Supplement to FreeNeRF: Improving Few-shot Neural Rendering with Free Frequency Regularization

Project page: FreeNeRF

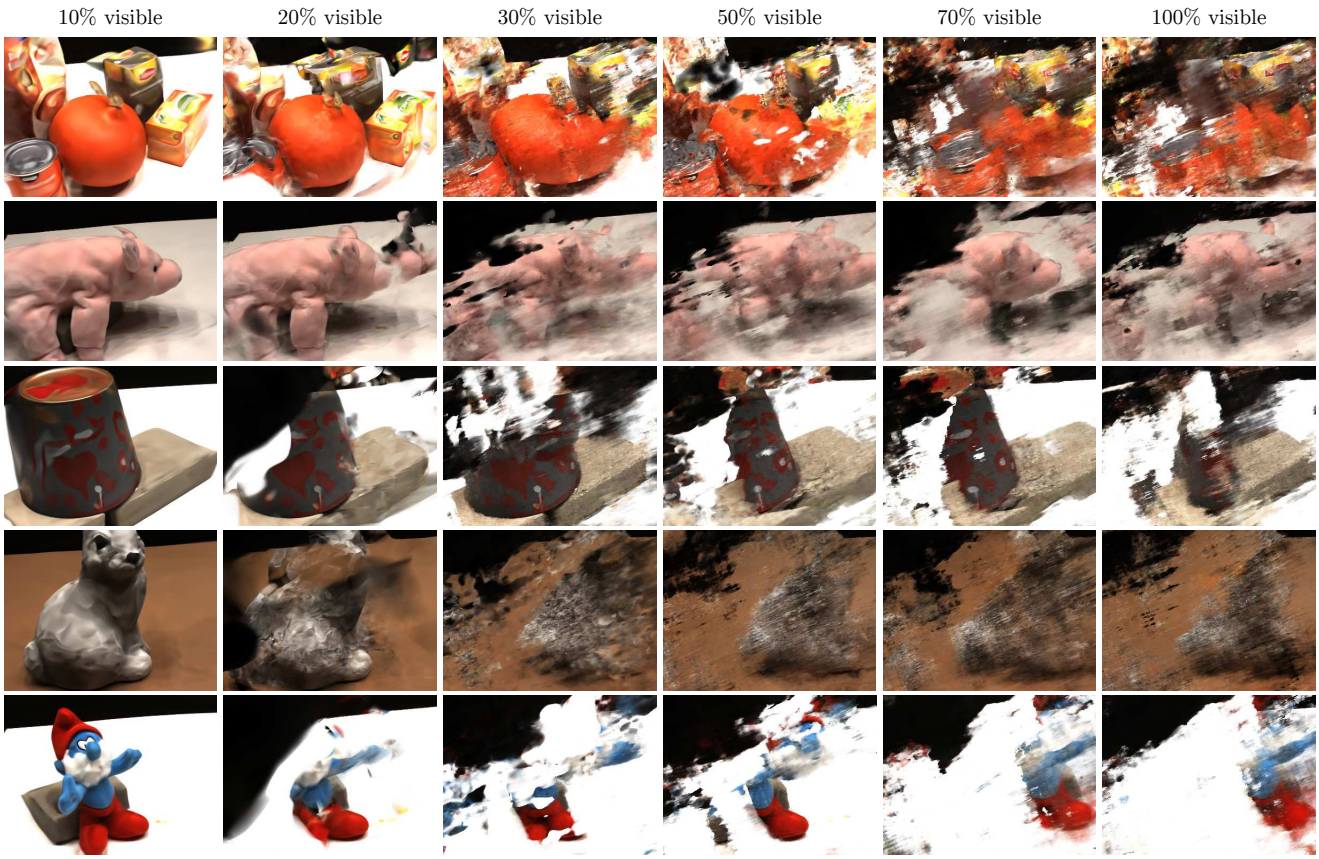


Figure A.1. **High-frequency inputs cause catastrophic failure in few-shot neural rendering.** We train mipNeRF [2] with masked (integrated) positional encoding by setting `pos_enc[int(L*x%]):]=0`, where L denotes the length of frequency bands (Eq. (1)) and x is the masking ratio. Using low-frequency components as inputs enables mipNeRF to learn meaningful scene representations despite their over-smoothness. Please refer to Figure 2 (in the main text) for numerical comparisons. We also provide animated visualizations on our project page.

In this supplement, we include additional quantitative and qualitative results to discuss more motivation and limitations of FreeNeRF in Appendix A. We also add details of experimental settings and implementations in Appendix B.

A. Additional Results

High-frequency inputs cause catastrophic failure. Figure A.1 shows more examples to demonstrate the failure mode revealed in Figure 2 that the high-frequency inputs lead to the catastrophic failure of few-shot neural rendering. When taking in 10% of the total embedding bits, mipNeRF can successfully reconstruct scenes despite their over-smoothness. However, with higher-frequency inputs, the scene reconstructions become more unrecognizable and collapse. This experimental finding lies at the heart of FreeNeRF: by restricting the inputs to the low-frequency components at the start of training, NeRF can start from significantly stabilized scene representations at the early stage of training. Upon these stable scene representations, NeRF continues refining the details when high-frequency signals become visible.

A.1. Limitations

In this subsection, we elaborate on the limitations and showcase the failure cases of FreeNeRF.

Trade-off between PSNR and LPIPS. Figure 7 studies the

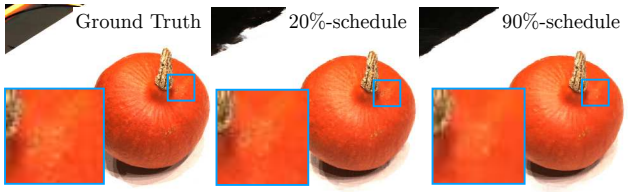


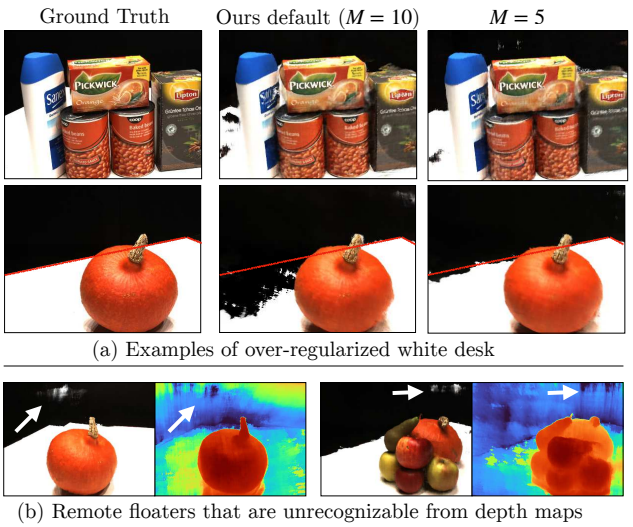
Figure A.2. **High-frequency details comparison.** We show the view synthesis results under the 9 input-view setting on the DTU dataset. With enough view information, a shorter frequency regularization enables NeRF models to render more high-frequency details.

effect of the duration of frequency regularization on PSNR and LPIPS. From the figure, we observe a trade-off between PSNR and LPIPS that a long-frequency curriculum usually results in a high PSNR score but a low LPIPS score. For example, under the 9 input-view setting, we obtain an object PSNR of 25.59 and an object LPIPS of 0.117 with a 90%-schedule and those of 25.38 and 0.096 with a 50%-schedule. Visually, when the number of input views is relatively sufficient (but still under few-shot settings), results under a shorter schedule usually present more high-frequency details (see the zoom-in patch in Fig. A.2). We thus use 70%-schedule and 50%-schedule for experiments under 6 and 9 input-view settings, respectively. We also found out that training FreeNeRF longer can obtain better LPIPS performance, *e.g.*, 0.182 to 0.167 and 0.308 to 0.290 for DTU-3 and LLFF-3 settings, respectively.

Limitations of L_{occ}. Over-regularization: our occlusion regularization can lead to an incomplete white desk on the DTU dataset due to over-regularization in some scenes, as shown in Figure A.3-(a). Reducing the regularization range of L_{occ} can ease this issue. A set of per-scene tuned hyperparameters can potentially provide better results. **Remote floaters:** Figure A.3-(b) shows some small cloudy floaters far from the camera. Our occlusion regularization that penalizes near-camera dense fields does not solve this problem. However, we do not observe these remote floaters in NeRF trained with only low-frequency inputs (10% visible). That said, though significantly regularized and stabilized, FreeNeRF still overfits to spurious occupancy to a certain degree. Better performance is excepted if FreeNeRF further exploits the low-frequency components to avoid such overfitting, leaving room for future work and improvements.

A.2. Depth Evaluation

Here we include results to compare the capability of different methods in depth estimation. As the datasets do not have actual ground truth depth, we utilized depth maps generated by mipNeRFs that were trained on all views as a substitute. FreeNeRF significantly improves its baseline, mipNeRF. RegNeRF, with its patch-based geometry regularization, achieves better performance on the object-centric DTU dataset, while FreeNeRF performs better on the scene-scale LLFF dataset without explicit geometry regularization. This experiment demonstrates the different features of FreeNeRF and RegNeRF, as well as the differences between DTU and LLFF datasets.



(a) Examples of over-regularized white desk

(b) Remote floaters that are unrecognizable from depth maps

Figure A.3. **Limitations of occlusion regularization.** (a) Aggressive occlusion regularization results in incomplete white desks that are visually annoying. Reducing the regularization range (from $M = 10$ to $M = 5$) can alleviate the issue to some extent. (b) Occlusion regularization does not solve remote floaters that are far from cameras.

Error=$\|D_{pseu} - D_{pred}\|$	DTU obj depth error↓			LLFF depth error↓		
# views	3	6	9	3	6	9
mipNeRF (baseline)	131.97	59.21	18.73	149.18	36.92	19.16
RegNeRF (explicit geo. reg.)	14.58	10.40	6.23	44.52	25.09	18.26
FreeNeRF	14.89	12.98	9.48	39.92	23.61	16.91

A.3. Additional Qualitative Results

Table A.1 provides more numeric results in addition to Table 2 on the DTU dataset. FreeNeRF achieves the best results under the "Average" metrics in most settings. However, we observe less improvement in terms of LPIPS. As we analyze in Appendix A.1, the slight blurriness introduced by FreeNeRF will result in a low LPIPS score. This is a limitation that could be addressed in the future.

A.4. Additional Visualizations

Blender. In Figure A.4, we show more qualitative comparisons between DietNeRF [11] and our FreeNeRF on the Blender dataset. From the zoom-in patches of DietNeRF's results, we see the generated patches are blurry and do not reflect the same distribution of style as that of ground truth. This is due to implicit semantics distillation behavior

	Setting	Object LPIPS ↓			Object Average ↓			Full-image LPIPS ↓			Full-image Average ↓		
		3-view	6-view	9-view	3-view	6-view	9-view	3-view	6-view	9-view	3-view	6-view	9-view
SRF [5]	Trained on DTU	0.304	0.250	0.232	0.171	0.132	0.120	0.482	0.401	0.359	0.207	0.162	0.145
PixelNeRF [37]		0.270	0.232	0.220	0.147	0.115	0.100	0.401	0.340	0.323	0.154	0.119	0.105
MVSNeRF [4]		0.197	0.156	0.135	0.113	0.088	0.068	0.385	0.321	0.280	0.184	0.146	0.114
SRF ft [5]	Trained on DTU and Optimized per Scene	0.281	0.225	0.205	0.162	0.114	0.093	0.431	0.353	0.325	0.196	0.143	0.125
PixelNeRF ft [37]		0.269	0.223	0.203	0.125	0.104	0.090	0.456	0.351	0.338	0.185	0.121	0.117
MVSNeRF ft [4]		0.197	0.155	0.135	0.113	0.089	0.069	0.384	0.319	0.278	0.185	0.146	0.113
mip-NeRF [2]	Optimized per Scene	0.353	0.198	0.092	0.323	0.148	0.056	0.655	0.394	0.209	0.485	0.231	0.098
DietNeRF [11]		0.314	0.201	0.173	0.243	0.101	0.068	0.574	0.336	0.277	0.383	0.149	0.098
RegNeRF [22]		0.190	0.117	0.089	0.112	0.071	0.047	0.341	0.233	0.184	0.189	0.118	0.079
mip-NeRF concat. (repro.)	Optimized per Scene	0.348	0.197	0.100	0.311	0.144	0.057	0.643	0.403	0.218	0.472	0.240	0.099
†RegNeRF concat. (repro.)		0.196	0.118	0.088	0.117	0.070	0.046	0.350	0.236	0.183	0.197	0.118	0.078
Our FreeNeRF		0.182	0.137	0.096	0.098	0.068	0.046	0.318	0.240	0.187	0.146	0.094	0.068

Table A.1. **Quantitative comparison on DTU.** We provide additional quantitative results to Table 2. Results in the bottom row are reproduced by us, and others come from [22]. "concat.": inputs concatenation (Eq. (2)). †ReNeRF: w/o. appearance regularization. The best, second-best, and third-best entries are marked in red, orange, and yellow, respectively.

done by DietNeRF. In contrast, our FreeNeRF reconstructs scenes closer to the ground truth.

DTU and LLFF. We provide more rendering results by FreeNeRF in Figures A.5 and A.6 under the 3 input-view setting on the DTU dataset and the LLFF dataset, respectively.

A.5. FreeNeRF for Normal Estimation

We briefly demonstrate a potential FreeNeRF's application beyond few-shot neural rendering. Specifically, we follow the similar settings in RefNeRF [32] to train a mipNeRF and a FreeNeRF on the "coffee" scene in the Shiny Blender dataset [32]. This dataset aims to benchmark NeRF's performance on glossy surfaces, where the key challenge is to estimate accurate normal vectors. Figure A.7 shows the comparison between mipNeRF and FreeNeRF. Compared to mipNeRF, FreeNeRF produces more accurate normal estimation and achieves much lower mean angular error (MAE) at no sacrifice of PSNR score. We conjecture that overfitting to high-frequency signals at the start of training is a very common issue in NeRF's training. However, such partial failure is veiled by good appearance results. We believe these partially degenerated results can be improved with frequency regularization, which makes NeRF's initial training more stable.

B. Experiment Details

We strictly follow the experimental settings in DietNeRF [11] and RegNeRF [22] to conduct our experiments. We provide some details in the following for completeness.

B.1. Dataset and metrics.

Blender Dataset. The Blender dataset [21] has 8 synthetic scenes in total. We follow the data split used in DietNeRF [11] to simulate a few-shot neural rendering scenario. For each scene, the training images with IDs (counting from "0") 26, 86, 2, 55, 75, 93, 16, 73, and 8 are used as the input views, and 25 images are sampled evenly from the testing images for evaluation. We follow [11] to use a $2\times$ downsampled resolution, resulting in 400×400 pixels for each image.

DTU Dataset. The DTU dataset [12] is a large-scale multi-view dataset that consists of 124 different scenes. PixelNeRF [37] uses a split of 88 training scenes and 15 test scenes to study the "pre-training & per-scene fine-tuning" setting in a few-shot neural rendering scenario. Different from theirs, our method does not require pre-training. We follow [22] to optimize NeRF models directly on the 15 test scenes. The test scan IDs are: 8, 21, 30, 31, 34, 38, 40, 41, 45, 55, 63, 82, 103, 110, and 114. In each scan, the images with the following IDs (counting from "0") are used as the input views: 25, 22, 28, 40, 44, 48, 0, 8, 13. The first 3 and 6 image IDs correspond to the input views in 3- and 6-view settings, respectively. The images with IDs in [1, 2, 9, 10, 11, 12, 14, 15, 23, 24, 26, 27, 29, 30, 31, 32, 33, 34, 35, 41, 42, 43, 45, 46, 47] serve as the novel views for evaluation. The remaining images are excluded due to wrong exposure. We follow [22, 37] to use a $4\times$ downsampled resolution, resulting in 300×400 pixels for each image.

LLFF Dataset. The LLFF dataset [20] is a forward-facing dataset that contains 8 scenes in total. Adhere to [21,22], we use every 8-th image as the novel views for evaluation, and evenly sample the input views across the remaining views. Images are downsampled $8\times$, resulting in 378×504 pixels for each image.

Metrics. To compute PSNR scores, we use the formula $-10 \cdot \log_{10}(\text{MSE})$ (assuming the maximum pixel value is 1). Additionally, we utilize the scikit-image's API[3] to com-

[3] https://scikit-image.org/docs/stable/auto_examples/

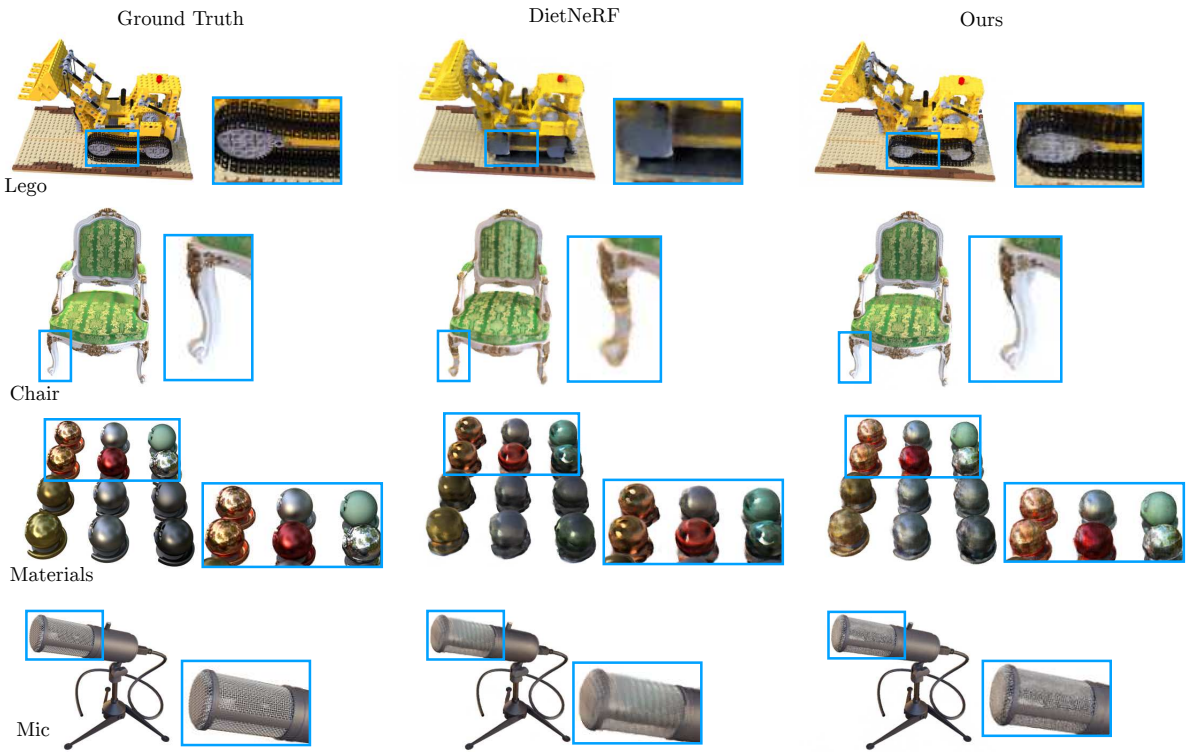


Figure A.4. **Qualitative comparison on the Blender dataset.** DietNeRF generates patches that can be reasonable and plausible to some extent but do not closely match the ground truth. This is a limitation of using a pre-trained model for semantic regularization. In contrast, our FreeNeRF reconstructs scenes that are more in line with the ground truth.

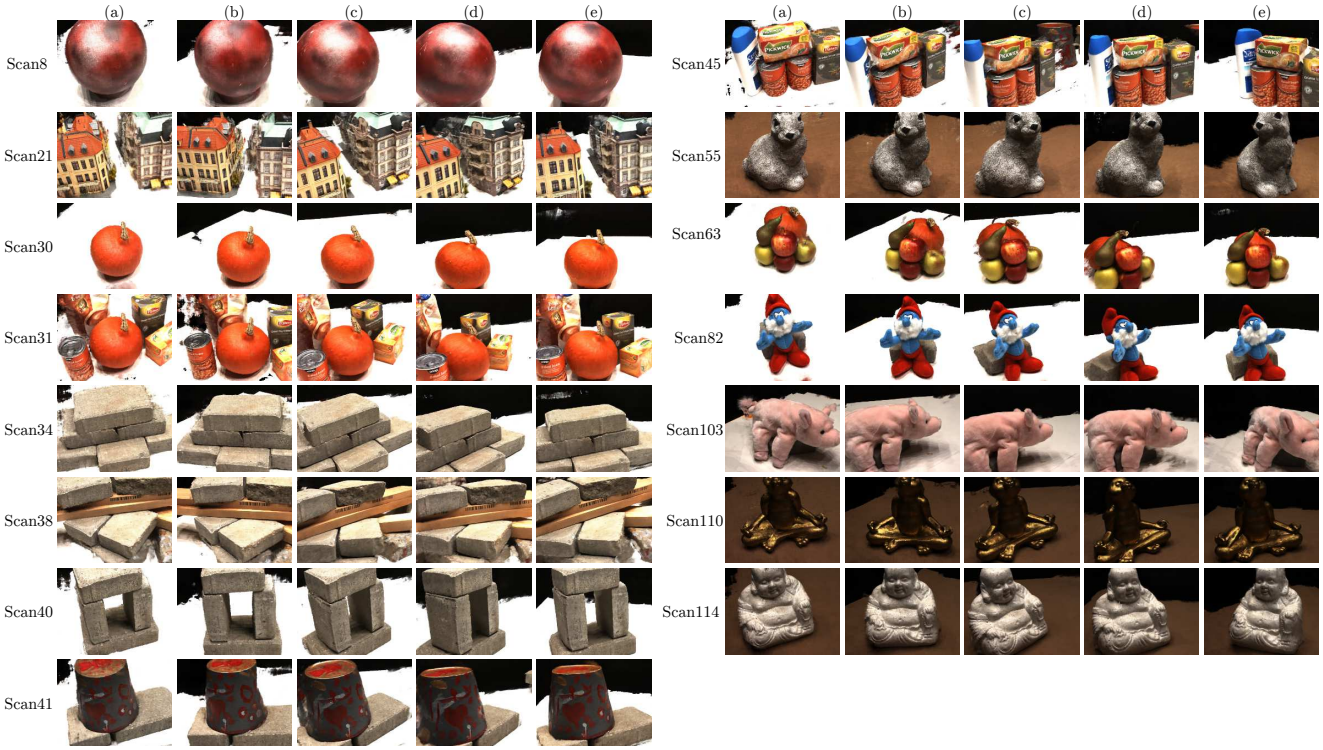


Figure A.5. **Example FreeNeRF's novel view synthesis results with 3 input views on the DTU dataset.**

transform/plot_ssim.html

Figure A.6. **Example FreeNeRF's novel view synthesis results with 3 input views on the LLFF dataset.**

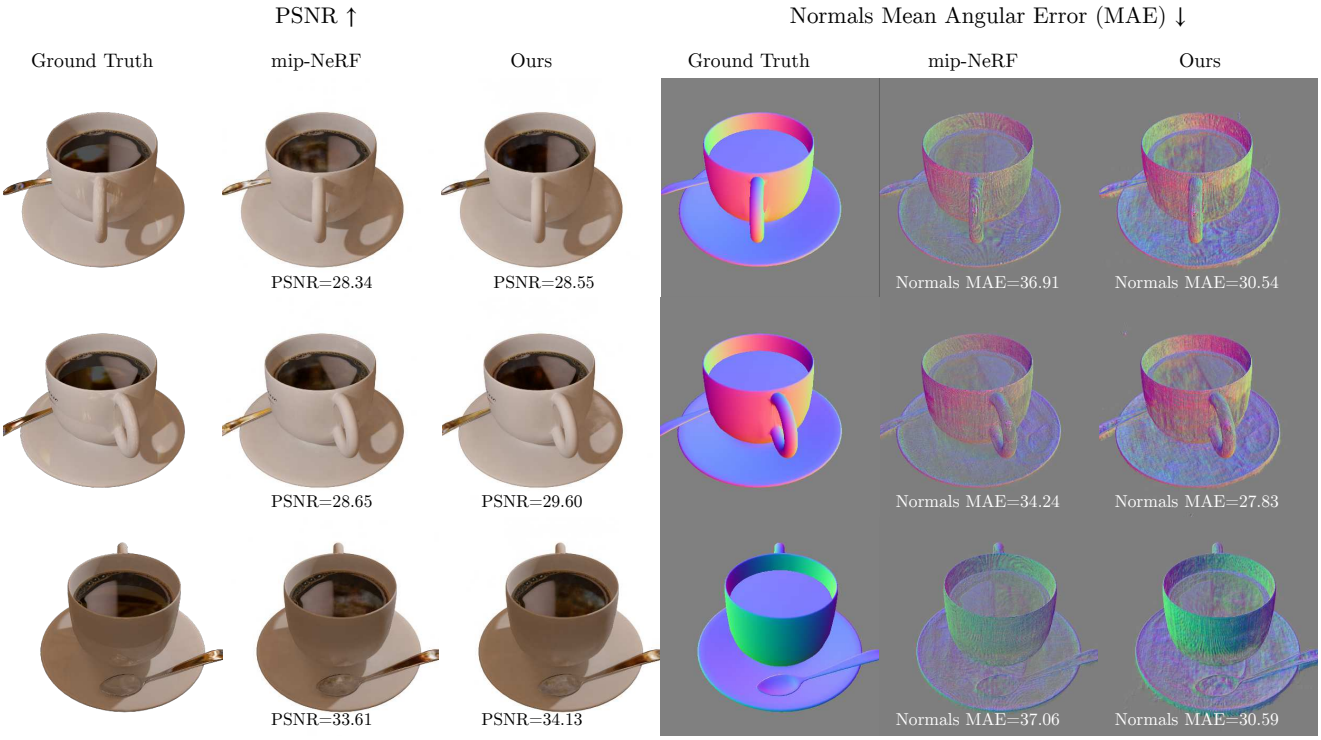


Figure A.7. **Comparison on normal vectors estimation.** Beyond the few-shot neural rendering problem, we train a mipNeRF and a FreeNeRF on the "coffee" scene in the Shiny Blender dataset [32] to demonstrate FreeNeRF's potential in estimating more accurate normal vectors. The PSNR scores for this scene are 30.839 and 31.364 for mipNeRF and FreeNeRF, respectively. The mean angular errors (the lower, the better) are 36.549 and 31.492 for mipNeRF and FreeNeRF, respectively. Note that we use a much smaller batch size (4096) than that in the original setting (16394), so the numerical results here are not comparable to those in RefNeRF [32].

pute the structural similarity index measure (SSIM) score and the interface provided by an open source repository[4] (using a learned VGG model) to compute the learned perceptual image patch similarity (LPIPS) score.

B.2. Implementations.

DietNeRF's codebase. In this codebase[5], a plain NeRF [21] that consists of two MLPs (one coarse MLP and one fine MLP) is used as the baseline. All NeRF models are trained with the Adam optimizer for 200k iterations. The learning rate starts at 5×10^{-4} and decays exponentially with a rate of 0.1. We refer readers to the codebase for more details. In this codebase, the maximum input frequency L (Eq. (1)) used in the position encoding for coordinates is 9. The original coordinates are concatenated with positional encodings by default.

RegNeRF's codebase. In this codebase[6], a plain mipNeRF [2] is used as the baseline. The maximum input frequency of coordinates is 16, which is larger than that of the original NeRF [21]. We further concatenate the original coordinates into the positional encodings. All NeRF models are trained with the Adam optimizer with an exponential learning rate decaying from $2 \cdot 10^{-3}$ to $2 \cdot 10^{-5}$ and 512 warm-up steps with a multiplier of 0.01 [2]. Following [22], we clip gradients by value at 0.1 and then by norm at 0.1 for all experiments. All NeRF models in the main experiments are optimized for 500 epochs with a batch size of 4096. This setting results in around 44k, 88k and 132k training iterations on the DTU dataset for 3/6/9 input views, respectively, and 70k, 140k and 210k training iterations for those on the LLFF dataset, respectively. Note that in the ablation study we use a batch size of 1024 instead of 4096 for faster training.

Occlusion regularization. We use a weight of 0.01 for L_{occ} in all experiments. For simplicity, we compute this loss on the secondary stage's outputs, *i.e*. those from the fine MLP in NeRF [21] and the second query in mipNeRF [2].

[4] https://github.com/richzhang/PerceptualSimilarity

[5] https://github.com/ajayjain/DietNeRF

[6] https://github.com/google-research/google-research/tree/master/regnerf