

Supplementary Material for Learning Neural Transmittance for Efficient Rendering of Reflectance Fields

Mohammad Shafiei¹

moshafie@ucsd.edu

Sai Bi^{1,2}

bisai@cs.ucsd.edu

Zhengqin Li¹

zhl378@ucsd.edu

Aidas Liaudanskas³

aliaudanskas@fyusion.com

Rodrigo Ortiz-Cayon³

rcayon@fyusion.com

Ravi Ramamoorthi¹

ravir@ucsd.edu

¹ University of California San Diego

² Adobe Research

³ Fyusion Inc.

1 Architecture.

Our neural network architecture is identical to that used by Mildenhall et al. [10] and Bi et al. [11]. The architecture is depicted in Figure 1.



Figure 1: We use the same neural network architecture as that of Mildenhall et al. [10] and Bi et al. [11].

2 Transmittance map

In the main paper, we proposed efficient rendering with precomputed transmittance map (main paper section 4.2). Creating transmittance map is described by Algorithm 1. After its creation, it can be used to compute the transmittance for each point of interest \mathbf{x} and \mathbf{r} . To this end, we first find the four closest rays to \mathbf{x} , namely $\mathcal{Q} = \{\mathbf{r}_0, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3\}$ and the closest points to \mathbf{x} that reside on those ray, denoted as $\mathcal{X} = \{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}$. Then we find the

transmittance value for points in \mathcal{X} as described in Algorithm 2. Having the transmittance of four closest points to \mathbf{x} , we can use interpolation to compute the transmittance on point \mathbf{x} as in Algorithm 3.

Algorithm 1: Transmittance map (references are to the main paper)

```

 $\mathbf{x}_i \leftarrow$   $i$ th pixel of a virtual image plane with  $m$  pixels
 $\mathbf{p}_i \leftarrow$   $i$ th point of a set uniformly sampled on a circle
Result:  $(a_i, b_i) \in \{1, \dots, m\}$ 
foreach  $i \in \{1, 2, \dots, m\}$  do
  if Light source is point light then
     $\mathbf{q} \leftarrow$  Compute two-spheres parameters of pixel  $\mathbf{x}_i$  (sec. 4)
     $\{a_i, b_i\} \leftarrow \{a(\mathbf{q}), b(\mathbf{q})\}$  (sec. 4)
  else if Light source is directional then
     $\mathbf{p}'_i \leftarrow$  Rotate  $\mathbf{p}_i$  toward light
     $\mathbf{q} \leftarrow$  Compute two-spheres parameters of point  $\mathbf{p}'_i$  in direction of  $\boldsymbol{\omega}_i$  (sec. 4)
     $\{a_i, b_i\} \leftarrow \{a(\mathbf{q}), b(\mathbf{q})\}$  (sec. 4)
  end
end

```

Algorithm 2: Nearest rays and points on transmittance map (references are to the main paper)

```

Result:  $(\mathbf{q}_i, t_i) i \in \{1, 2, 3, 4\}$ 
 $\boldsymbol{\omega}_i \leftarrow$  light direction on point  $\mathbf{p}$ 
foreach  $i \in 1, 2, 3, 4$  do
  if Light source is point light then
     $\mathbf{x} \leftarrow$  Project  $\mathbf{p}$  to virtual image plane
     $\mathbf{y}_i \leftarrow$  Find  $i$ th nearest points on image plane to  $\mathbf{x}$ 
     $\mathbf{q} \leftarrow$  Compute two-spheres parameters of pixel  $\mathbf{y}_i$  (sec. 4)
     $\mathbf{q}' \leftarrow$  Intersect the orthogonal plane to  $\boldsymbol{\omega}_i$  that passes point  $\mathbf{p}$  with  $\mathbf{q}$ 
     $t_i = \|\boldsymbol{\sigma}'(\mathbf{q}) - \mathbf{q}'\|$  (eq. 5)
     $\tau_i \leftarrow \tau(\mathbf{q}, t_i)$  (eq. 7)
  else if Light source is directional then
     $\mathbf{p}' \leftarrow$  Project  $\mathbf{p}$  on orthogonal plane to  $\boldsymbol{\omega}_i$ 
     $\mathbf{u} \leftarrow$  Rotate  $\mathbf{p}'$  to  $xy$  plane
     $\mathbf{v}_i \leftarrow$  Find  $i$ th closest rays
     $\mathbf{q} \leftarrow$  Intersect the orthogonal plane to  $\boldsymbol{\omega}_i$  that passes  $\mathbf{p}$  with ray  $\mathbf{v}_i$ 
     $t_i \leftarrow \|\mathbf{q} - \mathbf{p}'\|$ 
     $\tau_i \leftarrow \tau(\mathbf{q}, t_i)$ 
  end
end

```

Algorithm 3: Transmittance map and interpolation step (equation references are to the main paper)

Result: $\tau(\mathbf{p}, \omega_o)$

$l \leftarrow$ number of light sources

$n \leftarrow$ number of sampled points in space

\mathbf{P} set of all sampled points along view direction

$\mathbf{a}, \mathbf{b} \leftarrow$ initialize transmittance map alg. 1

foreach $i \in \{1, 2, \dots, l\}$ **do**

foreach $\mathbf{p} \in \mathbf{P}$ **do**

$\mathbf{Q} \leftarrow$ four nearest rays to \mathbf{p} towards i th light source alg. 2

 //Compute transmittance on neighboring rays

foreach $\mathbf{q} \in \mathbf{Q}$ **do**

$\mathbf{q}' \leftarrow$ find a point on ray \mathbf{q} close to $\mathbf{r}(t)$ alg. 2

$t_k \leftarrow$ distance of \mathbf{q}' to ray origin (eq. 5)

$\tau_k \leftarrow \tau(\mathbf{q}, t_k)$ (eq. 7)

end

 //Compute $\tau(\mathbf{p}, \omega_o)$ by bilinear interpolation

$\mathbf{u} = \frac{\mathbf{p}_0 - \mathbf{p}_1}{\|\mathbf{p}_0 - \mathbf{p}_1\|}$

$\mathbf{v} = \frac{\mathbf{p}_0 - \mathbf{p}_2}{\|\mathbf{p}_0 - \mathbf{p}_2\|}$

$\boldsymbol{\alpha} = \langle \mathbf{p} - \mathbf{q}_0, \mathbf{u} \rangle$

$\boldsymbol{\beta} = \langle \mathbf{p} - \mathbf{q}_0, \mathbf{v} \rangle$

$\tau_t \leftarrow \alpha_x \tau_0 + (1 - \alpha_x) \tau_1$

$\tau_b \leftarrow \alpha_x \tau_2 + (1 - \alpha_x) \tau_3$

$\tau(\mathbf{p}, \omega_o) \leftarrow \beta_y \tau_t + (1 - \beta_y) \tau_b$

end

end

3 Additional Results

Neural Transmittance. Our method is based on the assumption that, objects in the scene are opaque. More specifically, the transmittance function for an opaque scene can be well modelled by a logistic function. Our proposed Neural Transmittance Function (Equation 7 in the main paper) estimates the transmittance along a ray by two parameters namely slope a and center b . We compare the neural transmittance to the transmittance function computed by NRF [14] in Figure 2. NRF [14] computes the transmittance by the volume density values queried from a neural network. These queries are made for the sampled points along a ray in the ray marching step. Then, these values are used for numeric integration. While in the Neural Transmittance Function, we simply estimate the transmittance values for the same points by the prediction of function parameters a and b . These parameters allow us to compute the transmittance without ray marching. In Figure 2 we compare our Neural Transmittance Function to the transmittance computed by NRF [14] in different stages of training. In the early stages and before convergence the both functions are smooth. In the late stages, the both functions look like a logistic function with large slope. This figure is a plot of the transmittance values of points along a ray in different stages of optimization. The ray is chosen from a training image of the Sitting Buddha scene which passes from the

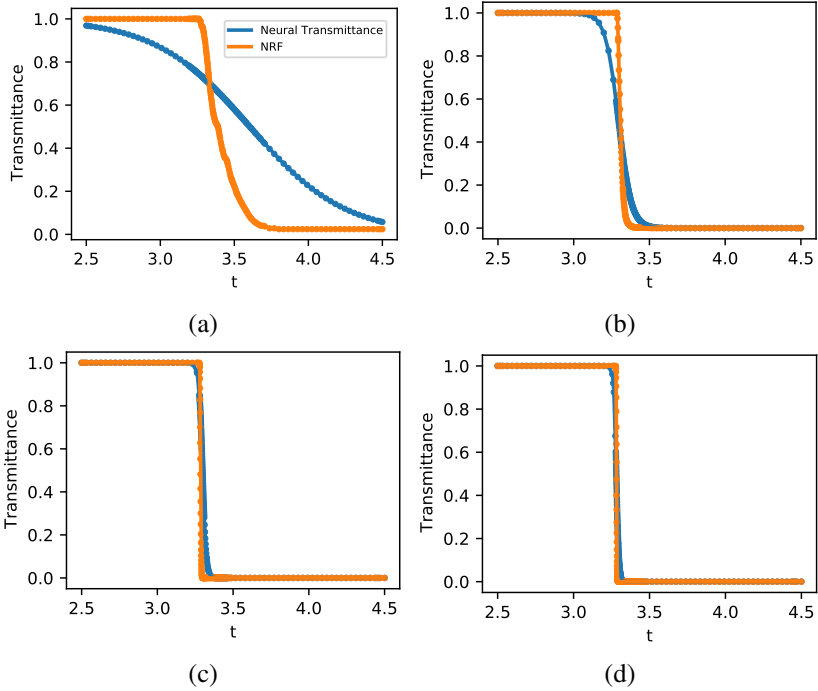


Figure 2: Numerically computed transmittance (red curve) is smooth in the early stages of training (a,b and c) and gradually converges to nearly a step function (d). Neural transmittance function (blue curve) fits well to the numerically computed transmittance, while this function is simultaneously trained with those of NRF (training details explained in Section 4.1 in the main paper). (a),(b),(c) and (d) are respectively generated after 6000, 12000, 30000 and 670000 iterations.

geometry.

Runtime and complexity. Relighting a scene with NRF is a time consuming process especially for the case of environment map rendering. We compare the time-complexity of our precomputation method to that of Bi et al. [10] in Table 1. We also compare the runtime of the precomputation step for our method compared to NRF in Figure 3. Our method is significantly faster for various scenes.

Ablation. We show the effect of our augmentation method in Figure 4. Relighting the globe scene without augmentation leads to clear artifacts.

Visual comparison. We visually compare our method with Bi et al. [10] in Figure 4. This figure includes relighting real and synthetic scenes under point light and an environment map.

References

- [1] Sai Bi, Zexiang Xu, Pratul Srinivasan, Ben Mildenhall, Kalyan Sunkavalli, Miloš Hašan, Yannick Hold-Geoffroy, David Kriegman, and Ravi Ramamoorthi. Neural reflectance

Naive	NRF	Ours AG	Ours TM
$l \cdot n^2 \cdot c$	$l \cdot m \cdot n$	$l \cdot m \cdot n$	$l \cdot m$

Table 1: Time-complexity of relighting a scene with an environment map. l, m, n and c are respectively the number of pixels on an environment map, pixels on a virtual camera on each light source, samples on a ray and pixels on the camera.

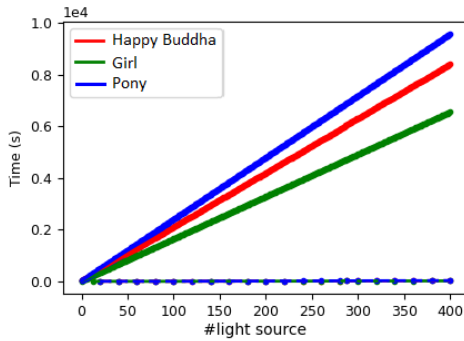


Figure 3: Runtime of our method (broken lines) compared to NRF (solid lines) for evaluating transmittance map and transmittance volume [14]. Transmittance map is faster to compute compared to transmittance volume for 3 different scenes each with a unique image resolution. The vertical axis is runtime in seconds and the horizontal axis is the resolution of environment maps in pixels. The lines corresponding to our method almost overlay each other on this plot.

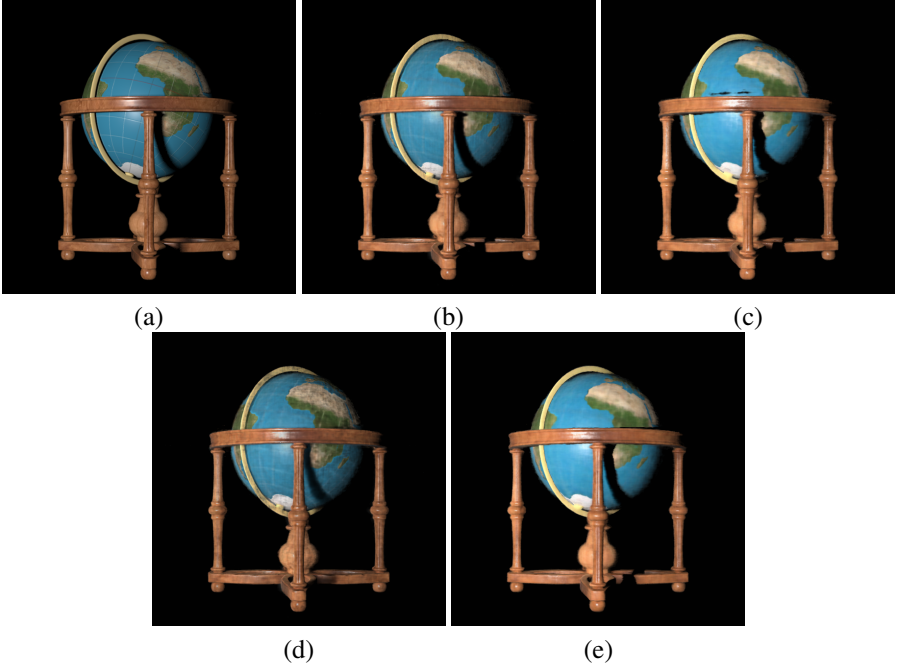


Figure 4: We can relight the scene with our method (e) faster than NRF (b) and with less artifacts compared to NeRV visibility function (d). Our augmentation approach removes the artifacts caused by overfitting of neural transmittance on input images (c).

fields for appearance acquisition. *arXiv preprint arXiv:2008.03824*, 2020.

- [2] 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.