Learning Neural Transmittance for Efficient Rendering of Reflectance Fields

Mohammad Shafiei[1]
moshafie@ucsd.edu
Sai Bi[1,2]
bisai@cs.ucsd.edu
Zhengqin Li[1]
zhl378@ucsd.edu
Aidas Liaudanskas[3]
aliaudanskas@fyusion.com
Rodrigo Ortiz-Cayon[3]
rcayon@fyusion.com
Ravi Ramamoorthi[1]
ravir@ucsd.edu

[1] University of California San Diego
[2] Adobe Research
[3] Fyusion Inc.

Abstract

Recently neural volumetric representations such as neural reflectance fields have been widely applied to faithfully reproduce the appearance of real-world objects and scenes under novel viewpoints and lighting conditions. However, it remains challenging and time-consuming to render such representations under complex lighting such as environment maps, which requires individual ray marching towards each single light to calculate the transmittance at every sampled point. In this paper, we propose a novel method based on precomputed Neural Transmittance Functions to accelerate the rendering of neural reflectance fields. Our neural transmittance functions enable us to efficiently query the transmittance at an arbitrary point in space along an arbitrary ray without tedious ray marching, which effectively reduces the time-complexity of the rendering. We propose a novel formulation for the neural transmittance function, and train it jointly with the neural reflectance fields on images captured under collocated camera and light, while enforcing monotonicity. Results on real and synthetic scenes demonstrate almost two order of magnitude speedup for renderings under environment maps with minimal accuracy loss.

1 Introduction

Reproducing the appearance of real-world objects and scenes is challenging. Traditional methods [9, 14, 15, 24] recover the reflectance of the scenes and geometry, represented with triangle meshes, to synthesize images under novel conditions. These methods often fail to handle challenging scenes such as those with thin structures and severe occlusions. Neural rendering methods such as neural radiance fields (NeRF) [13] exploit volumetric

scene representations and differentiable ray marching to significantly improve the image quality. The original NeRF is limited to pure view synthesis. Thus, recent works [2, 23] extend the volumetric rendering framework and recover neural reflectance fields (NRF) to support rendering under both novel viewpoints and lighting conditions. They consider light transport and recover the spatial reflectance properties and volume density information.

NRF faithfully reproduces the appearance of objects, but it is time-consuming to render them under complex lighting conditions e.g. environment lighting. Specifically, assume a single-scattering model. For each sampled point on the camera ray, we need to evaluate the transmittance for each light in the scene, which involves another ray marching towards it. Moreover, each sample on the light ray requires an additional network inference.

In this paper, we significantly reduce the time-complexity of the rendering by jointly learning a *Neural Transmittance* function, along with other NRF parameters while training. Given such a function, we can directly query the transmittance of a point along an arbitrary direction without ray marching. The Neural transmittance function is modeled by a Multilayer Perceptron (MLP). In its simplest form, the network could take the position of the point and the desired direction as input, and output the transmittance. However, such a simple formulation does not account for physical priors on the transmittance function such as its monotonicity along rays. Therefore, instead of predicting the transmittance for each sampled point independently, we directly predict the transmittance function for the desired ray.

Our method is based on NRF [2], which applies an MLP to predict the volume density, reflectance, and normal of a point. Similarly, we train our model on multi-view images captured with collocated camera and light. We jointly train the networks for NRF and our neural transmittance function, where we use the transmittance calculated with the former model to supervise the training of the latter model. Since the camera/light rays corresponding to pixels on training images only cover a small portion of the rays that pass the objects, simply training the neural transmittance networks on the rays corresponding to training pixels results in poor generalization to unseen rays. Hence, we introduce an effective data augmentation by randomly sampling rays in space and calculating their transmittance to supervise the training of our neural transmittance networks. In Section 5, we show that our data augmentation effectively improves the quality of the results.

In summary, our main contributions are:

- We propose to jointly train a novel neural transmittance network that enables us to effectively query the transmittance of a point along an arbitrary direction.
- We apply a novel generalized logistic function to model the monotonic transmittance along the ray and use the network to predict the parameters for the logistic function with a two-sphere parameterization.
- We introduce a novel data augmentation method to provide additional supervisions for the training of our neural transmittance networks, which allows our network to better generalize to unseen rays and achieve renderings of higher quality.
- We demonstrate that our method achieves almost two orders of magnitude speedup for rendering under complex lighting conditions such as environment lighting.

2 Related Work

Geometry and reflectance acquistion. To reproduce the appearance of real-world objects, traditional methods [4, 14, 19, 24] apply multi-view stereo to find correspondence across

input images to reconstruct the geometry of the objects, usually represented with a triangle mesh. They estimate the material properties of the object, commonly represented by SVBRDFs, by optimization to match the appearance of rendered images to the captured images. In contrast, we learn a volumetric representation for the objects and propose an efficient method based on learnt precomputed transmittance for efficient rendering.

Neural representations. Recent works exploit neural representations to reproduce the appearance of scenes, which involves using neural networks to learn scene geometry and reflectance. Some methods directly predict the explicit geometry with representations such as point clouds [1], occupancy grids [8, 20] and signed distance fields [10, 18]. Volumetric representations [3, 12, 21] are also applied to acquire the appearance of objects and scenes, where ray marching is performed to render the desired images. Mildenhall et al. [13] propose to use an implicit representation for the volume where an MLP is used to predict the volume density and radiance of an arbitrary point by taking its 3D location and view direction as input. Bi et al. [2] further proposes to jointly learn volume density and reflectance from flash images captured by mobile phones, which supports joint view synthesis and relighting.

Precomputation techniques. Precomputing scene components such as radiance transfer [16, 22] global illumination [6, 17] and visibility [2, 7, 11, 23] is used to accelerate the rendering. Lokovic and Veach [11] precompute a deep shadow map for efficient visibility lookup and high-quality shadows. Kallweit et al. [9] learn a neural network to fit the in-scattered radiance for an arbitrary point and view direction. Bi et al. [2] precompute a transmittance volume at the light by calculating the transmittance of each sampled point on the rays corresponding to the pixels on the virtual image plane placed at the light.

The work concurrent to ours by Srinivasan et al. [23] predicts the transmittance with a network by taking the position of the 3D point and the lighting direction, which fails to conform to physical monotonicity of the transmittance. Compared to Bi et al. [2] and Srinivasan et al. [23] that predict the transmittance for each sample on the ray, our method achieves significant speedups while maintaining high-quality renderings.

3 Background

Our method builds on the framework by Bi et al. [2] that estimates NRF [2] for joint view synthesis and relighting. They use MLPs to predict the reflectance, normal and volume density of points in the scene. They apply differentiable volume rendering via ray marching to render the image. At each sample point on the ray, they determine its contribution with a differentiable reflectance model, which integrates to give the color of that pixel:

$$\boldsymbol{x} = \boldsymbol{r}(t) = \boldsymbol{o} + \boldsymbol{\omega}_o t \tag{1}$$

$$\boldsymbol{L}(\boldsymbol{r}) = \int_0^\infty \tau(\boldsymbol{o}, \boldsymbol{x}, \boldsymbol{\omega}_o)\sigma(\boldsymbol{x})\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{\omega}_o)dt \tag{2}$$

$$\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{\omega}_o) = \int_\Omega \tau(\boldsymbol{l}, \boldsymbol{x}, \boldsymbol{\omega}_i)\boldsymbol{\rho}(\boldsymbol{R}(\boldsymbol{x}), \boldsymbol{\omega}_o, \boldsymbol{\omega}_i, \boldsymbol{n}(\boldsymbol{x}))d\boldsymbol{\omega}_i \tag{3}$$

$$\tau(\boldsymbol{l}, \boldsymbol{x}, \boldsymbol{\omega}) = \exp(-\int_0^{||\boldsymbol{l}-\boldsymbol{x}||} \sigma(\boldsymbol{l} + u\boldsymbol{\omega})du) \tag{4}$$

where $\boldsymbol{x}$ is a 3D point that lies on the ray $\boldsymbol{r}$ at a distance t from camera origin $\boldsymbol{o}$ on ray direction $\boldsymbol{\omega}_o$. Incident radiance $\boldsymbol{L}$ of that ray on the image plane is computed by an integral on the ray. The integrand is a product of transmittance τ along view direction, volume density σ

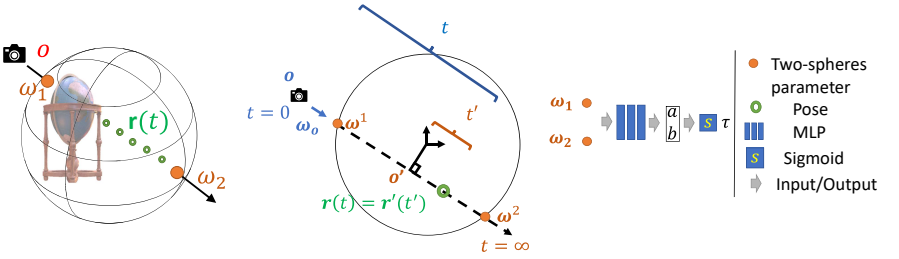

Figure 1: We represent a ray by two-spheres [5] representation i.e. two intersection points on the unit sphere (left). Distance to ray origin is invariant to the camera location (middle). Two-spheres representation is used to define the neural transmittance function (right).

and outgoing radiance $\boldsymbol{h}$. Outgoing radiance is an integral over the product of transmittance along light direction $\boldsymbol{\omega}_i$, a differentiable reflectance model $\boldsymbol{\rho}$ that is a function of reflectance parameters $\boldsymbol{R}$, incoming and outgoing directions and normal vector $\boldsymbol{n}$ on the point $\boldsymbol{x}$. This integral is over the domain of upper hemisphere Ω to the normal. τ is a function of light source location $\boldsymbol{l}$, $\boldsymbol{\omega}_i$ and $\boldsymbol{x}$. Boldface notations in this paper represent vectors.

Note that in this case evaluating the integral in Equation 2 involves a double integral, where we integrate over multiple samples along the camera ray and the outgoing radiance of each sample is determined by an integral over the upper hemisphere at its local surface. During training, with the assumption of a single point light collocated with the camera and a single scattering model, the equation above can be simplified significantly, i.e., the integral in equation 3 is removed, and the transmittance for the light ray and camera ray would be identical, so only a single evaluation is needed. While this assumption alleviates the complicated integral at training time, relighting the scene at test time is still computationally intensive. Specifically, to render the scene under uncollocated camera and light, a naïve rendering process would still need to evaluate a double integral: for each sample on the camera ray, we need to perform another raymarching towards the light to evaluate the light transmittance, where the time-complexity is quadratic in the number of samples.

Bi et al. [2] address this problem by precomputing a transmittance volume inspired by Lokovic and Veach [11]. They place a virtual image plane at the light and march a ray through each pixel and calculate the transmittance of each sample point on the ray, which effectively forms a 3D transmittance volume. During testing, they directly query the transmittance of the desired point by interpolating the precomputed transmittance volume. This strategy reduces the number of network inferences for light transmittance to be linear in the number of samples. However, it requires a large memory to construct such a volume, which can only have a limited resolution. In comparison, we compute a 2D transmittance map instead of an expensive 3D volume, which reduces the time-complexity to be linear in the number of pixels and requires a much smaller amount of memory.

4 Method

We accelerate the rendering of neural volumetric representations such as NRF [2]. Since the dominating factor in running time is the evaluation of light transmittance, we train a network to predict that. Unlike prior work that predicts the transmittance per point [23], we predict it per ray. We model it by a logistic function and achieve almost two orders of magnitude speedup compared to baseline methods while maintaining high-quality renderings.

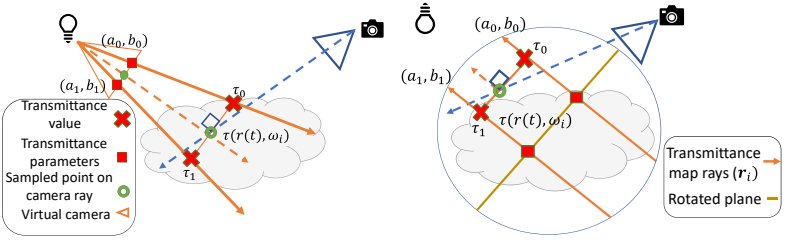

Figure 2: Transmittance map for a point light (left) and a directional light (right). For each point (green dot) we first find the nearest rays on the transmittance map and then evaluate the neural transmittance function for interpolation (Section 4.2).

Ray parametrization. We use two-spheres [5] parametrization to represent a ray independent of the camera and light source positions. Specifically, we assume that the object of interest resides in the unit sphere centered at the world origin. Therefore, every ray passing through the object has two intersection points with the unit sphere, denoted $\boldsymbol{\omega}^1$ and $\boldsymbol{\omega}^2$. We use the two points as our ray representation. To sample a point $\boldsymbol{r}'(t')$ on the ray, we define the original point $\boldsymbol{o}'$ to be the midpoint between $\boldsymbol{\omega}^1$ and $\boldsymbol{\omega}^2$. Then we have,

$$\boldsymbol{r}'(t') = \boldsymbol{o}' + \boldsymbol{\omega}' t' \tag{5}$$

where $\boldsymbol{\omega}'$ is the unit vector pointing from $\boldsymbol{\omega}^1$ to $\boldsymbol{\omega}^2$. Two-spheres parameterization is visualized in Figure 1. Note that we use notation $'$ to distinguish the two-spheres parameterization.

Neural Transmittance. NRF [2] computes the transmittance through ray marching, by sampling σ along the ray and querying a MLP. This is computationally expensive, especially for the environment map rendering where we need to compute the incoming radiance from many directions. We train another MLP to predict transmittance along a ray without ray marching. Hence, we achieve 100 times speedup compared to prior work.

The key observation is that for opaque objects, the numerically computed transmittance can be well modeled by a sigmoid function $\boldsymbol{S}$, (demonstrated in the supplementary material). Therefore, we train a MLP $\boldsymbol{F}$ to predict the slope a and center b of a sigmoid function. The transmittance of a ray $\{\boldsymbol{\omega}^1, \boldsymbol{\omega}^2\}$ can be written as,

$$a, b = \boldsymbol{F}(\boldsymbol{\omega}^1, \boldsymbol{\omega}^2) \qquad a > 0,\ b \in [-1, 1] \tag{6}$$

$$\tau(\boldsymbol{r}'(t'), \omega') = \boldsymbol{S}(a(t' - b)) \tag{7}$$

The predicted transmittance τ can then be substituted in differentiable rendering equation (2) and (3). Therefore, we can train the MLP of $\boldsymbol{F}$ end-to-end, as described in the next section.

4.1 Training

Similar to Bi et al. [2], we train our networks on images captured under previously setup camera and light. We jointly train the networks for NRF and those for transmittance. Given a pixel $\boldsymbol{p}$, our loss function is as follows:

$$L = \alpha_1 L_{\text{nrf}} + \alpha_2 L_{\text{nt}} \tag{8}$$

$$L_{\text{nrf}} = \sum_{\boldsymbol{p}} ||\boldsymbol{I}^{\boldsymbol{p}}_{\text{coarse}} - \boldsymbol{I}^{\boldsymbol{p}}||_2^2 + ||\boldsymbol{I}^{\boldsymbol{p}}_{\text{fine}} - \boldsymbol{I}^{\boldsymbol{p}}||_2^2 \tag{9}$$

$$L_{\text{nt}} = \sum_{\boldsymbol{p}} \sum_{\boldsymbol{x} \in \boldsymbol{r}(\boldsymbol{p})} ||\tau^{\boldsymbol{x}}_{\text{nrf}} - \tau^{\boldsymbol{x}}_{\text{nt}}||_2^2 + \sum_{p} ||\boldsymbol{I}^{\boldsymbol{p}}_{\text{nt}} - \boldsymbol{I}^{\boldsymbol{p}}||_2^2 \tag{10}$$

where α_1 and α_2 are the weight coefficients. Here the losses in Equation 9 minimize the difference between the coarse prediction color $\boldsymbol{I}^{\boldsymbol{p}}_{\text{coarse}}$, fine prediction color $\boldsymbol{I}^{\boldsymbol{p}}_{\text{fine}}$ and ground truth color $\boldsymbol{I}^{\boldsymbol{p}}$. The predicted colors are calculated via differentiable ray marching as specified in Equation 2 and Equation 3. The loss function L_{nrf} is the same used by Bi et al. [2].

Transmittance network loss L_{nt} consists of two terms, namely the transmittance and the color loss. In terms of transmittance loss, for each fine sample $\boldsymbol{x}$ on the ray $\boldsymbol{r}(\boldsymbol{p})$ corresponding to a pixel $\boldsymbol{p}$, we predict its transmittance τ^x_{nt} using our transmittance network, and minimize its difference from the transmittance τ^x_{nrf} calculated from the volume density predicted by the fine NRF network (Equation 2). For the color loss, we calculate the pixel colors using our predicted transmittance and minimize their difference from the ground truth colors.

Training augmentation. Training the transmittance networks on the training images fails to generalize to unseen rays, since they cover a small portion of rays that pass the unit sphere. Thus, we increase the number of training rays particularly for those that are not included in the existing camera rays. Instead of capturing more input images and increasing acquisition complexity, we randomly sample rays in space and use the NRF network to predict their color and the transmittance for points on them as additional supervision to our transmittance network. To this end we randomly sample pairs of points $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$ on the sphere at each iteration, which uniquely determine a set of rays. These sampled rays are used to train the transmittance network only, and we use the same loss function as in Equation 10 to train the transmittance network except that the predicted color by the NRF network is used for the ground truth. In Figure 4 and the supplementary material we visually compare relighting with and without augmentation

4.2 Efficient rendering with precomputed transmittance map

At test time, computing the transmittance of a point is computationally expensive. Naïvely rendering a scene with an environment map requires $l \cdot m \cdot n^2$ queries of the transmittance network where m, l and n are respectively the number of pixels, light sources and samples for ray marching. State of the art methods reduce this time-complexity to $m \cdot l \cdot n$ [2, 23].

We propose a novel method to accelerate the rendering in three steps. First, we sample m rays $\boldsymbol{r}_i, i \in \{0, 1..., m\}$ and precompute the transmittance parameters for each. These parameters are denoted as a_i and b_i and require $m \cdot l$ neural network queries. Second, for each point of interest, we find points that lie on four closest $\boldsymbol{r}_i$. Third, we interpolate the transmittance for the point of interest. Algorithms are provided in the supplementary material.

Transmittance map is computed in two steps. We compute the rays $\boldsymbol{r}_i$ depending on the type of light source and then we compute the parameters for each ray. For a point light, we place a virtual image plane there with m pixels. $\boldsymbol{r}_i$ is the outgoing ray from pixel i of the virtual image plane. For a directional light, we sample a set of parallel rays toward the direction of the light that pass from the unit sphere. To this end, we uniformly sample m points on a circle perpendicular to the light direction passing the world origin. $\boldsymbol{r}_i$ in this case originates on the sample point on the circle and points toward the light source direction.

After computing rays $\boldsymbol{r}_i$, we find the two-spheres parameters by intersecting them with the unit sphere (Section 4). We use our transmittance network to precompute a_i and b_i. Transmittance map is depicted in Figure 2 on a 2D example.

Nearest Points. During rendering, given a point of interest $\boldsymbol{x}$ and light direction $\boldsymbol{\omega}_i$, we approximate the transmittance as follows. We first find a set of four rays close to $\boldsymbol{x}$ denoted as $\boldsymbol{Q} = \{\boldsymbol{r}_0, \boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3\}$. We can then find a set of four close points to $\boldsymbol{x}$, namely $\mathcal{X} = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3\}$, that reside on these rays. Using Equation 7 we can compute the transmittance

Scene	Transmittance			Overall		
	Ours	NRF	Speed up	Ours	NRF	Speed up
Point light	0.25	96.73	**384.10**	65.49	162.88	**2.48**
Environment map	122.38	47 806.81	**390.62**	522.45	48 151.80	**92.16**

Table 1: Runtime of our method compared to NRF in seconds. Our method is 2 times faster than NRF for relighting with a point light and 92 times faster for relighting with a 50×10 environment map.

values for points in $\mathcal{X}$. Then we approximate the transmittance on $\boldsymbol{x}$ by interpolation. These steps are explained below and depicted in Figure 2.

For the case of a point light, we project $\boldsymbol{x}$ on the virtual image plane and locate its four nearest neighbor pixels and their corresponding rays $\boldsymbol{Q}$. Four closest points in $\mathcal{X}$ are merely the intersection of the perpendicular plane to $\boldsymbol{\omega}_i$ that passes through $\boldsymbol{x}$ and the nearest rays $\boldsymbol{Q}$. For rendering with a directional light, we find closest rays $\boldsymbol{Q}$ by projecting the point $\boldsymbol{x}$ on a plane orthogonal to $\boldsymbol{\omega}_i$ that passes the origin. We then find $\mathcal{X}$ by intersecting the plane orthogonal to the light direction which passes through $\boldsymbol{x}$ with four rays in $\boldsymbol{Q}$.

After finding the set $\mathcal{X}$ of four closest points, we can compute their transmittance by Equation 7 and find the transmittance on $\boldsymbol{x}$ using bilinear interpolation. As shown in Figure 4 and Table 1, our method achieves almost two orders of magnitude speedup compared to baseline methods without sacrificing the image quality.

Implementation. We follow the training protocols of NRF. The input of our algorithm is a set of images taken from an object with a mobile phone with flashlight in a darkroom. We capture around 400 per object. The camera poses and intrinsic parameters are estimated with COLMAP [19]. We translate and scale the scene to fit the reconstructed geometry into a unit sphere at the origin. We jointly train the neural networks for NRF and the transmittance network with a batch size of 4. For each batch, we randomly select a training image and sample 16×16 pixels for training. In addition, we also randomly sample 128 rays as described in Section 4.1 to generate augmented training data for the transmittance networks. We optimize the networks using Adam optimizer [9] with a learning rate of $2e^{-5}$. The networks are trained on 4 Nvidia RTX 2080 GPUs, and the full training takes around 2 days.

5 Results

We evaluate our method in three aspects. First, we demonstrate that our method is more than 92 times faster than Bi et al.[2]. Second, we show that our augmentation method is necessary for neural transmittance to avoid overfitting. Third, we show that, even though our method includes approximation steps, our results are still qualitatively and quantitatively comparable to prior work. We run our experiments on 3 synthetic scenes, namely, *Happy Buddha*, *Sitting Buddha* and *Globe* and 2 real scenes, namely, *Girl* and *Pony*. All of the synthetic scenes, Girl and Pony are rendered with 600×600, 592×478 and 763×544 resolution images respectively. We use 500 images to train the synthetic scenes and respectively 384 and 380 images for training the Girl and Pony. We show our results on view synthesis and relighting with point light on an image sequence in the supplementary material.

Runtime. We compare our method with that of Bi et al. [2]. We gain more (92 times) speed up for rendering a scene under environment maps with higher resolution than those with low resolution (2 times). We plot the runtime of our method and Bi et al. [2] in Figure 3. To this end, we run both methods on 500 different environment maps. Each of these maps have a unique resolution, namely, $i \in \{1,2,3,...,500\}$ pixels. We compute the runtime for

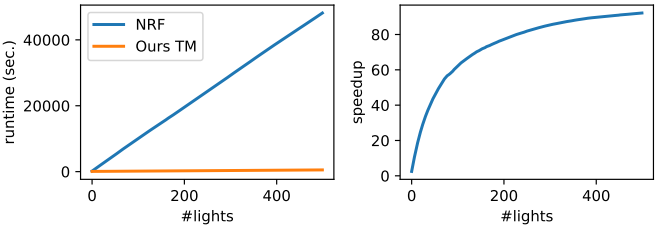

Figure 3: Overall runtime for relighting of the Happy Buddha scene (left). We relight the scene with many environment maps each with a unique resolution. The speed up is the division of NRF runtime by our method with Transmittance Map (TM), shown on the right.

these experiments and the results are shown in Figure 3. Figure 3 shows the runtime of both methods and the speed up of our method compared to Bi et al. [2]. We show that our method allows rendering with high resolution environment map in Figure 5. NRF [2] takes more than 251 hours to render in this resolution.

The speed up curve in Figure 3 shows that our method is faster for larger environment maps compared to smaller ones. In more detail, the bottle neck for the runtime of neural rendering is neural network queries. These queries are required to retrieve reflectance and transmittance values. Our method, similar to NRF, requires $n \cdot m$ queries for reflectance. Additionally, our method requires $l \cdot m$ queries for transmittance while NRF requires $l \cdot m \cdot n$. As a result, the ratio of overall queries for NRF compared to ours is $\psi = \frac{v_{NRF}}{v_{ours}} = \frac{l \cdot m \cdot n + n \cdot m}{l \cdot m + n \cdot m}$ where v_{NRF}, v_{ours} and ψ are respectively the time complexity of NRF [2], that of our method and the hypothetical speed up of our method. For one light source this is a constant $\frac{2n}{n+1}$ and for large number of light sources it approaches to n that is the sample count along a ray. Sample count is normally chosen to be 192 which suggests that our method should be almost 192 times faster than Bi et al.[2] for large environment maps. Our speed up plot in Figure 3 however, converges to a constant value around 92. This value is lower than the hypothetical speed up (192) since our algorithm requires additional steps including nearest neighbor lookup and interpolation. It is also clear from Table 1 which shows that our method is 2 $\times$ faster for relighting with uncollocated point light. Table 1 moreover shows that our method is more than 380$\times$ faster for in the precomputation stage compared to NRF. In the supplementary material we compare the time for creating transmittance map compared to transmittance volume of Bi et al [2] for different resolutions.

Parametrization and Augmentation. The two-spheres parametrization enables us to use a monotonic function for the definition of neural transmittance. Naively fitting such a function only over the view directions of input images would lead to overfitting and our augmentation method leads to higher accuracy. To demonstrate this, we compute the accuracy of our method with and without augmentation in Table 2 for both cases of relighting the synthetic scenes with point light and environment map. For most of the cases the augmentation method leads to higher accuracy by either or both metrics. We visually show the necessity of augmentation in Figure 4 and in the supplementary material. In Figure 4 we compare our method with Neural Transmittance (NT) only and to that with augmentation and transmittance map. This figure shows that the overfitting artifacts are removed by our augmentation method. We show an ablation study in the supplementary material.

Accuracy. Although our method uses many approximation steps in favor of efficient rendering, it is still quantitatively and qualitatively comparable to prior work. We show this in Table 2 which includes the error values of different steps of our algorithm. These errors are

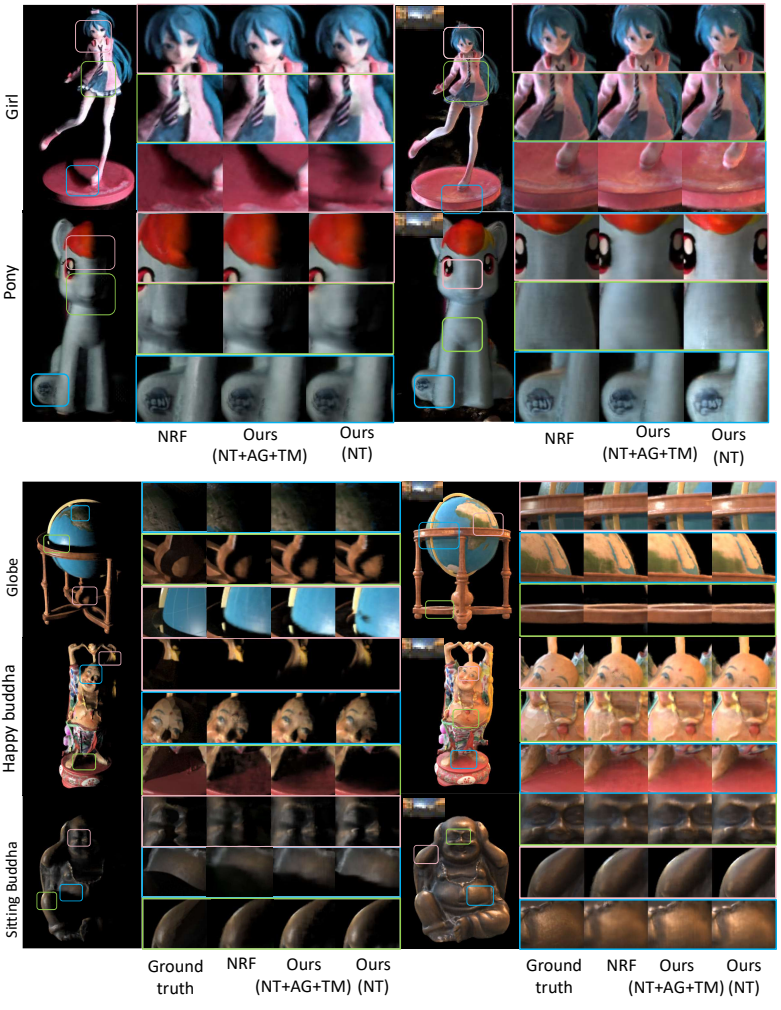

Figure 4: Relighting results with a point light (left) and with 24×12 pixels environment map (right). The quality of relighting with our method is comparable to that of NRF and ground truth. We compare our method with Neural Transmittance (NT) only and with augmentation (AG) and Transmittance Map (TM). Augmentation removes the overfitting artifacts.

computed on synthetic scenes under point light and environment map lighting. We evaluate our method quantitatively on 400 and 18 images respectively for point light and environment lighting. In the first 200, the light source moves around the object on a sphere. In the next 200, the view angle also moves on the same position as light. For environment map cases, we relight each scene with 18 different environment maps from a fixed view point. We compare each of these images with the ground truth and take the average error. In most of the rows in Table 2, our method is quantitatively similar or slightly worse than NRF despite the many approximation steps. Nevertheless, the resulting error is small.

Scene	NRF	Ours TM	Ours AG	No AG	NRF	Ours TM	Ours AG	No AG
	Point light							
	MS-SSIM				LPIPS			
Happy Buddha	0.963	0.954	0.954	0.948	0.049	0.055	0.055	0.058
Sitting Buddha	0.985	0.973	0.973	0.965	0.066	0.049	0.049	0.042
Globe	0.681	0.666	0.666	0.668	0.219	0.229	0.229	0.228
	rMSE				SSIM			
Happy Buddha	0.051	0.060	0.059	0.058	0.955	0.945	0.945	0.939
Sitting Buddha	0.011	0.022	0.022	0.026	0.976	0.967	0.967	0.959
Globe	0.107	0.126	0.126	0.127	0.739	0.732	0.732	0.732
	Environment map							
	MS-SSIM				LPIPS			
Happy Buddha	0.958	0.937	0.938	0.940	0.071	0.077	0.077	0.076
Sitting Buddha	0.952	0.903	0.890	0.870	0.095	0.085	0.090	0.075
Globe	0.911	0.859	0.837	0.840	0.086	0.107	0.114	0.117
	rMSE				SSIM			
Happy Buddha	0.148	0.201	0.201	0.190	0.937	0.914	0.914	0.916
Sitting Buddha	0.041	0.081	0.085	0.097	0.939	0.895	0.880	0.864
Globe	0.159	0.240	0.281	0.281	0.899	0.851	0.831	0.835

Table 2: We compare the accuracy of our method to the baseline. Our approximation method with Transmittance Map (Ours TM), has only a minimal drop of accuracy compared to NRF. We achieve higher accuracy by Augmentation (Ours AG) than no augmentation (No AG). (rMSE, (MS-)SSIM and LPIPS respectively stand for Root Mean Squared Error, (Multi-Scale) Structure Similarity Index Measure and Learned Perceptual Image Patch Similarity.)

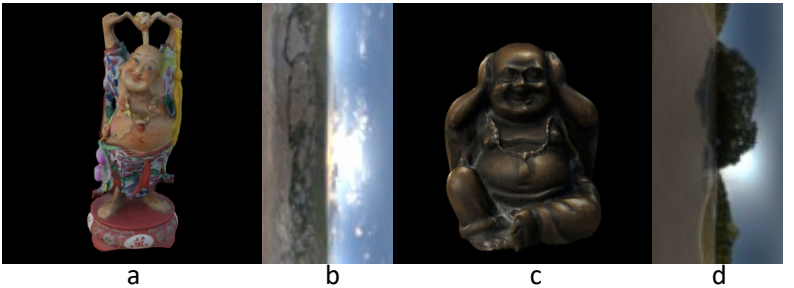

Figure 5: We render Happy Buddha (a) and Sitting Buddha (b) with 64×128 environment maps (b and d). It takes 2.7 hours with our method to render each of these scenes and potentially more than 251 hours with prior work.

6 Discussion

We introduced Neural Transmittance and Transmittance Map which allow more than $92\times$ faster rendering with Neural Reflectance Fields under environment maps and almost $2\times$ under point lights. Despite the necessary approximations, our method is qualitatively and quantitatively comparable to prior work. We also introduced an augmentation method that avoids overfitting of transmittance. We expect to see this augmentation method used for other quantities defined on the light field domain such as radiance, depth, reflectance, etc.

An interesting future direction is to formulate a physically consistent transmittance function to derive the volume density. Such a formulation would allow training only one network for both of these quantities and potentially for all of those required for volumetric rendering. In our method, these two quantities are decoupled that results in inaccuracies such as intensity shift. However, this does not lead to significant error as demonstrated in Section 5.

7 Acknowledgement

The authors thank Manmohan Chandraker for helpful discussion. This work was funded in part by the Ronald L. Graham Chair, ONR grants N000141912293, N000141712687 and NSF grant 1730158.

References

[1] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. *arXiv preprint arXiv:1906.08240*, 2019.

[2] Sai Bi, Zexiang Xu, Pratul Srinivasan, Ben Mildenhall, Kalyan Sunkavalli, Miloš Hašan, Yannick Hold-Geoffroy, David Kriegman, and Ravi Ramamoorthi. Neural reflectance fields for appearance acquisition. *arXiv preprint arXiv:2008.03824*, 2020.

[3] Sai Bi, Zexiang Xu, Kalyan Sunkavalli, Miloš Hašan, Yannick Hold-Geoffroy, David Kriegman, and Ravi Ramamoorthi. Deep reflectance volumes: Relightable reconstructions from multi-view photometric images. *ECCV*, 2020.

[4] Sai Bi, Zexiang Xu, Kalyan Sunkavalli, David Kriegman, and Ravi Ramamoorthi. Deep 3d capture: Geometry and reflectance from sparse multi-view images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5960–5969, 2020.

[5] Emilio Camahort, Apostolos Lerios, and Donald Fussell. Uniformly sampled light fields. In *Eurographics Workshop on Rendering Techniques*, pages 117–130. Springer, 1998.

[6] Zhao Dong, Jan Kautz, Christian Theobalt, and Hans-Peter Seidel. Interactive global illumination using implicit visibility. In *15th Pacific Conference on Computer Graphics and Applications (PG'07)*, pages 77–86. IEEE, 2007.

[7] Simon Kallweit, Thomas Müller, Brian Mcwilliams, Markus Gross, and Jan Novák. Deep scattering: Rendering atmospheric clouds with radiance-predicting neural networks. *ACM Transactions on Graphics (TOG)*, 36(6):1–11, 2017.

[8] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. *NIPS*, 2017.

[9] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[10] Yiyi Liao, Simon Donne, and Andreas Geiger. Deep marching cubes: Learning explicit surface representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2916–2925, 2018.

[11] Tom Lokovic and Eric Veach. Deep shadow maps. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 385–392, 2000.

[12] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *arXiv preprint arXiv:1906.07751*, 2019.

[13] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[14] Giljoo Nam, Joo Ho Lee, Diego Gutierrez, and Min H Kim. Practical svbrdf acquisition of 3d objects with unstructured flash photography. *ACM Transactions on Graphics (TOG)*, 37(6):1–12, 2018.

[15] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '11, pages 127–136, Washington, DC, USA, 2011. IEEE Computer Society. ISBN 978-1-4577-2183-0. doi: 10.1109/ISMAR.2011.6092378. URL http://dx.doi.org/10.1109/ISMAR.2011.6092378.

[16] Ren Ng, Ravi Ramamoorthi, and Pat Hanrahan. All-frequency shadows using nonlinear wavelet lighting approximation. In *ACM SIGGRAPH 2003 Papers*, pages 376–381. 2003.

[17] Tobias Ritschel, Thomas Engelhardt, Thorsten Grosch, H-P Seidel, Jan Kautz, and Carsten Dachsbacher. Micro-rendering for scalable, parallel final gathering. *ACM Transactions on Graphics (TOG)*, 28(5):1–8, 2009.

[18] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2304–2314, 2019.

[19] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[20] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2437–2446, 2019.

[21] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems*, pages 1121–1132, 2019.

[22] Peter-Pike Sloan, Jan Kautz, and John Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 527–536, 2002.

[23] Pratul P. Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T. Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *arXiv*, 2020.

[24] Zhiming Zhou, Guojun Chen, Yue Dong, David Wipf, Yong Yu, John Snyder, and Xin Tong. Sparse-as-possible SVBRDF acquisition. *ACM Transactions on Graphics*, 35 (6):189, 2016.