Neural Radiance Transfer Fields for Relightable Novel-view Synthesis with Global Illumination – Supplemental Document –

Linjie Lyu[1], Ayush Tewari[2], Thomas Leimkühler[1], Marc Habermann[1], and Christian Theobalt[1]

[1]Max Planck Institute for Informatics, Saarland Informatics Campus
[2]MIT

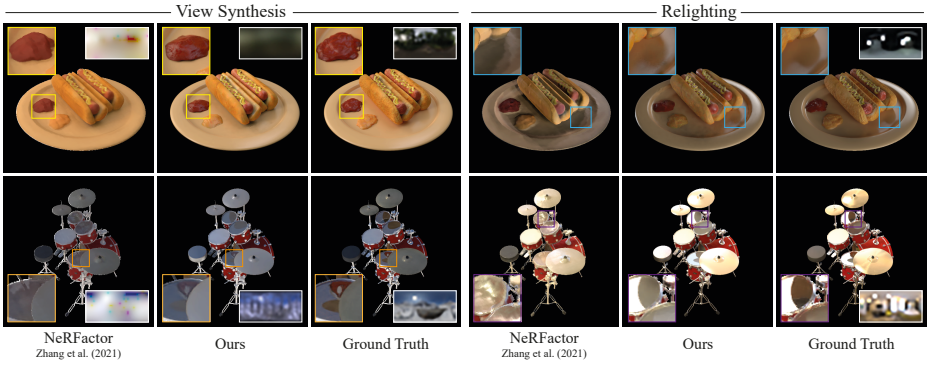

Fig. 1. Comparison of our approach to NeRFactor [4] on the tasks of novel-view synthesis and relighting. For view synthesis, white insets show estimated (first two columns) and ground truth (third column) environment maps. For relighting, the white inset shows the environment map used to illuminate the scene.

In the following, we compare our approach to NeRFactor [4] on their own dataset (Sec. 1). Further, we provide more implementation details concerning the network architecture, training, and the regularizers we use (Sec. 2).

1 Comparison with NeRFactor

In addition to the comparison in the main document,we provide a comparison with NeRFactor [4] on their own dataset, i.e. the Drums and Hotdog scenes, in Fig. 1 and in Tab. 1. For generating the results for NeRFactor, we used trained models provided by the original authors. For view synthesis, we randomly sampled 8 views from the 100 test views and computed the average for all metrics. For relighting, we use the same views and chose the environment map number 3 of their test set. Note that, visually, our approach synthesizes more detailed images and is able to recover secondary light bounce effects such as the reflection on the

Table 1. Numerical evaluation for novel-view synthesis and relighting. We compare to the recent state of the art NeRFactor [4] in terms of image-based metrics, i.e. PSNR and SSIM, and perceptual metrics, i.e. LPIPS. PSNR and SSIM were determined on foreground pixels only. For both tasks, novel-view synthesis and relighting, we achieve the best performance.

	Novel View Synthesis			Novel View Synthesis & Relighting		
Method	**PSNR ↑**	**SSIM ↑**	**LPIPS ↓**	**PSNR ↑**	**SSIM ↑**	**LPIPS ↓**
NeRFactor [4]	18.67	0.684	0.13315	16.48	0.6439	0.0965
Ours	**20.82**	**0.7577**	**0.0659**	**20.18**	**0.7556**	**0.0786**

drums, which NeRFactor cannot reproduce at all. This observation is consistent for both the view synthesis and the relighting task. Moreover, we also found that our method outperforms NeRFactor quantitatively on their own dataset for all metrics, which further confirms the improvement of our method compared to the previous state-of-the-art.

2 Implementation Details

Next, we provide more details on the network architecture (Sec. 2.1), the training procedure (Sec. 2.2), and our regularizers (Sec. 2.3).

2.1 Network Architecture

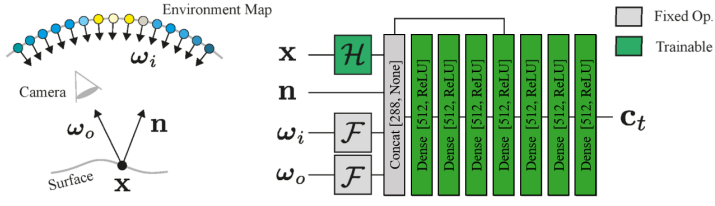

Fig. 2. The geometry of our setup (left) and our network architecture (right). Expressions in brackets denote [output features, activation function]. To shade a pixel, we iterate over all incoming directions $\boldsymbol{\omega}_i$. Note that the transferred radiance $\mathbf{c}_t$ takes multiple light bounces into account. For details on $\mathcal{H}$ and $\mathcal{F}$ refer to the main text.

We model the radiance transfer function T using our neural radiance transfer field

$$T_\theta\left(\mathcal{H}(\mathbf{x}), \mathbf{n}, \mathcal{F}(\boldsymbol{\omega}_i), \mathcal{F}(\boldsymbol{\omega}_o)\right) = \mathbf{c}_t, \tag{1}$$

which is represented by a multi-layer perceptron (MLP). First, we encode position $\mathbf{x}$ using a multi-resolution hash function $\mathcal{H}$ as proposed by Müller et al.

[2], with parameters $L = 16$, $F = 16$, $N_{\text{min}} = 16$, and $N_{\text{max}} = 256$. We only create $1 : 1$ hash mapping for the voxels near the mesh surface,thus avoiding a hash collision. The directions ω_i and ω_o are positionally encoded using spherical harmonics features $\mathcal{F}$ [3] with $n = 4$ frequencies, resulting in 16 features each. The surface normal $\mathbf{n}$ is fed directly. To parameterize T_θ, we use an 8-layer MLP, where each hidden layer has 512 features and a ReLU activation function. We additionally use a skip connection between the input layer and the fourth layer. θ denotes the trainable parameters of the network and the hash encoding. A visualization is shown in Fig. 2.

2.2 Training

Our approach is implemented in Pytorch. For training, we use the Adam optimizer [1]. For optimizing the environment map, the albedo map, and the material coefficients, we optimize for 3000 iterations for each scene with a learning rate of 1 for the environment map, 0.1 for the albedo map, and 0.001 for the material coefficients. For training our NRTF, during pre-training on the OLAT dataset only, we apply 150k iterations for each scene with a learning rate of 5e-4 and a batch size of 20 OLAT images. For joint optimization, we apply 100k iterations for each scene with a learning rate of 1e-4. Each batch contains five OLAT images and one real image.

2.3 Regularizers

For the image-based regularizers, we approximate image gradients with backward differencing. Further, the weights of the inidividual regularizers are $\lambda_{\text{reg}} = 0.001$, $\lambda_{\text{BSDF}} = 0.1$, $\lambda_{\text{OLAT}} = 0.1$, $\lambda_{\text{PRT}} = 1$, $\lambda_{\text{EnvC}} = 0.001$.

References

1. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR **abs/1412.6980** (2015)
2. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. arXiv:2201.05989 (Jan 2022)
3. Yu, A., Fridovich-Keil, S., Tancik, M., Chen, Q., Recht, B., Kanazawa, A.: Plenoxels: Radiance fields without neural networks. arXiv preprint arXiv:2112.05131 (2021)
4. Zhang, X., Srinivasan, P.P., Deng, B., Debevec, P., Freeman, W.T., Barron, J.T.: Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. ACM Transactions on Graphics (TOG) **40**(6), 1–18 (2021)