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

Jiawei Yang
UC, Los Angeles
jiawei118@ucla.edu

Marco Pavone
Nvidia Research, Stanford University
pavone@stanford.edu

Yue Wang
Nvidia Research
yuewang@nvidia.com

Turning the left to the right by adding ***one*** line of code: `pos_enc[int(t/T*L)+3:]=0`

Figure 1. **Example novel view synthesis results from sparse inputs.** The only difference between NeRF (left) and FreeNeRF (right) is the use of our frequency regularization, which can be implemented as few as, approximately, *one* line of code (bottom, where t and T denote the current training iteration and regularization duration, respectively; L is the length of the input positional encoding).

Abstract

Novel view synthesis with sparse inputs is a challenging problem for neural radiance fields (NeRF). Recent efforts alleviate this challenge by introducing external supervision, such as pre-trained models and extra depth signals, or by using non-trivial patch-based rendering. In this paper, we present ***Fre****quency regularized* ***NeRF*** *(FreeNeRF), a surprisingly simple baseline that outperforms previous methods with minimal modifications to plain NeRF. We analyze the key challenges in few-shot neural rendering and find that frequency plays an important role in NeRF's training. Based on this analysis, we propose two regularization terms: one to regularize the frequency range of NeRF's inputs, and the other to penalize the near-camera density fields. Both techniques are "free lunches" that come at no additional computational cost. We demonstrate that even with just one line of code change, the original NeRF can achieve similar performance to other complicated methods in the few-shot setting. FreeNeRF achieves state-of-the-art performance across diverse datasets, including Blender, DTU, and LLFF. We hope that this simple baseline will motivate a rethinking of the fundamental role of frequency in NeRF's training, under both the low-data regime and beyond. This project is released at FreeNeRF.*

1. Introduction

Neural Radiance Field (NeRF) [21] has gained tremendous attention in 3D computer vision and computer graphics due to its ability to render high-fidelity novel views. However, NeRF is prone to overfitting to training views and struggles with novel view synthesis when only a few inputs are available. We term this view synthesis from sparse inputs problem as a few-shot neural rendering problem.

Existing methods address this challenge using different strategies. Transfer learning methods, *e.g.*, PixelNerf [37] and MVSNeRF [4], pre-train on large-scale curated multi-view datasets and further incorporate per-scene optimization at test time. Depth-supervised methods [6, 29] introduce estimated depth as an external supervisory signal, leading to a complex training pipeline. Patch-based regularization methods impose regularization from different sources on rendered patches, *e.g.*, semantic consistency regularization [11], geometry regularization [22, 8], and appearance regularization [22], all at the cost of computation overhead since an additional, non-trivial number of patches must be rendered during training [11, 22, 8].

In this work, we find that a plain NeRF can work surprisingly well with *none* of the above strategies in the few-shot setting by adding (approximately) as few as *one* line of code (see Fig. 1). Concretely, we analyze the common failure modes in training NeRF under a low-data regime. Drawing on this analysis, we propose two regularization terms. One is frequency regularization, which directly regularizes the visible frequency bands of NeRF's inputs to stabilize the learning process and avoid catastrophic overfitting at the start of training. The other is occlusion regularization, which penalizes the near-camera density fields that cause "floaters," another failure mode in the few-shot neural rendering problem. Combined, we call our method **Fre**que**n**cy regularized **NeRF** (FreeNeRF), which is "free" in two ways. First, it is dependency-free because it requires neither costly pre-training [37, 4, 11, 22] nor extra supervisory signals [6, 29]. Second, it is overhead-free as it requires no additional training-time rendering for patch-based regularization [11, 22, 8].

We consider FreeNeRF a simple baseline (with minimal modifications to a plain NeRF) in the few-shot neural rendering problem, although it already outperforms existing state-of-the-art methods on multiple datasets, including Blender, DTU, and LLFF, at almost no additional computation cost. Our contributions can be summarized as follows:

- We reveal the link between the failure of few-shot neural rendering and the frequency of positional encoding, which is further verified by an empirical study and addressed by our proposed method. To our knowledge, our method is the first attempt to address few-shot neural rendering from a frequency perspective.
- We identify another common failure pattern in learning NeRF from sparse inputs and alleviate it with a new occlusion regularizer. This regularizer effectively improves performance and generalizes across datasets.
- Combined, we introduce a simple baseline, FreeNeRF, that can be implemented with a few lines of code modification while outperforming previous state-of-the-art methods. Our method is dependency-free and overhead-free, making it a practical and efficient solution to this problem.

We hope the observations and discussions in this paper will motivate people to rethink the fundamental role of frequency in NeRF's positional encoding.

2. Related Work

Neural fields. Neural fields [36] use deep neural networks to represent 2D images or 3D scenes as continuous functions. The seminal work, Neural Radiance Fields (NeRF) [21], has been widely studied and advanced in a variety of applications [2, 3, 32, 19, 23, 13, 25], including novel view synthesis [21, 18], 3D generation [25, 10], deformation [23, 26, 28], video [15, 35, 7, 24, 14]. Despite tremendous progress, NeRF still requires hundreds of input images to learn high-quality scene representations; it fails to synthesize novel views with a few input views, *e.g.*, 3, 6, and 9 views, limiting its potential applications in the real world.

Few-shot Neural Rendering. Many works have attempted to address the challenging few-shot neural rendering problem by leveraging extra information. For instance, external models can be used to acquire normalization-flow regularization [22], perceptual regularization [38], depth supervision [29, 6, 34], and cross-view semantic consistency [11]. Another thread of works [5, 37, 4] attempts to learn transferable models by training on a large, curated dataset instead of using an external model. Recent works argue that geometry is the most important factor in few-shot neural rendering and propose geometry regularization [22, 1, 8] for better performance. However, these methods require expensive pre-training on tailored multi-view datasets [5, 37, 4] or costly training-time patch rendering [11, 22, 1, 8], introducing significant overhead in methodology, engineering implementation, and training budgets. In this work, we show that a plain NeRF can work surprisingly well with minimal modifications (a few lines of code) by incorporating our frequency regularization and occlusion regularization. Unlike most previous methods, our approach maintains the same computational efficiency as the original NeRF.

Frequency in neural representations. Positional encoding lies at the heart of NeRF's success [21, 31]. Previous studies [31, 30] have shown that neural networks often struggle to learn high-frequency functions from low-dimensional inputs. Encoding inputs with sinusoidal functions of different frequencies can alleviate this issue. Recent works show the benefits of gradually increasing the input frequency in different applications, such as non-rigid scene deformation [23], bundle adjustment [16], surface reconstruction [33], and fitting functions with a wider frequency band [9]. Our work leverages frequency curriculum to tackle the few-shot neural rendering problem. Notably, our approach not only demonstrates the surprising effectiveness of frequency regularization in learning from sparse inputs, but also reveals

the failure modes behind this problem and why frequency regularization helps.

3. Method

3.1. Preliminaries

Neural radiance fields. A neural radiance field (NeRF) [21] uses a multi-layer perceptron (MLP) to represent a scene as a volumetric density field σ and associated RGB values $\mathbf{c}$ at each point in the scene. It takes as input a 3D coordinate $\mathbf{x} \in \mathbb{R}^3$ and a viewing directional unit vector $\mathbf{d} \in \mathbb{S}^2$, and outputs the corresponding density and color. In its most basic form, NeRF learns a continuous function $f_\theta(\mathbf{x}, \mathbf{d}) = (\sigma, \mathbf{c})$ where θ denotes MLP parameters.

Positional encoding. Directly optimizing NeRF over raw inputs $(\mathbf{x}, \mathbf{d})$ often leads to difficulties in synthesizing high-frequency details [31, 21]. To address this issue, recent work has used sinusoidal functions with different frequencies to map the inputs into a higher-dimensional space [21]:

$$\gamma_L(\mathbf{x}) = \left[\sin(\mathbf{x}), \cos(\mathbf{x}), ..., \sin(2^{L-1}\mathbf{x}), \cos(2^{L-1}\mathbf{x})\right], \tag{1}$$

where L is a hyperparameter that controls the maximum encoded frequency and may differ for coordinates $\mathbf{x}$ and directional vectors $\mathbf{d}$. A common practice is to concatenate the raw inputs with the frequency-encoded inputs as follows:

$$\mathbf{x}' = [\mathbf{x}, \gamma_L(\mathbf{x})] \tag{2}$$

This concatenation is applied to both coordinate inputs and view direction inputs.

Rendering. To render a pixel in NeRF, a ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ is cast from the camera's origin $\mathbf{o}$ along the direction $\mathbf{d}$ to pass through the pixel, where t is the distance to the origin. Within the near and far bounds $[t_{\text{near}}, t_{\text{far}}]$ of the cast ray, NeRF computes the color of that ray using the quadrature of K sampled points $\mathbf{t}_K = \{t_1, \ldots, t_K\}$:

$$\hat{\mathbf{c}}(\mathbf{r}; \theta, \mathbf{t}_K) = \sum_K T_k(1 - \exp(-\sigma_k(t_{k+1} - t_k)))\mathbf{c_k},$$
$$\text{with} \quad T_k = \exp\left(-\sum_{k'<k} \sigma'_k\,(t_{k'+1} - t_{k'})\right), \tag{3}$$

where $\hat{\mathbf{c}}(\mathbf{r}; \theta, \mathbf{t}_K)$ is the final integrated color. Note that the sampled points $\mathbf{t}_K$ are in a near-to-far order, *i.e.*, a point with a smaller index k is closer to the camera's origin.

3.2. Frequency Regularization

The most common failure mode of few-shot neural rendering is overfitting. NeRF learns 3D scene representations from a set of 2D images without explicit 3D geometry. 3D geometry is implicitly learned by optimizing appearance in its 2D projected views. However, given only a few input views, NeRF is prone to overfitting to these 2D images with small loss while not explaining 3D geometry in a multi-view consistent way. Synthesizing novel views from such models leads to systematic failure. As shown on the left of Figure 1, no NeRF model can successfully recover the scene geometry when synthesizing novel views.

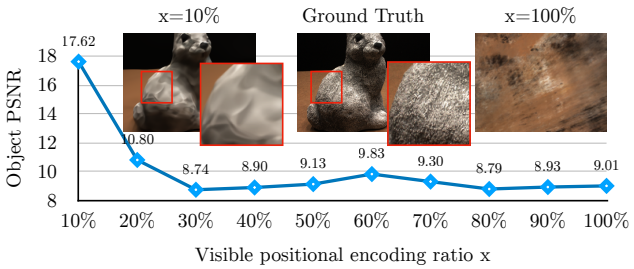


Figure 2. **Masking high-frequency inputs helps few-shot neural rendering.** We investigate how NeRF performs with positional encodings under different masking ratios on the DTU dataset using 3 input views. Despite its over-smoothness, the plain NeRF succeeds in the few-shot setting when only low-frequency inputs are visible.

The overfitting issue in few-shot neural rendering is presumably exacerbated by high-frequency inputs. [31] shows that higher-frequency mappings enable faster convergence for high-frequency components. However, the over-fast convergence on high-frequency impedes NeRF from exploring low-frequency information and significantly biases NeRF towards undesired high-frequency artifacts (horns and room examples in Fig. 1). In the few-shot scenario, NeRF is even more sensitive to susceptible noise as there are fewer images to learn coherent geometry. Thus, we hypothesize that high-frequency components are a major cause of the failure modes observed in few-shot neural rendering. We provide empirical evidence below.

We investigate how a plain NeRF performs when inputs are encoded by different numbers of frequency bands. To achieve this, we train mipNeRF [2] using masked (integrated) positional encoding. Specifically, we set `pos_enc[int(L*x%]):]=0`, where L denotes the length of frequency encoded coordinates after the positional encoding (Eq. (1)), and x is the visible ratio. We briefly demonstrate our observation here and defer the experiment details to §4.1. Figure 2 shows the results for the DTU dataset under the 3 input-view setting. As anticipated, we observe a significant drop in mipNeRF's performance as higher-frequency inputs are presented to the model. When 10% of total embedding bits are used, mipNeRF achieves a high PSNR of 17.62, while the plain mipNeRF achieves only 9.01 PSNR on its own (at 100% visible ratio). The *only* difference between these two models is whether masked positional encodings are used. Although removing a significant portion of high-frequency components avoids catas-

trophic failure at the start of training, it does not result in competitive scene representations, as the rendered images are usually oversmoothed (as seen in Fig. 2 zoom-in patches). Nonetheless, it is noteworthy that in few-shot scenarios, models using low-frequency inputs may produce significantly better representations than those using high-frequency inputs.

Building on this empirical finding, we propose a frequency regularization method. Given a positional encoding of length $L+3$ (Eq. (2)), we use a linearly increasing frequency mask $\boldsymbol{\alpha}$ to regulate the visible frequency spectrum based on the training time steps, as follows:

$$\gamma'_L(t, T; \mathbf{x}) = \gamma_L(\mathbf{x}) \odot \boldsymbol{\alpha}(t, T, L), \tag{4}$$

$$\text{with } \boldsymbol{\alpha}_i(t,T,L) = \begin{cases} 1 & \text{if } i \leq \frac{t \cdot L}{T} + 3 \\ \frac{t \cdot L}{T} - \lfloor \frac{t \cdot L}{T} \rfloor & \text{if } \frac{t \cdot L}{T} + 3 < i \leq \frac{t \cdot L}{T} + 6 \\ 0 & \text{if } i > \frac{t \cdot L}{T} + 6 \end{cases} \tag{5}$$

where $\boldsymbol{\alpha}_i(t, T, L)$ denotes the i-th bit value of $\boldsymbol{\alpha}(t, T, L)$; t and T are the current training iteration and the final iteration of frequency regularization, respectively. Concretely, we start with raw inputs without positional encoding and linearly increase the visible frequency by 3-bit each time as training progresses. This schedule can also be simplified as one line of code, as shown in Figure 1. Our frequency regularization circumvents the unstable and susceptible high-frequency signals at the beginning of training and gradually provides NeRF high-frequency information to avoid over-smoothness.

We note that our frequency regularization shares some similarities with the coarse-to-fine frequency schedules used in other works [23, 16]. Different from theirs, our work focuses on the few-shot neural rendering problem and reveals the catastrophic failure patterns caused by high-frequency inputs and their implication to this problem.

3.3. Occlusion Regularization

Frequency regularization does not solve all problems in few-shot neural rendering. Due to the limited number of training views and the ill-posed nature of the problem, certain characteristic artifacts may still exist in novel views. These failure modes often manifest as "walls" or "floaters" that are located extremely close to the camera, as seen in the bottom of Figure 3. Such artifacts can still be observed even with a sufficient number of training views [3]. To address these issues, [3] proposed a distortion loss. However, our experiments show that this regularization does not help in the few-shot setting and may even exacerbate the issue.

We find most of these failure patterns originate from the least overlapped regions in the training views. Figure 3 shows an example of 3 training views and 2 novel views with "white walls". We manually annotate the least overlapped regions in the training views for demonstration ((a) and (b) in Fig. 3). These regions are difficult to estimate in terms of geometry due to the extremely limited information available (one-shot). Consequently, a NeRF model would interpret these unexplored areas as dense volumetric floaters located near the camera. We suspect that the floaters observed in [3] also come from these least overlapped regions.

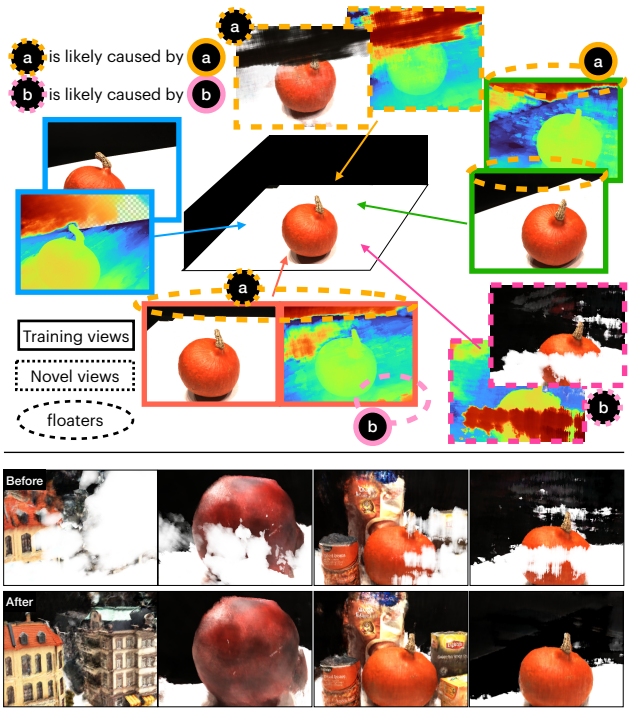


Figure 3. **Illustration of occlusion regularization.** We show 3 training views (solid rectangles) and 2 novel views (dashed rectangles) rendered by a frequency-regularized NeRF. The floaters in the novel views appear to be *near-camera* dense fields in the *training* views (dashed circles) so that we can penalize them directly without the need for the costly novel-view rendering in [11, 22].

As discussed above, the presence of floaters and walls in novel views is caused by the imperfect training views, and thus can be addressed directly at training time without the need for novel-pose sampling [22, 11, 37]. To this end, we propose a simple yet effective "occlusion" regularization that penalizes the dense fields near the camera. We define:

$$\mathcal{L}_{occ} = \frac{\boldsymbol{\sigma}_K^\intercal \cdot \mathbf{m}_K}{K} = \frac{1}{K} \sum_K \sigma_k \cdot m_k, \tag{6}$$

where $\mathbf{m}_k$ is a binary mask vector that determines whether a point will be penalized, and $\boldsymbol{\sigma}_K$ denotes the density values of the K points sampled along the ray in the order of proximity to the origin (near to far). To reduce solid floaters near the camera, we set the values of $\mathbf{m}_k$ up to index M, termed as regularization range, to 1 and the rest to 0. The occlusion regularization loss is easy to implement and compute.

4. Experiments

4.1. Setups

Datasets & metrics. We evaluate our method on three datasets under few-shot settings: the NeRF Blender Synthetic dataset (Blender) [21], the DTU dataset [12], and the LLFF dataset [20]. For Blender, we follow DietNeRF [11] to train on 8 views and test on 25 test images. For DTU and LLFF, we adhere to RegNeRF's [22] protocol. On DTU, we use objects' masks to remove the background when computing metrics, as full-image evaluation is biased towards the background, as reported by [37, 22]. We report PSNR, SSIM, and LPIPS scores as quantitative results. We also report the geometric mean of MSE $= 10^{-\text{PSNR}/10}$, $\sqrt{1-\text{SSIM}}$, and LPIPS, following [22]. More details on the experimental setup can be found in the appendix.

Implementations. Our FreeNeRF can directly improve NeRF [21] and mipNeRF [2]. To demonstrate this, we use DietNeRF's codebase[1] for NeRF on the Blender dataset and RegNeRF's codebase[2] for mipNeRF on the DTU dataset and the LLFF dataset. We disable the proposed components in those papers and implement our two regularization terms on top of their baselines. We make one modification to mipNeRF [2], which is to concatenate positional encodings with the original Euclidean coordinates (Eq. (2)). This is a default step in NeRF but not in mipNeRF, and it helps unify our experiments' initial visible frequency range. We follow their training schedules for optimization. Please refer to the Appendix for full training recipes.

Hyper-parameters. We set the end iteration of frequency regularization as $T = \lfloor 90\% * \text{total\_iters} \rfloor$ for the 3-view setting and 70% for the 6-view setting and 20% for the 9-view setting. We regularize both coordinates $\mathbf{x}$ and view directions $\mathbf{d}$. For $\mathcal{L}_{occ}$, we use a weight of 0.01 in all experiments and set the regularization range $M = 20$ for LLFF and Blender and $M = 10$ for DTU. For DTU in particular, we find that the "walls" are mostly caused by the white desk and black background, so we use this information to penalize more points in a slightly wider range ($M = 15$) if their colors are black or white.

Comparing methods. Unless otherwise specified, we directly use the results reported in DietNeRF [11] and RegNeRF [22] for comparisons, as our method is implemented using their codebases. We also include our reproduced results for reference.

4.2. Comparison

We compare with state-of-the-art methods in terms of novel view synthesis quality and computation overhead. We show that FreeNeRF outperforms others in synthesis quality while maintaining a much lower cost.

Method	PSNR ↑	SSIM ↑	LPIPS ↓
NeRF [21]	14.934	0.687	0.318
NV [17]	17.859	0.741	0.245
Simplified NeRF [11]	20.092	0.822	0.179
DietNeRF [11]	23.147	0.866	0.109
DietNeRF + $\mathcal{L}_{\text{MSE}}$ ft $50k$	23.591	0.874	0.097
NeRF (repro.)	13.931	0.689	0.320
DietNeRF (repro.)	22.503	0.823	0.124
Our FreeNeRF	24.259	0.883	0.098

Table 1. **Quantitative comparison on Blender.** "$\mathcal{L}_{\text{MSE}}$ ft $50k$": fine-tune for another $50k$ iterations with $\mathcal{L}_{\text{MSE}}$. The top row section includes results from [11], while the bottom row section shows our reproduced results (repro.). Gray: our baseline. Red, orange, and yellow: the best, second-best, and third-best.

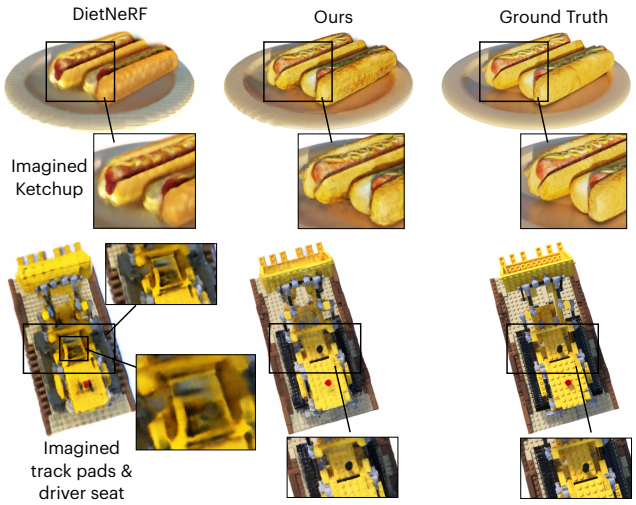


Figure 4. **Novel view synthesis examples on Blender.** Our results are qualitatively better than DietNeRF's. DietNeRF renders "imaginary" components that do not exist in the original images.

Blender dataset. Table 1 shows the image synthesis metrics on the Blender dataset [21]. Our approach outperforms all other methods in the PSNR and SSIM scores, with a comparable LPIPS score to the best one. The improved DietNeRF with fine-tuning still underperforms ours. Note that our direct baseline is "NeRF (repro.)" as we do not use any techniques from DietNeRF [11]. Figure 4 shows two examples for qualitative comparison (see Fig. 1 for plain NeRF's results). Interestingly, we observe that DietNeRF implicitly distills semantic information from a pre-trained CLIP model [27] into NeRF, which leads to unrealistic and "imaginary" patches that do not exist in the original scenes, such as "ketchup" in the hotdog and rubber-like track-pads in the bulldozer. This behavior is highly correlated to feature distillation [13] and recent developments in 3D object generation that combine NeRF with large pre-trained vision-language models [10, 25]. Although this potentially

[1] https://github.com/ajayjain/DietNeRF
[2] https://github.com/google-research/google-research/tree/master/regnerf

	Setting	Object PSNR ↑			Object SSIM ↑			Full-image PSNR ↑			Full-image SSIM ↑		
		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	15.32	17.54	18.35	0.671	0.730	0.752	15.84	17.77	18.56	0.532	0.616	0.652
PixelNeRF [37]		16.82	19.11	20.40	0.695	0.745	0.768	18.74	21.02	22.23	0.618	0.684	0.714
MVSNeRF [4]		18.63	20.70	22.40	0.769	0.823	0.853	16.33	18.26	20.32	0.602	0.695	0.735
SRF ft [5]	Trained on DTU and Optimized per Scene	15.68	18.87	20.75	0.698	0.757	0.785	16.06	18.69	19.97	0.550	0.657	0.678
PixelNeRF ft [37]		18.95	20.56	21.83	0.710	0.753	0.781	17.38	21.52	21.67	0.548	0.670	9.680
MVSNeRF ft [4]		18.54	20.49	22.22	0.769	0.822	0.853	16.26	18.22	20.32	0.601	0.694	0.736
mip-NeRF [2]	Optimized per Scene	8.68	16.54	23.58	0.571	0.741	0.879	7.64	14.33	20.71	0.227	0.568	0.799
DietNeRF [11]		11.85	20.63	23.83	0.633	0.778	0.823	10.01	18.70	22.16	0.354	0.668	0.740
RegNeRF [22]		18.89	22.20	24.93	0.745	0.841	0.884	15.33	19.10	22.30	0.621	0.757	0.823
mip-NeRF concat. (repro.)	Optimized per Scene	9.10	16.84	23.56	0.578	0.754	0.877	7.94	14.15	20.97	0.235	0.560	0.794
†RegNeRF concat. (repro.)		18.50	22.18	24.88	0.744	0.844	0.890	15.00	19.12	22.41	0.606	0.754	0.826
Our FreeNeRF		19.92	23.25	25.38	0.787	0.844	0.888	18.02	22.39	24.2	0.680	0.779	0.833

Table 2. **Quantitative comparison on DTU.** We present the PSNR and SSIM scores of foreground objects and full images. Our FreeNeRF synthesizes better foreground objects and full images than most of the others. Our direct baseline is mipNeRF [2] (marked in gray). Results in the bottom row section are our reproductions, 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.

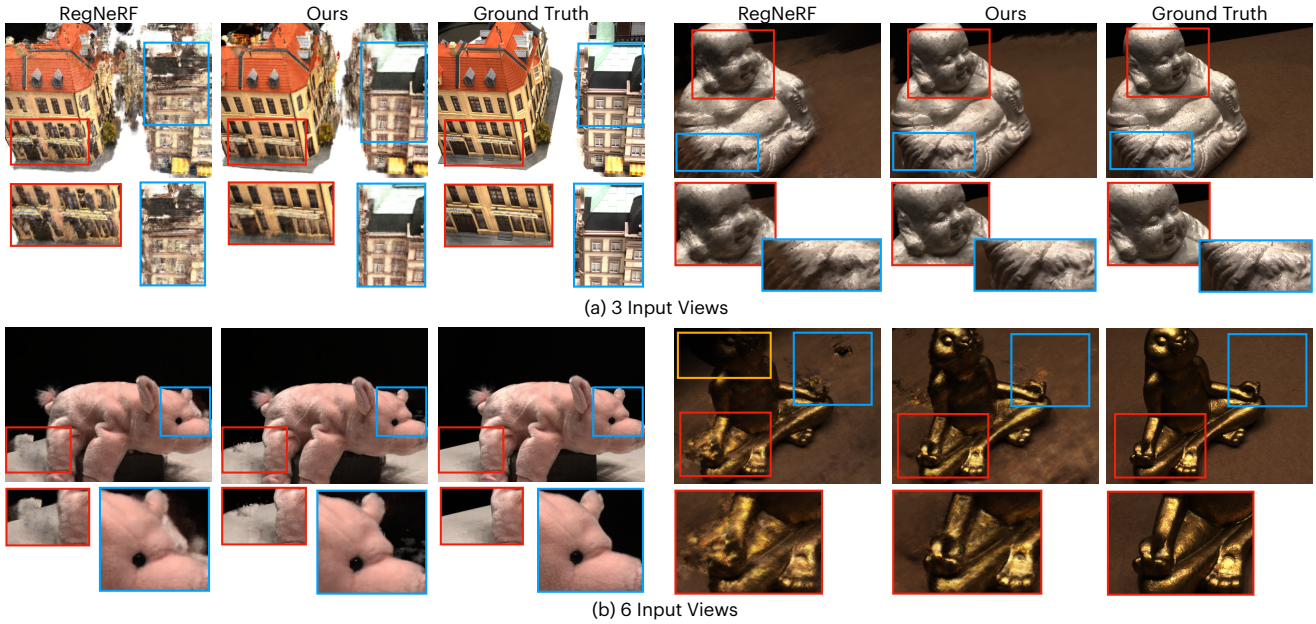


Figure 5. **Qualitative comparison on DTU.** We show novel views rendered by RegNeRF and ours in 3 and 6 input-view settings. For the Buddha example, the piece-wise geometry regularization used by RegNeRF [22] hurts the fine-grained geometry, erasing the details of eyes, fingers and wrinkles. RegNeRF's results are rendered by our reproduced †RegNeRF concat. (*c.f.* Tab. 2).

could be an interesting application, such behavior is undesired in our task and will hamper outputs' fidelity. In contrast, our method does not require semantics regularization while achieving better performance.

DTU dataset. Table 2 shows the quantitative results on the DTU dataset. Transfer learning-based methods that require expensive pre-training (SRF [5], PixelNeRF[37], and MVSNeRF [4]) underperform ours in almost all settings, except the full-image PSNR score under 3-view setting. This may be due to the bias introduced by the white table and black background present in many scenes in the DTU dataset, which can be learned as a prior through pre-training. Compared to per-scene optimization methods (mipNeRF [2], DietNeRF [11], and RegNeRF [22]), our approach achieves the best results. Figure 5 shows example novel views rendered by RegNeRF and ours. In the Buddha scene, for instance, piece-wise smoothness imposed by RegNeRF's geometry regularization [22] leads to the loss of fine-grained details, such as eyes, fingers, and wrinkles. In contrast, our frequency regularization, which can be seen as an implicit geometry regularization, forces smooth geometry at the beginning (due to the limited frequency spectrum) and gradually relaxes the constraint to facilitate the details. In the more challenging scenes (*e.g.*, buildings and the bronze statue in Fig. 5), FreeNeRF produces higher-quality results.

LLFF dataset. Table 3 and Figure 6 show quantitative and

	Setting	PSNR ↑			SSIM ↑			LPIPS ↓			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	12.34	13.10	13.00	0.250	0.293	0.297	0.591	0.594	0.605	0.313	0.293	0.296
PixelNeRF [37]		7.93	8.74	8.61	0.272	0.280	0.274	0.682	0.676	0.665	0.461	0.433	0.432
MVSNeRF [4]		17.25	19.79	20.47	0.557	0.656	0.689	0.356	0.269	0.242	0.171	0.125	0.111
SRF ft [5]	Trained on DTU and Optimized per Scene	17.07	16.75	17.39	0.436	0.438	0.465	0.529	0.521	0.503	0.203	0.207	0.193
PixelNeRF ft [37]		16.17	17.03	18.92	0.438	0.473	0.535	0.512	0.477	0.430	0.217	0.196	0.163
MVSNeRF ft [4]		17.88	19.99	20.47	0.584	0.660	0.695	0.327	0.264	0.244	0.157	0.122	0.111
mip-NeRF [2]	Optimized per Scene	14.62	20.87	24.26	0.351	0.692	0.805	0.495	0.255	0.172	0.246	0.114	0.073
DietNeRF [11]		14.94	21.75	24.28	0.370	0.717	0.801	0.496	0.248	0.183	0.240	0.105	0.073
RegNeRF [22]		19.08	23.10	24.86	0.587	0.760	0.820	0.336	0.206	0.161	0.149	0.086	0.067
mip-NeRF concat. (repro.)	Optimized per Scene	16.11	22.91	24.88	0.401	0.756	0.826	0.460	0.213	0.160	0.215	0.090	0.066
†RegNeRF concat. (repro.)		18.84	23.22	24.88	0.573	0.770	0.826	0.345	0.203	0.159	0.150	0.085	0.065
Our FreeNeRF		19.63	23.73	25.13	0.612	0.779	0.827	0.308	0.195	0.160	0.134	0.075	0.064

Table 3. **Quantitative comparison on LLFF.** Our FreeNeRF achieves the best results in most metrics under different input-view settings. Our direct baseline is mipNeRF [2] (marked in gray). Results in the bottom row section are our reproductions, 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.

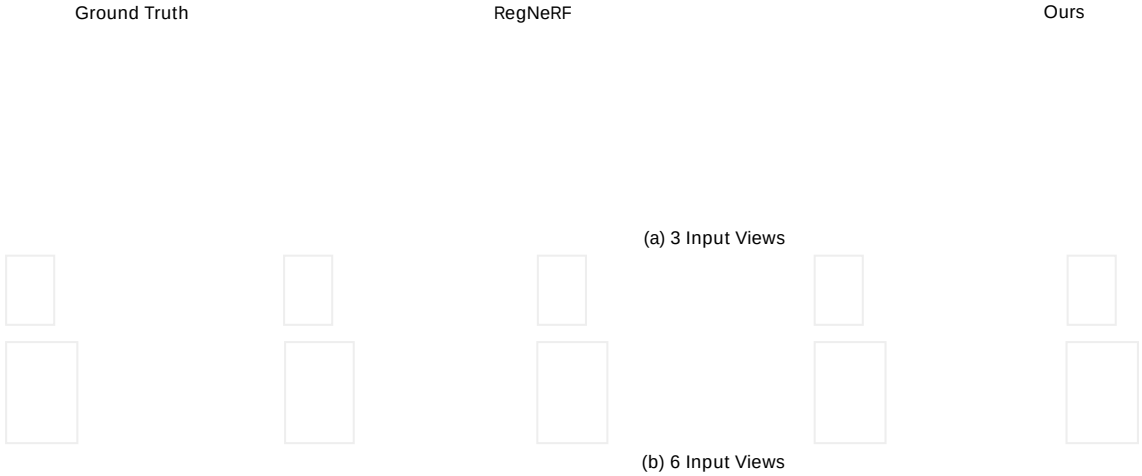


Figure 6. **Qualitative comparison on LLFFF.** RegNeRF [22] fails to estimate the accurate depth though it renders visually satisfactory RGB images (a). It also suffers from near-camera floaters (b). In contrast, our method reconstructs less noisy occupancy fields with fewer floaters. RegNeRF's results are rendered by our reproduced †RegNeRF concat. (*c.f.* Tab. 3).

qualitative results, respectively, on the LLFF dataset. We reproduce mipNeRF [2] and obtain better results. Our FreeNeRF is generally the best. Transfer learning-based methods [5, 4, 37] perform much worse than ours on the LLFF dataset due to the non-trivial domain gap between DTU and LLFF. Compared to RegNeRF [22], our approach predicts more precise geometry and exhibits fewer artifacts. For instance, RegNeRF's rendered "horns" example (Fig. 6-a) is perceptually acceptable but has poor depth map quality, indicating its incorrect geometry estimation. FreeNeRF, in contrast, renders a less noisy and smoother occupancy field. Also, our approach suffers less from "floaters" than ReNeRF (Fig. 6-b), further demonstrating the efficacy of our occlusion regularization.

Training overhead. In Table 4, we include the training time of different methods under the same setting. Our method only introduces negligible training overhead ($1.02-1.04\times$) compared to the other approaches ($1.62-2.8\times$). Both DietNeRF [11] and RegNeRF [22] render unobserved patches from novel poses for regularization, which significantly sets back the training efficiency. DietNeRF requires additional forward evaluation of a large model (CLIP ViT B/32, 224^2, [27]), and RegNeRF also experiences increased computation due to the use of a normalizing flow model (this part is not open-sourced and therefore not available for our experiments). In contrast, FreeNeRF does not require such additional steps, making it a lightweight and efficient solution for addressing few-shot neural rendering problems.

4.3. Ablation Study

In this section, we ablate our design choices on the DTU dataset and the LLFF dataset under the 3-view setting. We use a batch size of 1024 for faster training instead of 4096 for the main experiments in Tables 2 and 3.

Frequency curriculum. We investigate the impact of frequency regularization duration T in Figure 7. Our FreeN-

References

[1] Anonymous. Neural radiance fields with geometric consistency for few-shot novel view synthesis. In *Submitted to The Eleventh International Conference on Learning Representations*, 2023. under review.

[2] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021.

[3] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022.

[4] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14124–14133, 2021.

[5] Julian Chibane, Aayush Bansal, Verica Lazova, and Gerard Pons-Moll. Stereo radiance fields (srf): Learning view synthesis for sparse views of novel scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7911–7920, 2021.

[6] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12882–12891, 2022.

[7] Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B. Tenenbaum, and Jiajun Wu. Neural radiance flow for 4D view synthesis and video processing. *arXiv preprint arXiv:2012.09790*, 2020.

[8] Thibaud Ehret, Roger Marí, and Gabriele Facciolo. Nerf, meet differential geometry! *arXiv preprint arXiv:2206.14938*, 2022.

[9] Amir Hertz, Or Perel, Raja Giryes, Olga Sorkine-Hornung, and Daniel Cohen-Or. Sape: Spatially-adaptive progressive encoding for neural optimization. *Advances in Neural Information Processing Systems*, 34:8820–8832, 2021.

[10] Ajay Jain, Ben Mildenhall, Jonathan T Barron, Pieter Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 867–876, 2022.

[11] Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting nerf on a diet: Semantically consistent few-shot view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5885–5894, 2021.

[12] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 406–413, 2014.

[13] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing nerf for editing via feature field distillation. *arXiv preprint arXiv:2205.15585*, 2022.

[14] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, and Zhaoyang Lv. Neural 3d video synthesis, 2021.

[15] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. *https://arxiv.org/abs/2011.13084*, 2020.

[16] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[17] 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.

[18] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7210–7219, 2021.

[19] Ben Mildenhall, Peter Hedman, Ricardo Martin-Brualla, Pratul P Srinivasan, and Jonathan T Barron. Nerf in the dark: High dynamic range view synthesis from noisy raw images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16190–16199, 2022.

[20] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019.

[21] 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.

[22] Michael Niemeyer, Jonathan T Barron, Ben Mildenhall, Mehdi SM Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5480–5490, 2022.

[23] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. *ICCV*, 2021.

[24] Sida Peng, Yuanqing Zhang, Yinghao Xu, Qianqian Wang, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Neural body: Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans. In *CVPR*, 2021.

[25] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022.

[26] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural radiance fields for dynamic scenes. *https://arxiv.org/abs/2011.13961*, 2020.

[27] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[28] Daniel Rebain, Wei Jiang, Soroosh Yazdani, Ke Li, Kwang Moo Yi, and Andrea Tagliasacchi. DeRF: Decomposed radiance fields. *https://arxiv.org/abs/2011.12490*, 2020.

[29] Barbara Roessle, Jonathan T Barron, Ben Mildenhall, Pratul P Srinivasan, and Matthias Nießner. Dense depth priors for neural radiance fields from sparse input views. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12892–12901, 2022.

[30] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33:7462–7473, 2020.

[31] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547, 2020.

[32] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T Barron, and Pratul P Srinivasan. Ref-nerf: Structured view-dependent appearance for neural radiance fields. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5481–5490. IEEE, 2022.

[33] Yiqun Wang, Ivan Skorokhodov, and Peter Wonka. Hf-neus: Improved surface reconstruction using high-frequency details. *arXiv preprint arXiv:2206.07850*, 2022.

[34] Yi Wei, Shaohui Liu, Yongming Rao, Wang Zhao, Jiwen Lu, and Jie Zhou. Nerfingmvs: Guided optimization of neural radiance fields for indoor multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5610–5619, 2021.

[35] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. *https://arxiv.org/abs/2011.12950*, 2020.

[36] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. *Computer Graphics Forum*, 2022.

[37] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2021.

[38] Jason Zhang, Gengshan Yang, Shubham Tulsiani, and Deva Ramanan. Ners: Neural reflectance surfaces for sparse-view 3d reconstruction in the wild. *Advances in Neural Information Processing Systems*, 34:29835–29847, 2021.

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

Project page: FreeNeRF

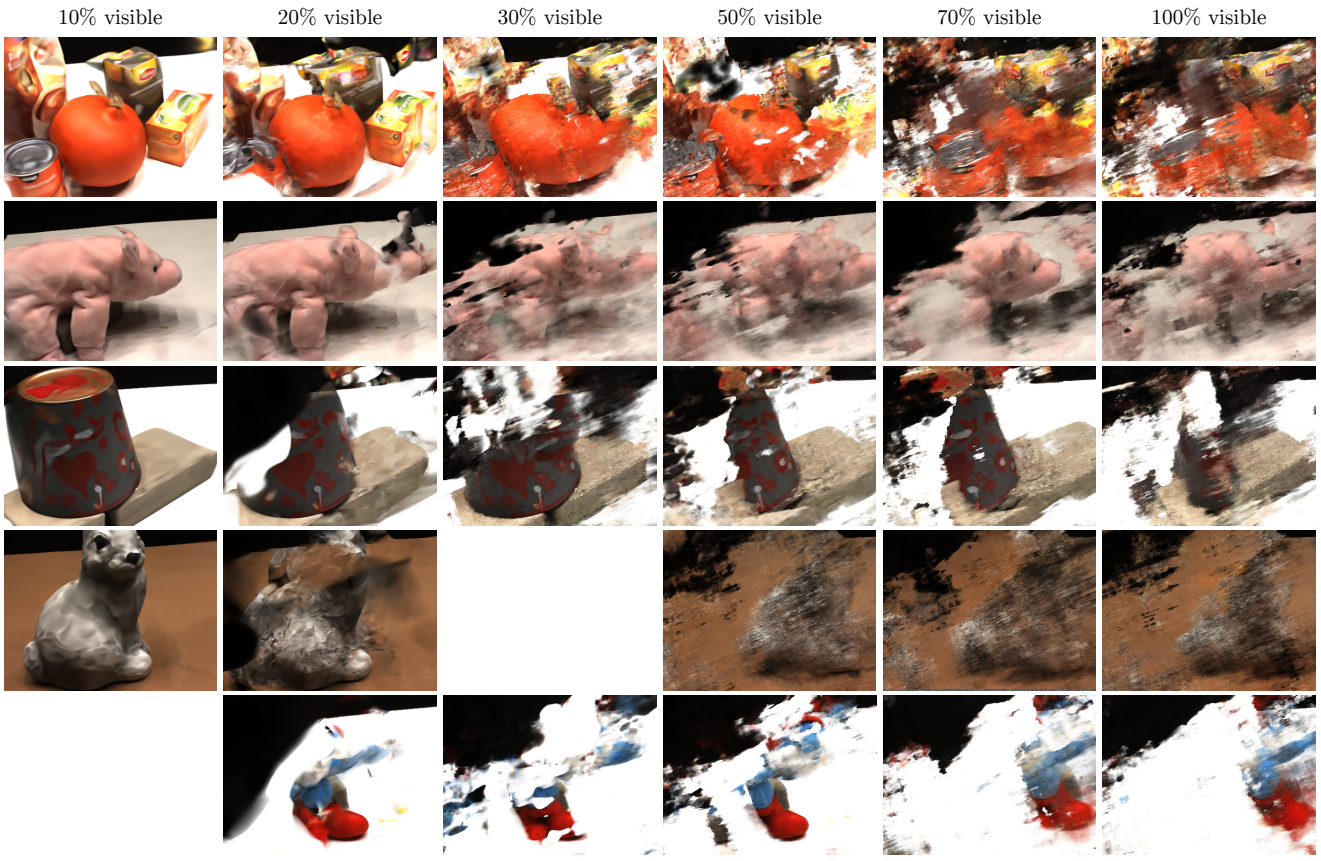


Figure A.1. **High-frequency inputs cause catastrophic failure in few-shot neural rendering.** We train mipNeRF [2] with masked (integrated) positional encoding by setting `pos_enc[int(L*x%]):]=0`, where L denotes the length of frequency bands (Eq. (1)) and x is the masking ratio. Using low-frequency 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

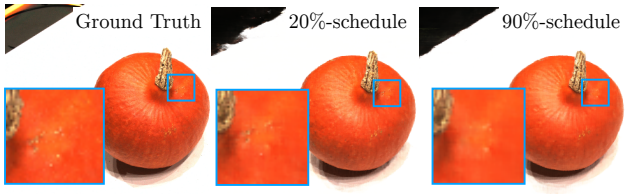


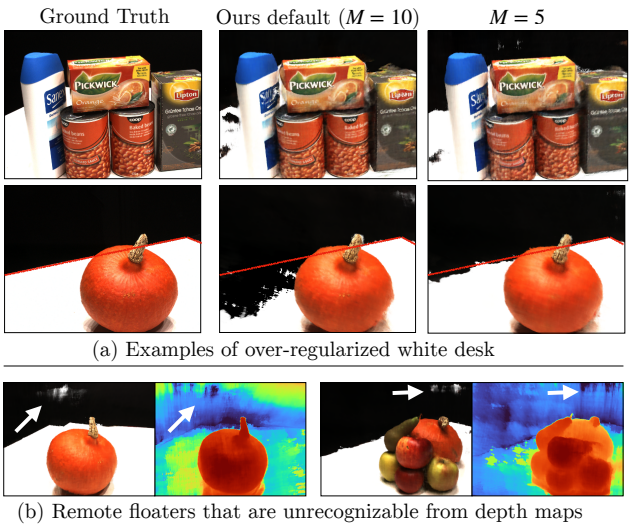
Figure A.2. **High-frequency details comparison.** We show the view synthesis results under the 9 input-view setting on the DTU dataset. With enough view information, a shorter frequency regularization enables NeRF models to render more high-frequency details.

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 hyper-parameters 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 [22, 21], 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/

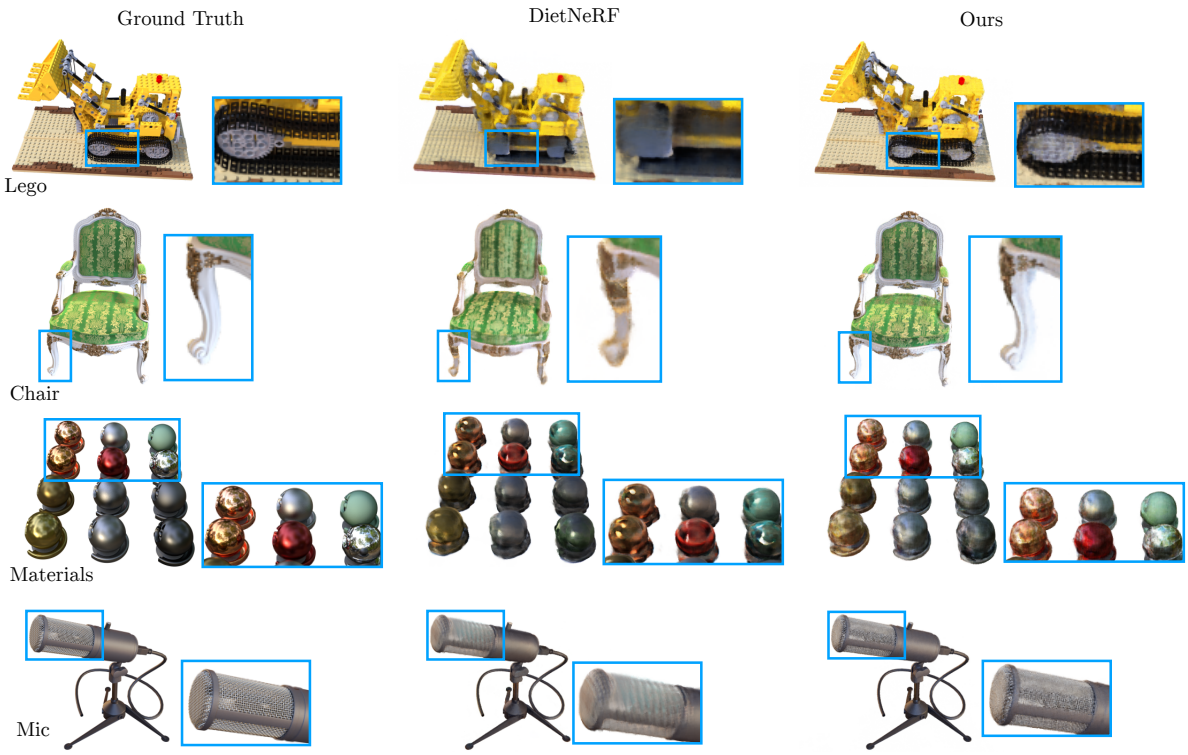


Figure A.4. **Qualitative comparison on the Blender dataset.** DietNeRF generates patches that can be reasonable and plausible to some extent but do not closely match the ground truth. This is a limitation of using a pre-trained model for semantic regularization. In contrast, our FreeNeRF reconstructs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

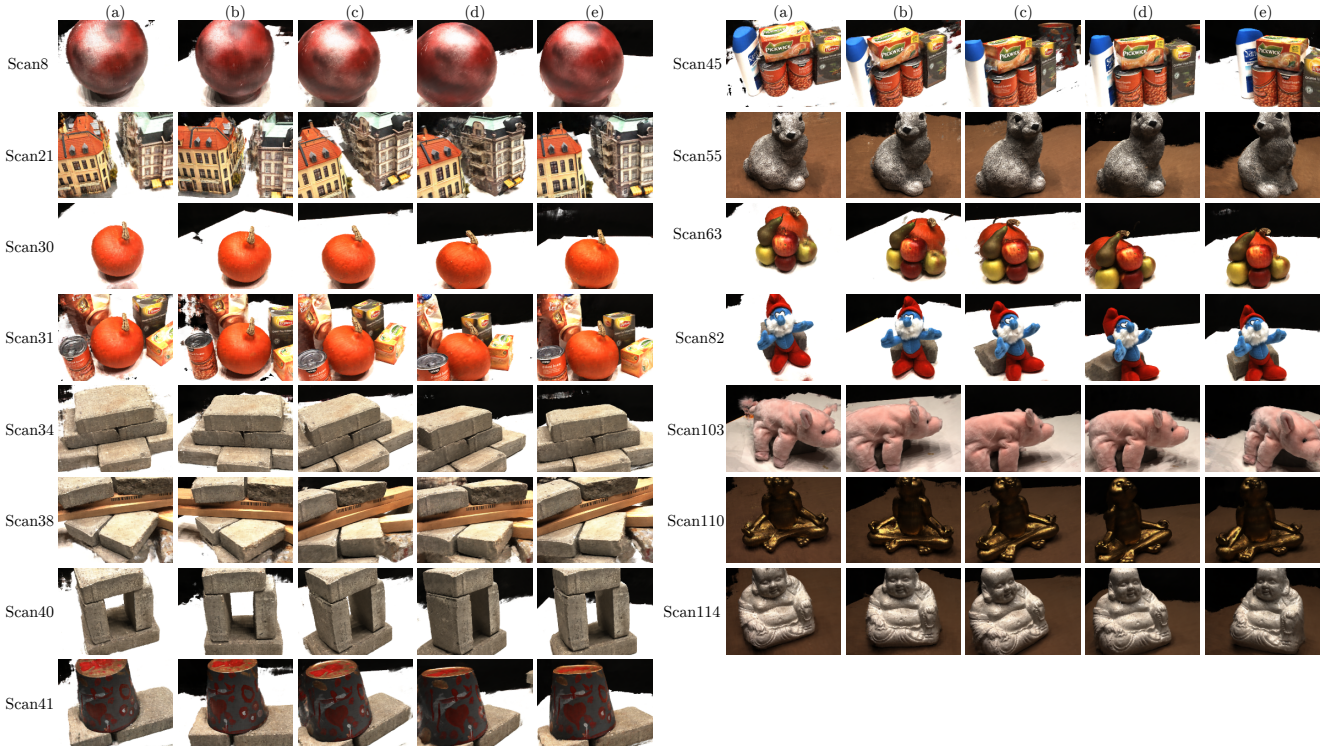


Figure A.5. **Example FreeNeRF's novel view synthesis results with 3 input views on the DTU dataset.**

transform/plot_ssim.html

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

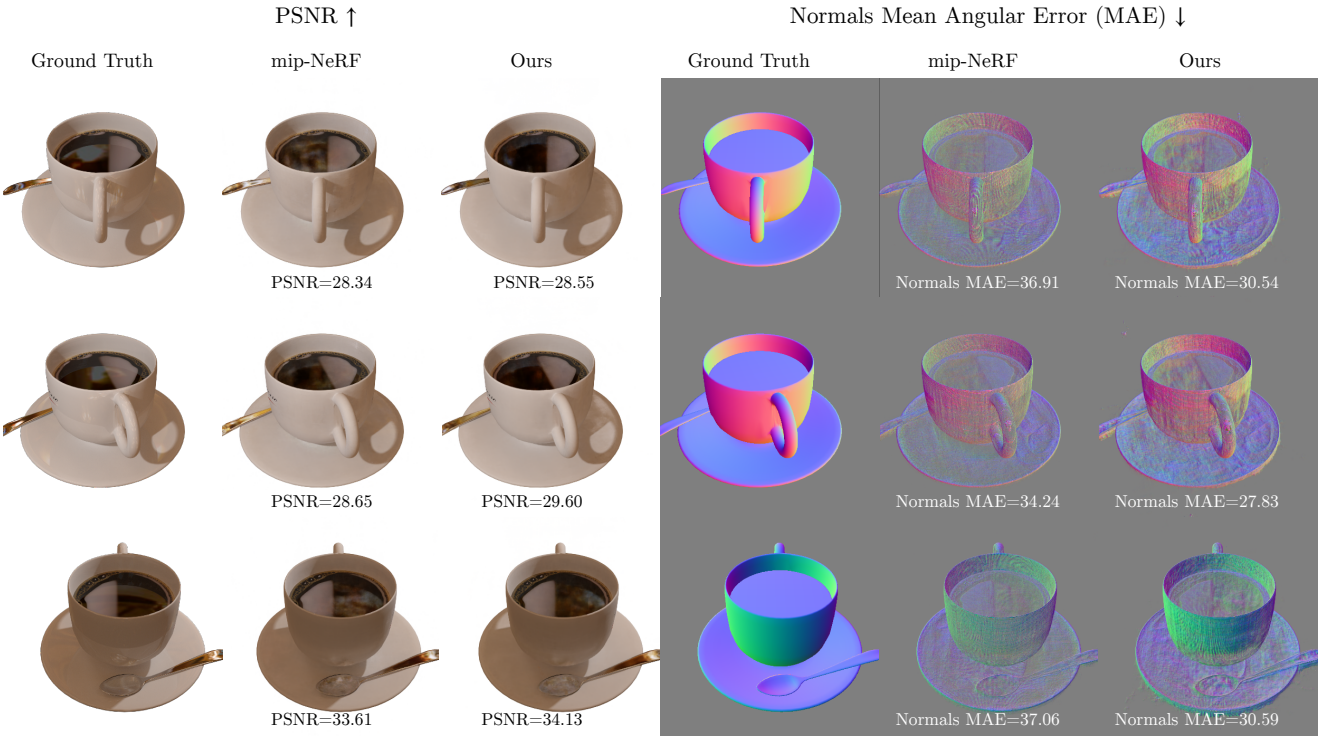


Figure A.7. **Comparison on normal vectors estimation.** Beyond the few-shot neural rendering problem, we train a mipNeRF and a FreeNeRF on the "coffee" scene in the Shiny Blender dataset [32] to demonstrate FreeNeRF's potential in estimating more accurate normal vectors. The PSNR scores for this scene 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