Computer Graphics* (2022). 3, 10

[GSKH21] GAO C., SARAF A., KOPF J., HUANG J.-B.: Dynamic view synthesis from dynamic monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 5712–5721. 3

[HDGS20] HELMINGER L., DJELOUAH A., GROSS M., SCHROERS C.: Lossy image compression with normalizing flows. *arXiv preprint arXiv:2008.10486* (2020). 2

[Hei20] HEITZ E.: Can't invert the cdf? the triangle-cut parameterization of the region under the curve. *Computer Graphics Forum 39*, 4 (2020), 121–132. 2

[HGC*20] HU B., GUO J., CHEN Y., LI M., GUO Y.: Deepbrdf: A deep representation for manipulating measured brdf. In *Computer Graphics Forum* (2020), vol. 39, Wiley Online Library, pp. 157–166. 10

[HHM22] HASSELGREN J., HOFMANN N., MUNKBERG J.: Shape, light & material decomposition from images using monte carlo rendering and denoising. *arXiv preprint arXiv:2206.03380* (2022). 10

[HKLC18] HUANG C.-W., KRUEGER D., LACOSTE A., COURVILLE A.: Neural autoregressive flows. In *International Conference on Machine Learning* (2018), PMLR, pp. 2078–2087. 5

[IS15] IOFFE S., SZEGEDY C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning* (2015), pmlr, pp. 448–456. 6

[IZZE17] ISOLA P., ZHU J.-Y., ZHOU T., EFROS A. A.: Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2017), pp. 1125–1134. 5

[JCJ09] JAROSZ W., CARR N. A., JENSEN H. W.: Importance sampling spherical harmonics. *Computer Graphics Forum (Proceedings of Eurographics) 28*, 2 (2009), 577–586. 2

[JSR*22] JAKOB W., SPEIERER S., ROUSSEL N., NIMIER-DAVID M., VICINI D., ZELTNER T., NICOLET B., CRESPO M., LEROY V., ZHANG Z.: Mitsuba 3 renderer, 2022. https://mitsuba-renderer.org. 3, 10

[KAL*21] KARRAS T., AITTALA M., LAINE S., HÄRKÖNEN E., HELLSTEN J., LEHTINEN J., AILA T.: Alias-free generative adversarial networks, 2021. 3

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

[KBE*19] KUMAR M., BABAEIZADEH M., ERHAN D., FINN C., LEVINE S., DINH L., KINGMA D.: Videoflow: A flow-based generative model for video. *arXiv preprint arXiv:1903.01434 2*, 5 (2019), 3. 2

[KBH03] KAINZ F., BOGART R., HESS D.: The openexr image file format. *ACM SIGGRAPH Technical Sketches* (2003). 2

[KD18] KINGMA D. P., DHARIWAL P.: Glow: Generative flow with invertible 1x1 convolutions. *Advances in Neural Information Processing Systems 31* (2018). 2, 5

[KL51] KULLBACK S., LEIBLER R. A.: On information and sufficiency. *The annals of mathematical statistics 22*, 1 (1951), 79–86. 7

[KMLH21] KHEMAKHEM I., MONTI R., LEECH R., HYVARINEN A.: Causal autoregressive flows. In *International Conference on Artificial Intelligence and Statistics* (2021), PMLR, pp. 3520–3528. 5

[KPB20] KOBYZEV I., PRINCE S. J., BRUBAKER M. A.: Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence 43*, 11 (2020), 3964–3979. 5

[KSJ*16] KINGMA D. P., SALIMANS T., JOZEFOWICZ R., CHEN X., SUTSKEVER I., WELLING M.: Improved variational inference with inverse autoregressive flow. *Advances in Neural Information Processing Systems 29* (2016). 5

[Kuz21] KUZNETSOV A.: Neumip: Multi-resolution neural materials. *ACM Transactions on Graphics (TOG) 40*, 4 (2021). 3, 6

[KVHS00] KAUTZ J., VÁZQUEZ P.-P., HEIDRICH W., SEIDEL H.-P.: Unified approach to prefiltered environment maps. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000* (2000), p. 185–196. 2

[KWM*22] KUZNETSOV A., WANG X., MULLIA K., LUAN F., XU Z., HASAN M., RAMAMOORTHI R.: Rendering neural materials on curved surfaces. In *ACM SIGGRAPH 2022 Conference Proceedings* (2022), pp. 1–9. 3

[KZPD22] KASTRYULIN S., ZAKIROV J., PROKOPENKO D., DYLOV D. V.: Pytorch image quality: Metrics for image quality assessment, 2022. 7

[LBFS21] LAVOUÉ G., BONNEEL N., FARRUGIA J.-P., SOLER C.: Perceptual quality of brdf approximations: dataset and metrics. In *Computer Graphics Forum* (2021), vol. 40, Wiley Online Library, pp. 327–338. 5

[LDVGT20] LUGMAYR A., DANELLJAN M., VAN GOOL L., TIMOFTE R.: Srflow: Learning the super-resolution space with normalizing flow. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16* (2020), Springer, pp. 715–732. 2

[LKA13] LAINE S., KARRAS T., AILA T.: Megakernels considered harmful: Wavefront path tracing on gpus. In *Proceedings of the 5th High-Performance Graphics Conference* (2013), pp. 137–143. 6

[LPL*22] LIU Y., PENG S., LIU L., WANG Q., WANG P., THEOBALT C., ZHOU X., WANG W.: Neural rays for occlusion-aware image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 7824–7833. 10

[LS98] LARSON G. W., SHAKESPEARE R.: *Rendering with Radiance: The Art and Science of Lighting Visualization*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998. 2

[LSR*20] LI Z., SHAFIEI M., RAMAMOORTHI R., SUNKAVALLI K., CHANDRAKER M.: Inverse rendering for complex indoor scenes: Shape, spatially-varying lighting and svbrdf from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 2475–2484. 10

[MESK22] MÜLLER T., EVANS A., SCHIED C., KELLER A.: Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (TOG) 41*, 4 (2022), 102:1–102:15. 10

[MGB*21] MEHTA I., GHARBI M., BARNES C., SHECHTMAN E., RAMAMOORTHI R., CHANDRAKER M.: Modulated periodic activations for generalizable local functional representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 14214–14223. 3, 10

[MMR*19] MÜLLER T., MCWILLIAMS B., ROUSSELLE F., GROSS M., NOVÁK J.: Neural importance sampling. *ACM Transactions on Graphics (TOG) 38*, 5 (2019), 1–19. 2, 5, 7, 8

[MNA*18] MICIKEVICIUS P., NARANG S., ALBEN J., DIAMOS G., ELSEN E., GARCIA D., GINSBURG B., HOUSTON M., KUCHAIEV O., VENKATESH G., ET AL.: Mixed precision training. In *International Conference on Learning Representations* (2018). 6

[MR10] MARCEL S., RODRIGUEZ Y.: Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM International Conference on Multimedia* (2010), pp. 1485–1488. 6

[MRKN20] MÜLLER T., ROUSSELLE F., KELLER A., NOVÁK J.: Neural control variates. *ACM Transactions on Graphics (TOG) 39*, 6 (2020), 1–19. 2, 5

[MST*21] MILDENHALL B., SRINIVASAN P. P., TANCIK M., BARRON J. T., RAMAMOORTHI R., NG R.: Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM 65*, 1 (2021), 99–106. 3

[NDVZJ19] NIMIER-DAVID M., VICINI D., ZELTNER T., JAKOB W.: Mitsuba 2: A retargetable forward and inverse renderer. *ACM Transactions on Graphics (TOG) 38*, 6 (2019), 1–17. 10

[OLK*21] OUYANG Y., LIU S., KETTUNEN M., PHARR M., PANTALEONI J.: Restir gi: Path resampling for real-time path tracing. In *Computer Graphics Forum* (2021), vol. 40, pp. 17–29. 2

[PGM*19] PASZKE A., GROSS S., MASSA F., LERER A., BRADBURY J., CHANAN G., KILLEEN T., LIN Z., GIMELSHEIN N., ANTIGA L., ET AL.: Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems* (2019), vol. 32. 6

[Pha19] PHARR M.: Visualizing warping strategies for sampling environment map lights, 2019. URL: https://pharr.org/matt/blog/2019/06/05/visualizing-env-light-warpings. 3, 7

[PJH16] PHARR M., JAKOB W., HUMPHREYS G.: *Physically Based Rendering: From Theory to Implementation*, 3rd ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2016. 2, 3, 7, 9

[PJH20] PHARR M., JAKOB W., HUMPHREYS G.: Implementation of the forth- coming 4th edition of physically based rendering: From theory to implementation, 2020. https://github.com/mmp/pbrt-v4. 3, 7

[PNR*21] PAPAMAKARIOS G., NALISNICK E., REZENDE D. J., MOHAMED S., LAKSHMINARAYANAN B.: Normalizing flows for probabilistic modeling and inference. *The Journal of Machine Learning Research 22*, 1 (2021), 2617–2680. 5

[PPM17] PAPAMAKARIOS G., PAVLAKOU T., MURRAY I.: Masked autoregressive flow for density estimation. *Advances in Neural Information Processing Systems 30* (2017). 5

[PSM19] PAPAMAKARIOS G., STERRATT D., MURRAY I.: Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In *The 22nd International Conference on Artificial Intelligence and Statistics* (2019), PMLR, pp. 837–848. 5

[RBRD22] RAINER G., BOUSSEAU A., RITSCHEL T., DRETTAKIS G.: Neural precomputed radiance transfer. In *Computer Graphics Forum* (2022), vol. 41, Wiley Online Library, pp. 365–378. 10

[RCKP22] RHO D., CHO J., KO J. H., PARK E.: Neural residual flow fields for efficient video representations. In *Proceedings of the Asian Conference on Computer Vision* (2022), pp. 3447–3463. 3

[RH01] RAMAMOORTHI R., HANRAHAN P.: An efficient representation for irradiance environment maps. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (2001), pp. 497–500. 6, 7, 9

[RPDEPHG23] RODRIGUEZ-PARDO C., DOMINGUEZ-ELVIRA H., PASCUAL-HERNANDEZ D., GARCES E.: Umat: Uncertainty-aware single image high resolution material capture. *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2023). 5

[RPG21] RODRIGUEZ-PARDO C., GARCES E.: Neural photometry-guided visual attribute transfer. *IEEE Transactions on Visualization and Computer Graphics* (2021). 5

[RPR*20] REZENDE D. J., PAPAMAKARIOS G., RACANIERE S., ALBERGO M., KANWAR G., SHANAHAN P., CRANMER K.: Normalizing flows on tori and spheres. In *International Conference on Machine Learning* (2020), PMLR, pp. 8083–8092. 3

[See66] SEELEY R. T.: Spherical harmonics. *The American Mathematical Monthly 73*, 4P2 (1966), 115–121. 2, 6

[SFMP20] SELVAN R., FAYE F., MIDDLETON J., PAI A.: Uncertainty quantification in medical image segmentation with normalizing flows. In *Machine Learning in Medical Imaging: 11th International Workshop, MLMI 2020, Held in Conjunction with MICCAI 2020, Lima, Peru, October 4, 2020, Proceedings 11* (2020), Springer, pp. 80–90. 2

[SHHD17] SCHÜSSLER V., HEITZ E., HANIKA J., DACHSBACHER C.: Microfacet-based normal mapping for robust monte carlo path tracing. *ACM Transactions on Graphics (TOG) 36*, 6 (2017), 1–12. 9

[SK13] SIK M., KRIVANEK J.: Fast Random Sampling of Triangular Meshes. In *Pacific Graphics Short Papers* (2013), Levy B., Tong X., Yin K., (Eds.), The Eurographics Association. 2

[SMB*20] SITZMANN V., MARTEL J., BERGMAN A., LINDELL D., WETZSTEIN G.: Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems 33* (2020), 7462–7473. 3, 5, 6

[SMT*20] SRINIVASAN P. P., MILDENHALL B., TANCIK M., BARRON J. T., TUCKER R., SNAVELY N.: Lighthouse: Predicting lighting volumes for spatially-coherent illumination. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 8080–8089. 10

[SPY*22] STRÜMPLER Y., POSTELS J., YANG R., GOOL L. V., TOMBARI F.: Implicit neural representations for image compression. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVI* (2022), Springer, pp. 74–91. 3

[SRF*21] SITZMANN V., REZCHIKOV S., FREEMAN W. T., TENENBAUM J. B., DURAND F.: Light field networks: Neural scene representations with single-evaluation rendering. In *Advances in Neural Information Processing Systems* (2021). 3

[SRRW21] SZTRAJMAN A., RAINER G., RITSCHEL T., WEYRICH T.: Neural brdf representation and importance sampling. In *Computer Graphics Forum* (2021), vol. 40, Wiley Online Library, pp. 332–346. 2, 3, 5, 10, 11

[TCY*22] TANCIK M., CASSER V., YAN X., PRADHAN S., MILDENHALL B., SRINIVASAN P., BARRON J. T., KRETZSCHMAR H.: Block-NeRF: Scalable large scene neural view synthesis. In *arXiv* (2022). 3

[TLY*21] TAKIKAWA T., LITALIEN J., YIN K., KREIS K., LOOP C., NOWROUZEZAHRAI D., JACOBSON A., MCGUIRE M., FIDLER S.: Neural geometric level of detail: Real-time rendering with implicit 3D shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021). 3

[TSM*20] TANCIK M., SRINIVASAN P. P., MILDENHALL B., FRIDOVICH-KEIL S., RAGHAVAN N., SINGHAL U., RAMAMOORTHI R., BARRON J. T., NG R.: Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems* (2020). 3, 5

[TTM*22] TEWARI A., THIES J., MILDENHALL B., SRINIVASAN P., TRETSCHK E., YIFAN W., LASSNER C., SITZMANN V., MARTIN-BRUALLA R., LOMBARDI S., ET AL.: Advances in neural rendering. In *Computer Graphics Forum* (2022), vol. 41, Wiley Online Library, pp. 703–735. 3

[VA11] VAN ANTWERPEN D. G.: *Unbiased physically based rendering on the GPU*. Master's thesis, Electrical Engineering, Mathematics and Computer Science, 2011. 6

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

[VHM*22] VERBIN D., HEDMAN P., MILDENHALL B., ZICKLER T., BARRON J. T., SRINIVASAN P. P.: Ref-NeRF: Structured view-dependent appearance for neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2022). 3

[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, 9

[WLM*22] WANG J., LUVIZON D., MUELLER F., BERNARD F., KORTYLEWSKI A., CASAS D., THEOBALT C.: Handflow: Quantifying view-dependent 3d ambiguity in two-hand reconstruction with normalizing flow. In *International Symposium on Vision, Modeling, and Visualization* (2022). 2

[WPFK21] WANG Z., PHILION J., FIDLER S., KAUTZ J.: Learning indoor inverse rendering with 3d spatially-varying lighting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 12538–12547. 10

[WRG*09] WANG J., REN P., GONG M., SNYDER J., GUO B.: All-frequency rendering of dynamic, spatially-varying reflectance. In *ACM SIGGRAPH Asia 2009 papers*. 2009, pp. 1–10. 7, 9

[WSG*23] WANG Z., SHEN T., GAO J., HUANG S., MUNKBERG J., HASSELGREN J., GOJCIC Z., CHEN W., FIDLER S.: Neural fields meet explicit geometric representations for inverse rendering of urban scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023). 10

[WWB*14] WALD I., WOOP S., BENTHIN C., JOHNSON G. S., ERNST M.: Embree: A kernel framework for efficient cpu ray tracing. *ACM Transactions on Graphics (TOG) 33*, 4 (2014). 6

[WZL*22] WANG L., ZHANG J., LIU X., ZHAO F., ZHANG Y., ZHANG Y., WU M., YU J., XU L.: Fourier plenoctrees for dynamic radiance field rendering in real-time. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 13524–13534. 10

[WZY22] WANG C., ZHU Y., YUAN C.: Diverse image inpainting with normalizing flow. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIII* (2022), Springer, pp. 53–69. 2

[XSD*13] XU K., SUN W.-L., DONG Z., ZHAO D.-Y., WU R.-D., HU S.-M.: Anisotropic spherical gaussians. *ACM Transactions on Graphics (TOG) 32*, 6 (2013), 1–11. 2, 6

[XTS*22] XIE Y., TAKIKAWA T., SAITO S., LITANY O., YAN S., KHAN N., TOMBARI F., TOMPKIN J., SITZMANN V., SRIDHAR S.: Neural fields in visual computing and beyond. *Computer Graphics Forum* (2022). 3

[YBHK21] YANG G., BELONGIE S., HARIHARAN B., KOLTUN V.: Geometry processing with neural fields. In *Advances in Neural Information Processing Systems* (2021), vol. 34, pp. 22483–22497. 3

[YS19] YU Y., SMITH W. A.: Inverserendernet: Learning single image inverse rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 3155–3164. 10

[YS21] YU Y., SMITH W. A.: Outdoor inverse rendering from a single image using multiview self-supervision. *IEEE Transactions on Pattern Analysis and Machine Intelligence 44*, 7 (2021), 3659–3675. 10

[YZL*22] YAO Y., ZHANG J., LIU J., QU Y., FANG T., MCKINNON D., TSIN Y., QUAN L.: Neilf: Neural incident light field for material and lighting estimation. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXI* (2022). 3

[ZBX*21] ZHU J., BAI Y., XU Z., BAKO S., VELÁZQUEZ-ARMENDÁRIZ E., WANG L., SEN P., HAŠAN M., YAN L.-Q.: Neural complex luminaires: Representation and rendering. *ACM Trans. Graph. 40*, 4 (2021). 2

[ZHSJ20] ZHANG J., HE T., SRA S., JADBABAIE A.: Why gradient clipping accelerates training: A theoretical justification for adaptivity. In *International Conference on Learning Representations* (2020). 6

[ZIE*18] ZHANG R., ISOLA P., EFROS A. A., SHECHTMAN E., WANG O.: The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), pp. 586–595. 7, 9

[ZLW*21] ZHANG K., LUAN F., WANG Q., BALA K., SNAVELY N.: Physg: Inverse rendering with spherical gaussians for physics-based material editing and relighting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 5453–5462. 11

[ZRW*23] ZELTNER T., ROUSSELLE F., WEIDLICH A., CLARBERG P., NOVÁK J., BITTERLI B., EVANS A., DAVIDOVIČ T., KALLWEIT S., LEFOHN A.: Real-time neural appearance models. *arXiv preprint arXiv:2305.02678* (2023). 6