FourieRF: Few-Shot NeRFs via Progressive Fourier Frequency Control

Diego Gomez, Bingchen Gong, Maks Ovsjanikov
LIX, École Polytechnique, IP Paris
Palaiseau, France

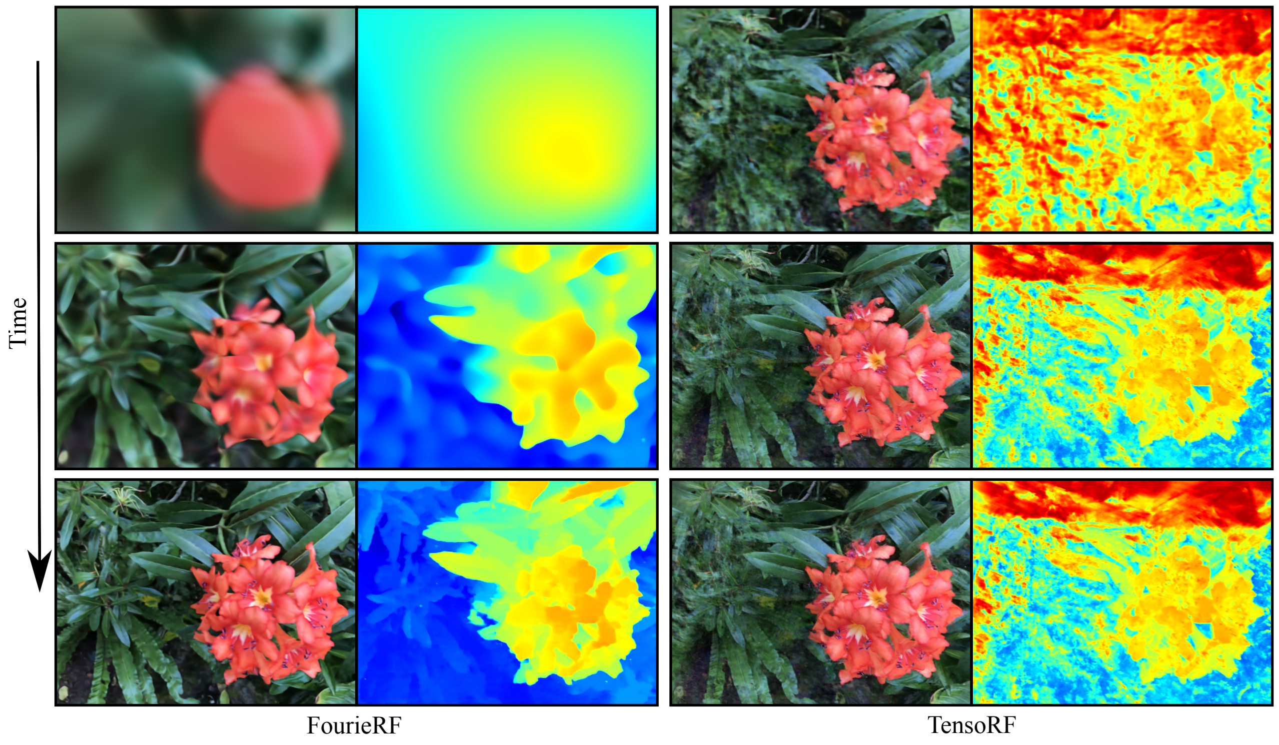


Figure 1. **FourieRF** serves as an effective and simple baseline for tackling the few-shot rendering problem. The vanilla approach often encounters high-frequency artifacts early in the optimization process. We introduce an explicit curriculum training procedure that gradually incorporates higher frequencies to mitigate this. This method ensures a stable training trajectory, eliminating major artifacts and enhancing overall rendering quality.

Abstract

We present a novel approach for few-shot NeRF estimation, aimed at avoiding local artifacts and capable of efficiently reconstructing real scenes. In contrast to previous methods that rely on pre-trained modules or various data-driven priors that only work well in specific scenarios, our method is fully generic and is based on controlling the frequency of the learned signal in the Fourier domain. We observe that in NeRF learning methods, high-frequency artifacts often show up early in the optimization process, and the network struggles to correct them due to the lack of dense supervision in few-shot cases. To counter this, we introduce an explicit curriculum training procedure, which progressively adds higher frequencies throughout optimization, thus favoring global, low-frequency signals initially, and only adding details later. We represent the radiance fields using a grid-based model and introduce an efficient approach to control the frequency band of the learned signal in the Fourier domain. Therefore our method achieves faster reconstruction and better rendering quality than purely MLP-based methods. We show that our approach is general and is capable of producing high-quality results on real scenes, at a fraction of the cost of competing methods. Our method opens the door to efficient and accurate scene acquisition in the few-shot NeRF setting.

1. Introduction

The introduction of Neural Radiance Fields (NeRFs) [13] has marked a significant milestone in the realm of 3D scene generation from 2D images using neural networks. NeRFs create continuous 3D scene representations by predicting color and density from various viewpoints, allowing them to synthesize photorealistic novel views from perspectives not included in the training data. This breakthrough has revolutionized applications in novel view synthesis, 3D asset generation, and inverse rendering, enabling unprecedented accuracy and realism in these domains [5, 6, 14].

However, one major challenge with NeRFs is their need for a large number of input images to ensure accurate scene reconstruction [22, 23]. This limitation highlights the importance of the *few-shot rendering problem*, which aims to perform novel view synthesis with only a limited set of input views. Advancements in this domain are critical, as they can expand the applicability of NeRFs to practical scenarios where data is sparse, effectively bridging the gap between 2D and 3D data representation. Addressing the few-shot rendering challenge requires robust reconstruction techniques and a deep understanding of image data, given the inherently under-constrained nature of the problem.

Several works tackle the few-shot rendering problem [2, 4, 8, 17, 18, 20, 22, 23], each with a unique approach but sharing the common goal of addressing the ambiguities of this ill-posed problem by introducing priors. Some methods incorporate *data-driven priors* [2, 4, 20, 23] by pre-training on diverse scenes [2, 23] or leveraging robust pre-trained modules, such as vision-language [4] or depth models [20]. A limitation of data-driven priors is that they often have difficulty generalizing their knowledge to new, unseen scenes. For example, these priors work well with scenes similar to their training data on indoor objects but struggle with novel or diverse scenes on wild natural or in vivo scenes due to the huge domain gap between the scene's distribution.

Unlike data-driven priors, some approaches rely solely on the information available in the given training data and use *explicit regularization* [8, 17, 18, 22]. Our work follows this latter direction, imposing a prior that constrains the search space of learnable parameters, similar in spirit to FreeNeRF's use of frequency masking [22] or ZeroRF's application of the Deep Image Prior [18, 19].

Existing methods for few-shot rendering face significant limitations in practical scenarios. Approaches leveraging *data-driven priors* are often computationally intensive, restricting their real-world applicability. Beyond the substantial computational cost of pre-training, most methods [2, 4, 20, 22, 23] are built on the original NeRF [13] or Mip-NeRF [1] frameworks, which are notoriously slow to train. While various techniques exist to accelerate NeRF training [3, 7, 14], only ZeroRF [18]—to the best of our knowledge—applies accelerated representations, such as grid-based methods, to the *few-shot setting without data priors*. ZeroRF adapts TensoRF [3] for this context but struggles to handle real-world scenes effectively, as acknowledged by its authors [18]. Other approaches address few-shot rendering using 3D Gaussian splatting [10, 24]; however, these rely on depth information, placing them outside the scope of our comparisons.

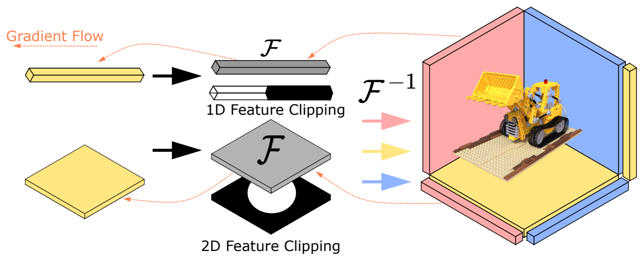


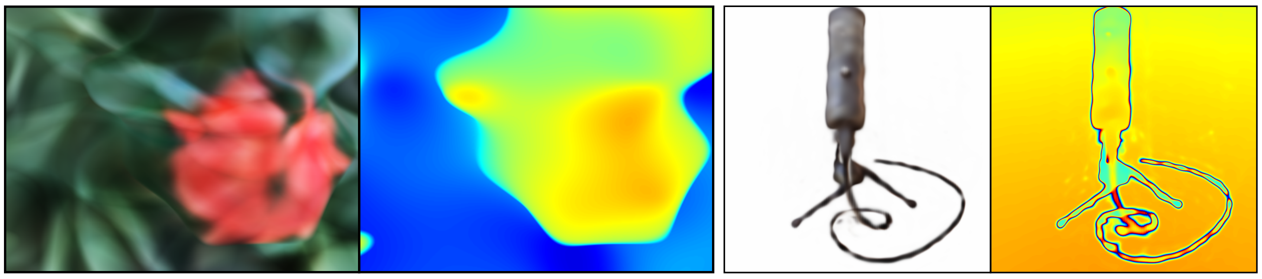
Figure 2. **Method illustration.** From left to right. Feature vectors and matrices are initialized in the spatial space. They are projected using the FFT. The Fourier coefficients are clipped using the masking procedure. Finally, the inverse FFT is applied to retrieve the smoothed features.

In this work, we present **FourieRF** a method to parameterize the features of grid-based NeRF methods using a general prior based on the Fourier Transform. Our approach has virtually no computational overhead since we apply it per iteration. FourieRF is fast, robust, and effective across a range of scenes, from synthetic to real. To showcase our method we compare FourieRF against the best existing *learning-free* methods and show that FourieRF establishes a new state-of-the-art by delivering robust results in record time. In summary, our contributions are as follows:

- We demonstrate that by constraining the maximum Fourier frequency, it is possible to regulate the level of details in NeRF's scene representations in an artifact-free manner.
- We show that the smooth shapes learned with low frequency accurately capture the scene's coarse geometry, providing robust and stable initialization in both few-shot and dense inputs.
- We introduce a fast grid-based NeRF representation that band-limits feature grids' frequency with a novel training curriculum. Our simple approach produces results on par with state-of-the-art while requiring a fraction of the time to converge.

2. Related Work

Radiance Field Representations The seminal work Neural Radiance Fields (NeRF) [13] uses deep neural architectures and a set of given images to produce photo-realistic novel-view synthesis (NVS). Subsequent grid methods, accelerate the pipeline by replacing the deep neural architecture with a 3D grid of features [3], or a multi-resolution hash grid [14]. These grids' NeRF approaches were shown to converge orders of magnitudes faster than the original work while maintaining high-quality results. Encoding the information in an explicit 3D representation is a weak prior that leads to significantly easier learning. However, a common


(a) Flower 3 Views (b) Mic 4 Views

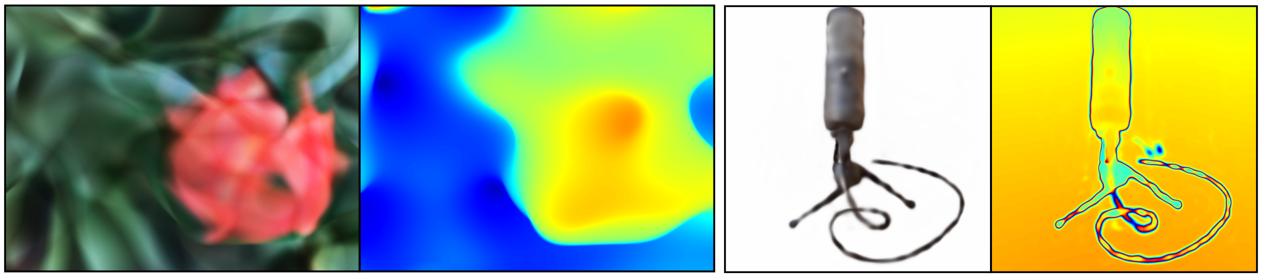

(c) Flower 9 Views (d) Mic 6 Views

Figure 3. **Coarse Geometry Extraction.** Our method is capable of extracting correct coarse geometry from as little as 3 views. This coarse geometry remains relatively stable regardless of the number of views we input.

struggle of all NeRF representations is that they are *data hungry*, they require several images to perform the NVS task properly. When given limited images, say 3 or 6, these models are extremely prone to overfitting; *they produce incoherent geometry and images from novel points of view.*

Few-shot Novel View Synthesis The problem described above is commonly referred to as the few-shot rendering problem. Approaches to address it can be categorized into two groups: methods that rely on data priors, such as scene depth [10, 20, 24] or diffusion models [21] to mitigate reconstruction ambiguities; and methods that operate without external data [2, 4, 8, 17, 18, 22, 23]. In this work, we focus on comparing our approach with state-of-the-art methods that do not rely on data priors, applying solely *explicit regularization*, specifically FreeNeRF [22] and ZeroRF [18]. We demonstrate that our method achieves state-of-the-art results while setting new records for speed, leveraging the Fourier transform to effectively control the complexity of features in a 3D grid.

Compression Neural Fields The work FreeNeRF [22] shows there exists a link between the failure of few-shot neural rendering and the positional encoding used by deep neural network-based NeRF methods. Their method proposes to mask the encoding and progressively give it to the network. This allows for a coarse-to-fine reconstruction. Nevertheless, their take is specific to NeRF representations that are based on deep neural architecture, it is not trivial to apply it to accelerated grid methods. Therefore making FreeNeRF *extremely* slow to train.

On the other hand, ZeroRF [18] leverages the Deep Image Prior [19] to parameterize a grid NeRF representation. This work constitutes, to our knowledge, the first instance of an accelerated NeRF representation that specifically tackles the few-shot rendering problem. The authors, however, acknowledge the limited applicability of their method to *real* (i.e., non-synthetic) scenes.

We present FourieRF a method to parameterize the features of grid-based NeRF representations. Our method trains in record time (approximately 10 minutes) and can tackle real and synthetic scenes.

3. Motivation and Overview

The few-shot rendering problem is inherently ill-posed. With only a few images, there is insufficient information to uniquely determine the reconstructed 3D scene. As a result, training a standard NeRF model on very limited data will inevitably lead to overfitting, where the model memorizes the few input images and captures noise instead of learning an underlying 3D structure. When an overfitted NeRF tries to synthesize new views, the results will be inaccurate or unnatural, producing distorted images that are unrealistic and inconsistent with the true 3D structure of the scene [22, 23].

The problem highlighted by the FreeNeRF paper is that the vanilla NeRF's use of high-frequency positional encodings can cause the model to overfit drastically during the initial training iterations when given only a few input views. These high-frequency positional encoding inputs allow the model to quickly learn to fit the available data (the few training views) too well, but in doing so, it generates unrealistic or degenerate geometries, such as floaters. These ge-

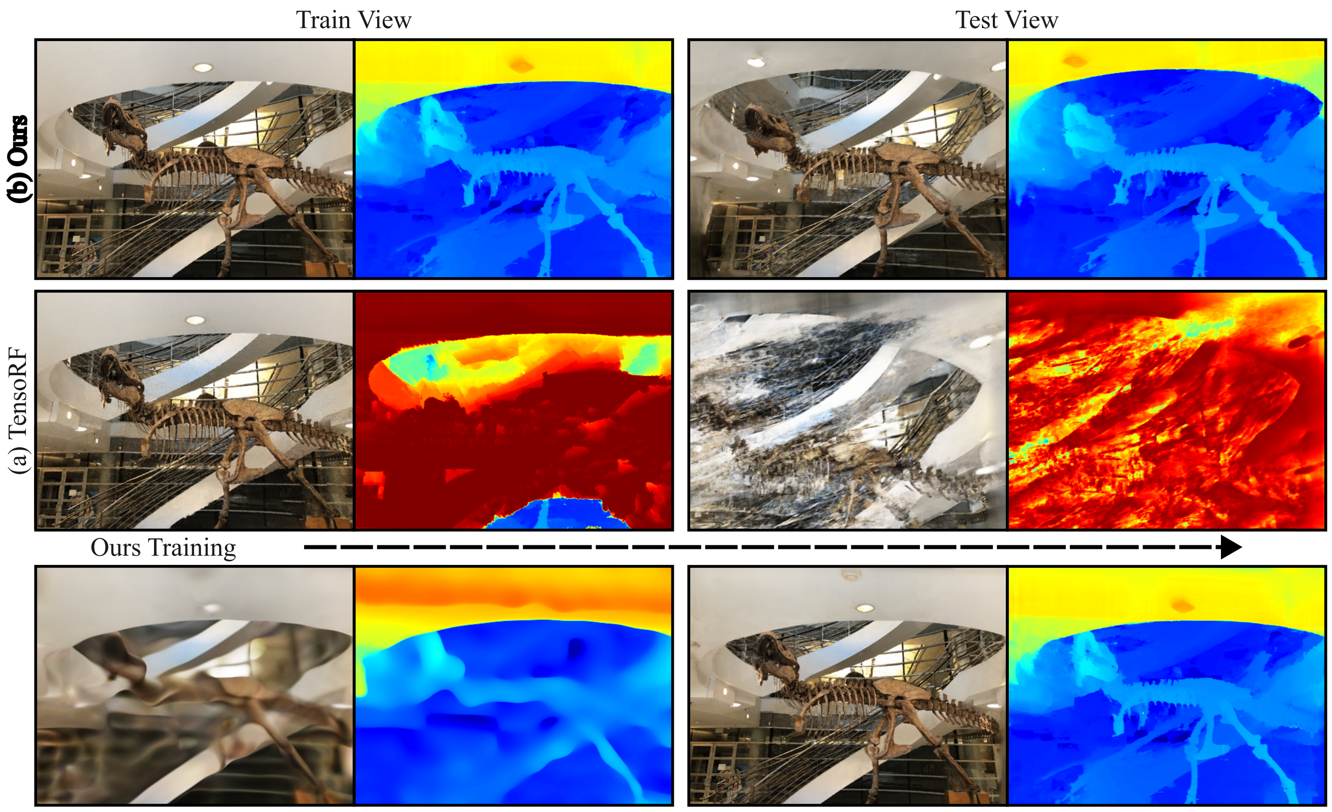


Figure 4. **Overfitting on the few-shot rendering problem.** “Catastrophic overfitting” is a common behavior for standard NeRF representations on the few-shot rendering problem. Degenerate geometry is learned, which might result in plausible views near train inputs but does not generalize to novel views.

ometries do not accurately represent the true structure of the scene but are instead arbitrarily constructed patches that help the network mimic patterns in input views.

This overfitting behavior is not limited to MLP-based NeRFs with positional encodings but is also observed in other NeRF representations. TensoRF [3] is an improved NeRF design that utilizes tensor decomposition to reduce computational complexity and memory usage while maintaining quality for 3D scene reconstruction. However, when directly used in few-shot scenarios, TensoRF also suffers from the overfitting problem. In Fig. 1, we can see that TensoRF quickly fills the space with floaters that help the reconstruction on the set of limited views. These floaters do not generalize well to new views, but since they fit correctly the input views, removing them is challenging.

FreeNeRF [22] addresses overfitting by masking positional encodings for the MLP scene representation. This method is not directly applicable to grid-based methods like TensoRF [3], limiting its application given the long training times of MLP-based NeRF. ZeroRF [18] is the first work to use accelerated NeRF structures to tackle the few-shot rendering problem. It employs a convolutional network to generate feature maps of a TensoRF [3] representation, resulting in quickly trained “clean” feature maps. However, this approach doesn’t generalize well to real scenes [18], as its strong prior is not valid for non-synthetic scenes.

FourieRF deals with these two issues: it is an **accelerated** method, training in less than 10 minutes, that can tackle a wide variety of scenes, from **synthetic** to **real**. See Fig. 2 for an illustration of our method. Using our approach, we can obtain correct coarse geometry from simple and complex scenes, see Fig. 3.

Overall, our method is built on two key observations. First, we note that both strong overfitting and high-frequency artifacts typically occur early in the optimization process (see Fig. 4), and, if avoided in these early stages, they are significantly less prominent in the final result. Second, we note that by *gradually* increasing the maximal Fourier frequency of the learned signal both significantly regularizes the learned NeRF, while at the same time, providing the network enough degrees of freedom to learn the fine details (in the final stages of the optimization).

In other words, progressively increasing the available frequencies builds a robust trajectory to maintain the correctness of the shape as well as produce photo-realistic results 1. This coarse to fine prior is not specific to any data type, and thus works in both real and synthetic scenes.

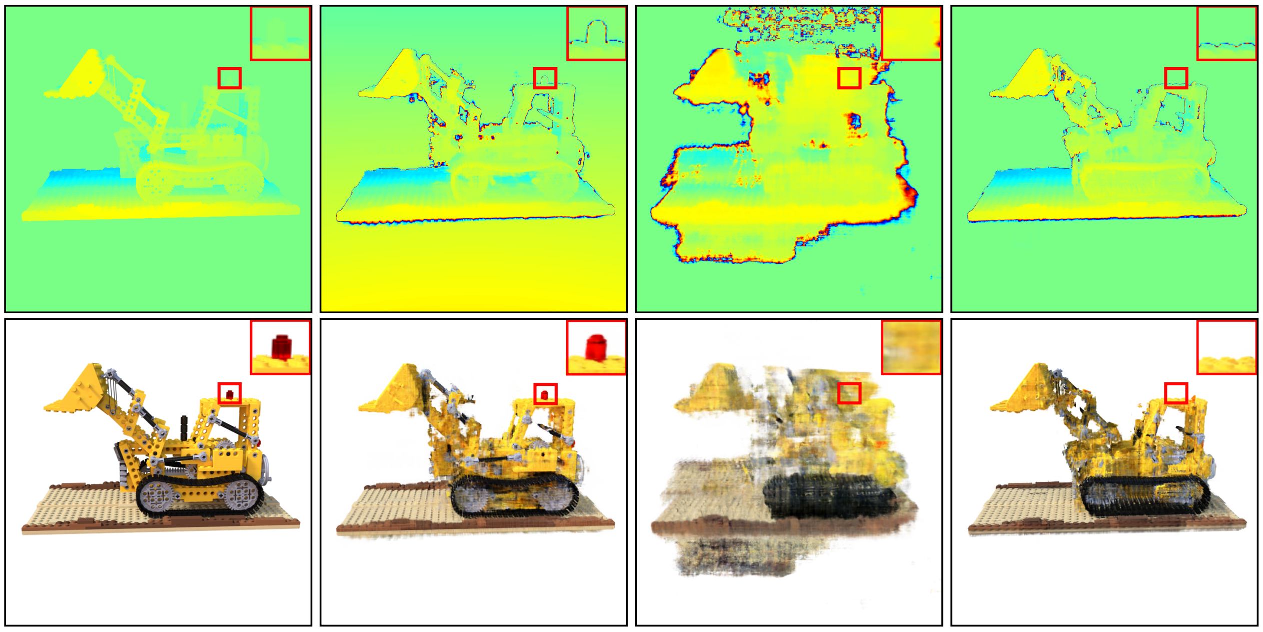


Figure 5. **Comparison on Blender Dataset.** In the Lego scene, trained with 4 views, we compare the performance of FreeNeRF, ZeroRF, and our method. ZeroRF renders a compact and clean reconstruction of the scene, however, at the cost of omitting some key details. FreeNeRF fails in this new setting due to its reliance on complex occlusion regularization. Despite employing a simple prior, FourieRF accurately captures both geometry and appearance, demonstrating a faithful reconstruction of the scene's details.

4. Method

We apply our Fourier parameterization to representations introduced by TensoRF [3], increasing the maximal Fourier frequency in tensor decomposition gradually.

4.1. Preliminaries

The key idea behind grid-based NeRF representation is to represent the scene using a decomposed feature grid rather than a deep neural network. We denote the 3D tensor of features that represents the scene as $\mathcal{T} \in \mathbb{R}^{I\times J\times K}$. In our NeRF representation, we experiment with two different methods to decompose this 3D tensor:

CANDECOMP/PARAFAC (CP) Decomposition In the CP decomposition, $\mathcal{T}$ is decomposed as a sum of outer products of vectors:

$$\mathcal{T} = \sum_{r=1}^{R} v_r^1 \circ v_r^2 \circ v_r^3$$

where $v_r^1 \circ v_r^2 \circ v_r^3$ corresponds to a rank-one tensor component, and $v_r^1 \in \mathbb{R}^I, v_r^2 \in \mathbb{R}^J, v_r^3 \in \mathbb{R}^K$ are factorized vectors of the three modes for the r-th component.

CP factorization reduces space complexity from $\mathcal{O}(n^3)$ to $\mathcal{O}(n)$, and offers low-rank regularization at the same time in the optimization, making it a good candidate NeRF representation for few-shot reconstruction. On the other hand, CP sacrifices the rendering quality to minimize the rank of the decomposition.

Vector-Matrix (VM) Decomposition Unlike CP factorization, VM decomposition enriches the product by using matrices. The decomposition is expressed as:

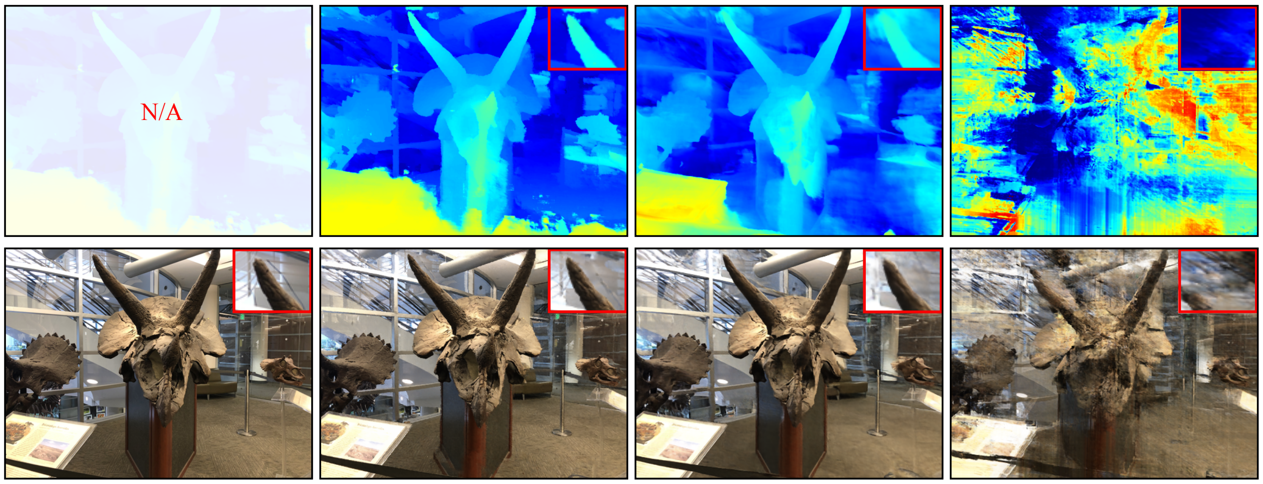
$$\mathcal{T} = \sum_{r=1}^{R_1} v_r^1 \circ M_r^{2,3} + \sum_{r=1}^{R_2} v_r^2 \circ M_r^{1,3} + \sum_{r=1}^{R_3} v_r^3 \circ M_r^{1,2},$$

where $M_r^{2,3} \in \mathcal{R}^{J\times K}, M_r^{1,3} \in \mathcal{R}^{I\times K}, M_r^{1,2} \in \mathcal{R}^{I\times J}$ are matrices for two of the three modes.

The VM decomposition reduces space complexity from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$. For complex scenes, the VM decomposition reduces the number of components required to achieve the same expressivity as CP, leading to faster reconstruction and better rendering. Our method can be applied to any of these decompositions. In practice, applying our method to the VM decomposition allows us to model more complex effects, leading to better quantitative performance (See Table 1).

4.2. Fourier Parameterization

As mentioned in Section 3, the key observation behind our method is that objects' geometry and appearance can be learned in a coarse-to-fine manner based on their corresponding low to high frequencies in the underlying NeRF representation. Our work makes the following claims: (i)



(a) GT **(b) Ours** (c) FreeNeRF (d) ZeroRF

Figure 6. **Comparison in LLFF Dataset.** In the horns scene, we evaluated the performance of FreeNeRF, ZeroRF, and our method under a 3-view training setup. ZeroRF struggled to reconstruct coherent geometry, resulting in significant inconsistencies. FreeNeRF, while more stable, produced renders with notably blurred geometry, failing to capture fine details accurately. In contrast, FourieRF delivers sharper renders, faithfully reconstructing the *key* geometric elements of the scene with high fidelity.

There are enough constraints in few-shot inputs to learn an accurate coarse geometry under low-frequency constraints; (ii) Lower frequencies are easier to learn correctly than higher frequencies; (iii) Learning the next set of higher frequencies is more straightforward given a correct set of lower frequencies.

Let us illustrate the above claims: in Fig. 3 we can see that using our method we can establish a good base for scenes, even in the case of complex real scenarios. Moreover, Fig. 1, shows that given a good estimation of the low-frequencies of the scene, we can progressively add complexity to the object while maintaining a clean reconstruction. In the following section, we demonstrate the process of parameterizing 1D and 2D features using our method. This parameterization allows us to begin with well-established coarse geometry and progressively incorporate fine details. In both cases, the principle remains the same: project the features into Fourier space and remove high frequencies up to a certain threshold. We make use of the discrete Fourier transform, so given a fixed grid size, there is a finite number of Fourier coefficients after transformation. The threshold we use is proportional to this finite number, please refer to the supplementary for typical values.

1D Features. For 1D features we have $\mathbf{v} \in \mathcal{R}^d$, a d-dimensional feature. At time t, given a feature threshold f_t we perform the following operations. The feature $\mathbf{v}$ is projected into the Fourier space, then the Fourier coefficients are clipped using the threshold f_t; finally, we apply the inverse Fourier transform,

$$\hat{\mathbf{v}} = \mathbf{IFFT}(\mathbf{FFT}(\mathbf{v}) \odot \alpha(f_t)) \tag{1}$$

Call d_f the dimension of $\mathbf{FFT}(\mathbf{v})$, then in practice the mask $\alpha(f_t)$ corresponds to an array of the same dimension where we keep cells up to index $t_\alpha(f_t) = d_f \times f_t$. Each cell i is given by,

$$\alpha_i(f_t) = \begin{cases} 1 \text{ if, } i < t_\alpha(f_t) \\ t_\alpha(f_t) - \lfloor t_\alpha(f_t) \rfloor \text{ if, } i = t_\alpha(f_t) \\ 0 \text{ otherwise} \end{cases} \tag{2}$$

The same 1D feature parameterization is applied to both, the CP and the VM decompositions.

2D Features. For 2D features, we have $\mathbf{w} \in \mathcal{R}^{d_1 \times d_2}$ a matrix feature. We proceed as above,

$$\hat{\mathbf{w}} = \mathbf{IFFT}(\mathbf{FFT}(\mathbf{w}) \odot \beta(f_t)) \tag{3}$$

The difference mainly lies in the mask $\beta(f_t)$. In this case, we define a 2D mask with a circle centered at $c = \lfloor \frac{d_1}{2} \rfloor, \lfloor \frac{d_2}{2} \rfloor$, of radius $r = \frac{f_t}{2}\sqrt{2\max(d_1, d_2)^2}$ This ensures that when f_t reaches 100% all parameters are used. The values outside of the circle are set to 0 to clip the corresponding coefficients. The 2D feature parameterization is only used in the VM decomposition.

Progressive Inclusion of Coefficients. To smoothly control the frequency of our feature grid decomposition, we define a mask to clip the corresponding Fourier coefficients and progressively increase the frequency using a clipping threshold. The illustration in Fig. 1 shows the progressive increase effects. When setting the clipping parameter, it is important to keep it sufficiently low initially to ensure the correct coarse geometry. Examples of successful choices of clipping can be seen in Fig. 3. In our method, we initially start with only $f_0 = 0.01\%$ of Fourier coefficients and then linearly increase the clipping parameter during training. To be specific, we update it every iteration as follows:

Table 1. Quantitative comparison on Blender.

Method	Prior	PSNR ↑		SSIM ↑		LPIPS ↓	
		4 views	6 views	4 views	6 views	4 views	6 views
DietNeRF [4]	CLIP	10.92	16.92	0.557	0.727	0.446	0.267
RegNeRF [4]	RealNVP	9.93	9.82	0.419	0.685	0.572	0.580
TensoRF [3]	No Prior	18.656	21.652	0.798	0.844	0.216	0.165
ZeroRF [18]	Deep Network	21.94	24.73	0.856	0.889	0.139	0.113
FreeNeRF [22]	Frequency	18.81	22.77	0.808	0.865	0.188	0.1495
Ours - CP	Frequency	20.799	22.496	0.825	0.849	0.217	0.195
Ours	Frequency	21.728	23.927	0.858	0.879	0.147	0.136

Table 2. Quantitative comparison on LLFF.

Method	PSNR ↑			SSIM ↑			LPIPS ↓		
	3 views	6 views	9 views	3 views	6 views	9 views	3 views	6 views	9 views
DietNeRF [4]	14.94	21.75	24.28	0.370	0.717	0.801	0.496	0.248	0.183
RegNeRF [15]	19.08	23.10	24.86	0.587	0.760	0.820	0.336	0.206	0.161
TensoRF [3]	14.292	18.183	23.677	0.315	0.576	0.777	0.545	0.370	0.213
ZeroRF [18]	16.74	21.371	22.425	0.434	0.698	0.750	0.470	0.302	0.275
FreeNeRF [22]	19.63	23.73	25.13	0.612	0.779	0.827	0.308	0.195	0.160
Ours	19.303	23.595	25.011	0.636	0.790	0.830	0.299	0.210	0.193

$f_t = f_{t-1} + \Delta$, with $\Delta = \frac{1-f_0}{N}$ Where t is the iteration number, f_0 is the initial clipping, and N is the number of iterations. The update is applied at the start of every iteration, before any gradient is accumulated, thus avoiding any differentiability problems.

View Dependence In the few-shot learning setting, it is extremely challenging to learn view-dependent information. We hypothesize that the model can use directional information (d) and positional encodings to overfit to a limited set of views. In practice, we have found it sufficient to restrict the directional information provided to the model. We adopt a similar approach to ZeroRF [18] by using a simplified decoder that *does not* use positional encodings for features or view directions.

5. Experiments

5.1. Setup

Datasets & metrics. Our method **FourieRF** can process a wide variety of scenes. We thus test it on synthetic and real scenes. The **NeRF-synthetic** dataset [13] was rendered using Blender containing 8 objects with complex material and geometric information. We use ZeroRF's [18] *same setting to train and evaluate* our method (with the number of views ranging from 4 to 6). The **LLFF** [12] contains 8 real scenes. We use RegNeRF's [15] *same setting to train and evaluate* our method (training on 3, 6, or 9 views).

Implementation. FourieRF can be easily added to the TensoRF [3]. Our method can enhance both the performance of the CP and VM decomposition, and in each case we create a class that inherits from the respective TensoRF decomposition. We find in practice that the VM decomposition leads to better results (See Table. 1), so we use it in all the experiments unless otherwise stated. Please see the supplementary material for more information about our code and the hyper parameters used.

Baselines Our first baseline is vanilla TensoRF-VM [3], as it is the foundation upon which our method is built. We also compare our method to ZeroRF [18] and FreeNeRF [22], the most recent and directly comparable baselines. ZeroRF specializes in the reconstruction of synthetic scenes and is an accelerated method that trains in around 30 minutes, but, as noted earlier, it struggles with real scenes. In contrast, FreeNeRF can process any type of scene [22], but its training time is extremely long (approximately one day). For completeness, we also include quantitative comparisons with well-established baselines such as DietNeRF [4], which leverages data priors, and RegNeRF [15], which relies on geometrical regularizations.

5.2. Results

Our method achieves performance that is on par with the state-of-the-art (SOTA) approaches, while being significantly faster than comparable methods. We refer the reader to the supplementary material, where we showcase a video presentation and animated results.

Indeed in Table 3, we can see that our method's training time is even faster than TensoRF. Our Fourier parameterization is done per iteration at virtually no cost. Moreover, as seen in Fig. 4, TensoRF's scene representation is filled with floaters. This hinders training, filling the scene with noise, thus slowing down the whole procedure. When compared to the other accelerated method, ZeroRF [18], we see that their Deep Image Prior [19], comes at the cost of evaluating an expensive convolutional neural network. Our method is by far the fastest to converge in the few-shot rendering task.

The results of our quantitative evaluation are showcased in Table 1 and Table 2. In the synthetic dataset, we greatly outperform all MLP-based methods by a significant margin. We achieve this while training an order of magnitude faster, and without the use, of the hard-to-tune, occlusion regularization used by FreeNeRF [22]. The SOTA in this dataset is the accelerated method ZeroRF [18]. However, we achieve similar results while training over 5 times faster, and Fig. 5 shows that ZeroRF's prior can lead to the omission of key geometry. Despite its state-of-the-art performance on synthetic datasets, ZeroRF [18] fails to handle real scenes effectively. This goes to show, that their Deep Image Prior [19] does not generalize to diverse scenes. Our method achieves results that are on par with FreeNeRF [22], while training *over 30 times faster.* Moreover, in Fig. 6, we see that the shapes we extract are in some cases smoother than FreeNeRF's.

These results demonstrate that we have introduced an exceptionally *simple* and *flexible* baseline. Our method performs well across a variety of scenes while training at record speeds, highlighting its practicality and ease of use.

Ablations. The progressive inclusion of complexity is a key aspect of our method. In Fig. 1, we illustrate a successful training trajectory. However, determining "how quickly" this complexity is integrated—i.e., setting the parameter Δ defined in Section 4.2—is crucial. Fig. 7 shows the relationship between the choice of this parameter and the PSNR obtained when training and testing across all scenes from the Blender synthetic dataset (both using 6 views). Fig. 7 shows that for large Δ, the method converges to the baseline, TensoRF. In practice, choosing a sufficiently small increment should yield adequate performance. We observe a slight decrease in performance for smaller values of Δ, which is likely due to the fixed 10k training iterations used in the experiment. For the smallest increments, the model simply did not have enough training time.

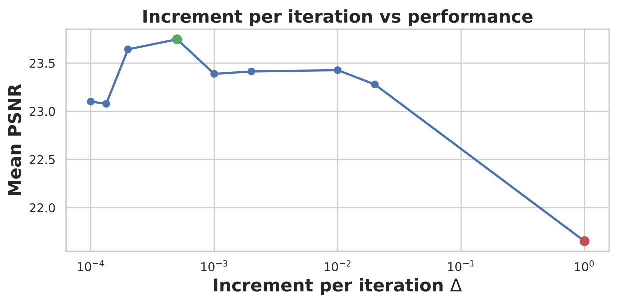


Figure 7. **Choice of Δ vs performance.** We investigated the effect of varying the speed at which high-frequencies are integrated during training, using the Blender Dataset with 6 views. The baseline performance without our method is highlighted in red, while our best result is shown in green.

Table 3. Global execution time comparison relative to TensoRF. Training on the Blender Dataset for 10k iterations.

Method	Training Time
TensoRF	1.0×
ZeroRF	5.181×
FreeNeRF	35.71×
Ours	0.93×

Finally, our method stands as the *fastest* baseline available for the few-shot rendering problem. As shown in Table 3, our execution time is virtually identical to that of vanilla TensoRF [3], with our method being slightly faster because we do not predict "useless" floaters. Additionally, we are more than five times faster than ZeroRF [18], the only other accelerated method addressing the few-shot rendering problem.

6. Conclusion & Future Work

In this work, we introduce **FourieRF**, a novel approach for achieving fast and high-quality reconstruction in the few-shot setting. Our method effectively parameterizes features through an explicit curriculum training procedure, incrementally increasing scene complexity during optimization. Experimental results show that the prior induced by our approach is both robust and adaptable across a wide variety of scenes, establishing **FourieRF** as a strong and versatile baseline for the few-shot rendering problem. While our approach significantly reduces artifacts, it may still lead to reconstruction errors in severely under-constrained scenarios, particularly where view occlusion leaves parts of the shape uncovered. In the future, our method could be enhanced by integrating foundation models to complete missing parts using large data-driven priors.

Acknowledgement. Parts of this work were supported by the ERC Consolidator Grant 101087347 (VEGA) and the ANR AI Chair AIGRETTE.

References

[1] 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. *ICCV*, 2021. 2

[2] 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. 2, 3

[3] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. TensoRF: Tensorial radiance fields. In *European Conference on Computer Vision (ECCV)*, 2022. 2, 4, 5, 7, 8, 1

[4] 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. 2, 3, 7

[5] Clément Jambon, Bernhard Kerbl, Georgios Kopanas, Stavros Diolatzis, Thomas Leimkühler, and George Drettakis. NeRFshop: Interactive editing of neural radiance fields. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 6(1), 2023. 2

[6] Haian Jin, Isabella Liu, Peijia Xu, Xiaoshuai Zhang, Songfang Han, Sai Bi, Xiaowei Zhou, Zexiang Xu, and Hao Su. TensoIR: Tensorial inverse rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2

[7] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 2

[8] Mijeong Kim, Seonguk Seo, and Bohyung Han. InfoNeRF: Ray entropy minimization for few-shot neural volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12912–12921, 2022. 2, 3

[9] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1

[10] Jiahe Li, Jiawei Zhang, Xiao Bai, Jin Zheng, Xin Ning, Jun Zhou, and Lin Gu. Dngaussian: Optimizing sparse-view 3d gaussian radiance fields with global-local depth normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20775–20785, 2024. 2, 3

[11] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 1

[12] 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 Trans. Graph.*, 38(4):1–14, 2019. 7

[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. 1, 2, 7

[14] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4), 2022. 2

[15] 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. 7

[16] Julien Philip and Valentin Deschaintre. Floaters no more: Radiance field gradient scaling for improved near-camera training. *arXiv preprint arXiv:2305.02756*, 2023. 1

[17] Seunghyeon Seo, Yeonjin Chang, and Nojun Kwak. Flipnerf: Flipped reflection rays for few-shot novel view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22883–22893, 2023. 2, 3

[18] Ruoxi Shi, Xinyue Wei, Cheng Wang, and Hao Su. Zerorf: Fast sparse view 360deg reconstruction with zero pretraining. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21114–21124, 2024. 2, 3, 4, 7, 8, 1

[19] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9446–9454, 2018. 2, 3, 8

[20] Guangcong Wang, Zhaoxi Chen, Chen Change Loy, and Ziwei Liu. Sparsenerf: Distilling depth ranking for few-shot novel view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9065–9076, 2023. 2, 3

[21] Rundi Wu, Ben Mildenhall, Philipp Henzler, Keunhong Park, Ruiqi Gao, Daniel Watson, Pratul P Srinivasan, Dor Verbin, Jonathan T Barron, Ben Poole, et al. Reconfusion: 3d reconstruction with diffusion priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21551–21561, 2024. 3

[22] Jiawei Yang, Marco Pavone, and Yue Wang. FreeNeRF: Improving few-shot neural rendering with free frequency regularization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8254–8263, 2023. 2, 3, 4, 7, 8, 1

[23] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *CVPR*, 2021. 2, 3

[24] Zehao Zhu, Zhiwen Fan, Yifan Jiang, and Zhangyang Wang. Fsgs: Real-time few-shot view synthesis using gaussian splatting. In *European conference on computer vision*, pages 145–163. Springer, 2025. 2, 3

FourieRF: Few-Shot NeRFs via Progressive Fourier Frequency Control

Supplementary Material

A. Supplementary Results on NeRF Synthetic and LLFF dataset

First, we note that as part of the supplementary materials we have **included a interactive website**, which provides qualitative comparisons with two closest baselines to our method (methods that have comparable running times): TensoRF [3] and ZeroRF [18]. As can be seen in the provided website, our method significantly outperforms those baselines in terms of quality of reconstructions in the few-shot setting, and has fewer artefacts, especially when considering very few input views. We provide qualitative comparisons both in terms of the novel view synthesis (RGB) as well as depth estimation compared to these baseline approaches. We encourage the reader to consider the videos provided in the interactive website (please allow a few seconds to load the videos) to see the improvement provided by our method.

In addition, below we include details on the statistics of our evaluations on the LLFF dataset in tables 4,5,6 and on the NeRF synthetic dataset in tables 7 and 8. For the LLFF dataset we reproduced the ZeroRF experiments to obtain the per scene score.

B. Details on Implementation and settings

Finally, we provide details surrounding the settings of our implementation and experiments. All of our experiments were run in an nvidia RTX 4090 graphics card. We build our code base in top of the TensoRF [3] repository. Our repository can be found in the following link: `https://github.com/diego1401/FourieRF`.

Our method uses the AdamW optimizer [9, 11] with $\beta_1 = 0.9, \beta_2 = 0.98$, and a weight decay of 0.2 for synthetic scenes and 0 for real scenes. When performing frequency-control on real scenes we have to deal with matter appearing in front of the camera as a form of overfitting, as highlighted by FreeNeRF [22]. We find that applying their occlusion regularization works without any modification in our pipeline, thus we use their hyper parameters to compute our metrics. Moreover, we note that it is also efficient to use the gradient scaling introduced in the Floaters No More paper [16], this approach does not require to set a hyper parameter. We train for 10k iterations to match with our baselines, mainly ZeroRF [18].

The key hyper parameters of our method differ in the synthetic and real datasets. This can be attributed to the fact that the synthetic dataset has a solid color, white background, which alters the behavior of our method.

For the synthetic dataset, the clipping threshold is initialized as $f_0 = 0.3$, and it is linearly increased with $\delta = \frac{1}{2000} = 2 \times 10^{-3}$. We use the same configuration parameters as TensoRF [3] with the following differences. We apply a TV loss (with weight $w_{TV} = 1.0$) on the appearance and density features. We find that setting the weight decay to 0.2 in the optimizer is the key to removing floaters (in our method and in ZeroRF [18]).

For the real dataset, the clipping threshold is initialized as $f_0 = 0.01$, and it is linearly increased until the end of training, i.e. $\delta = \frac{1}{10000} = 10^{-4}$. We use the same configuration parameters as TensoRF [3] with the following differences. We apply a TV loss (with weight $w_{TV} = 1.0$) on the appearance and density features, and an L1 loss (with weight $w_{L1} = 10^{-4}$) on the density features. We find that applying the L1 loss in this type of scenes is more efficient than setting a weight decay for the optimizer.

Table 4. Details quantitative comparison on the LLFF real dataset 3 views.

Method	Statistic	fortress	room	horns	orchids	leaves	fern	flower	trex	mean
FreeNeRF [22]	PSNR ↑	23.437	22.020	18.506	15.286	16.250	21.187	20.413	19.941	19.630
	SSIM ↑	0.583	0.834	0.585	0.407	0.521	0.662	0.617	0.687	0.612
	LPIPS ↓	0.319	0.190	0.355	0.377	0.350	0.286	0.291	0.297	0.308
ZeroRF [18]	PSNR ↑	20.633	18.833	13.688	13.900	16.275	18.700	17.880	16.786	17.087
	SSIM ↑	0.435	0.663	0.233	0.275	0.533	0.523	0.490	0.517	0.459
	LPIPS ↓	0.386	0.392	0.612	0.527	0.398	0.422	0.423	0.451	0.451
Ours	PSNR ↑	22.109	20.271	18.290	15.103	16.524	20.965	21.062	20.103	19.303
	SSIM ↑	0.573	0.792	0.627	0.422	0.587	0.667	0.674	0.745	0.636
	LPIPS ↓	0.305	0.294	0.336	0.359	0.290	0.271	0.266	0.271	0.299

Table 5. Details quantitative comparison on the LLFF real dataset 6 views.

Method	Statistic	fortress	room	horns	orchids	leaves	fern	flower	trex	mean
FreeNeRF [22]	PSNR ↑	28.728	27.302	23.592	17.263	19.047	24.647	24.665	24.596	23.730
	SSIM ↑	0.832	0.910	0.792	0.555	0.685	0.796	0.797	0.864	0.779
	LPIPS ↓	0.162	0.117	0.218	0.291	0.260	0.196	0.162	0.154	0.195
ZeroRF [18]	PSNR ↑	23.767	27.083	19.188	14.425	18.475	23.533	21.780	21.957	21.276
	SSIM ↑	0.802	0.880	0.606	0.318	0.670	0.753	0.712	0.796	0.692
	LPIPS ↓	0.195	0.211	0.387	0.519	0.319	0.280	0.277	0.279	0.308
Ours	PSNR ↑	29.031	28.792	23.273	17.484	19.187	24.466	24.510	22.019	23.595
	SSIM ↑	0.878	0.920	0.815	0.558	0.727	0.792	0.822	0.810	0.790
	LPIPS ↓	0.144	0.165	0.217	0.313	0.214	0.210	0.174	0.243	0.210

Table 6. Details quantitative comparison on the LLFF real dataset 9 views.

Method	Statistic	fortress	room	horns	orchids	leaves	fern	flower	trex	mean
FreeNeRF [22]	PSNR ↑	29.421	29.927	25.154	19.083	20.678	26.073	26.182	24.522	25.130
	SSIM ↑	0.865	0.938	0.846	0.662	0.756	0.831	0.843	0.875	0.827
	LPIPS ↓	0.124	0.091	0.174	0.237	0.222	0.159	0.133	0.139	0.16
ZeroRF [18]	PSNR ↑	24.350	26.883	21.675	16.125	19.200	24.400	23.240	24.629	22.563
	SSIM ↑	0.797	0.903	0.733	0.465	0.700	0.787	0.762	0.850	0.750
	LPIPS ↓	0.195	0.189	0.314	0.424	0.300	0.242	0.250	0.229	0.268
Ours	PSNR ↑	29.567	29.011	24.799	19.046	20.839	25.774	26.488	24.562	25.011
	SSIM ↑	0.881	0.931	0.860	0.636	0.775	0.825	0.854	0.876	0.830
	LPIPS ↓	0.153	0.171	0.194	0.283	0.200	0.187	0.158	0.198	0.193

Table 7. Details quantitative comparison on the NeRF synthetic dataset 4 views.

Method	Statistic	chair	drums	ficus	hotdog	lego	materials	mic	ship	mean
FreeNeRF [22]	PSNR ↑	20.22	14.99	17.35	23.58	20.43	21.36	15.05	17.52	18.81
	SSIM ↑	0.843	0.746	0.809	0.899	0.818	0.857	0.802	0.687	0.808
	LPIPS ↓	0.109	0.280	0.144	0.108	0.156	0.174	0.218	0.318	0.188
ZeroRF [18]	PSNR ↑	23.04	16.91	20.12	29.11	22.11	20.50	24.76	19.01	21.94
	SSIM ↑	0.880	0.791	0.866	0.944	0.868	0.848	0.944	0.707	0.856
	LPIPS ↓	0.074	0.131	0.100	0.075	0.085	0.132	0.050	0.256	0.113
Ours	PSNR ↑	24.13	17.33	18.56	27.26	22.41	21.15	23.35	19.64	21.73
	SSIM ↑	0.895	0.804	0.848	0.933	0.871	0.858	0.929	0.724	0.858
	LPIPS ↓	0.107	0.206	0.120	0.088	0.122	0.129	0.056	0.283	0.139

Table 8. Details quantitative comparison on the NeRF synthetic dataset 6 views.

Method	Statistic	chair	drums	ficus	hotdog	lego	materials	mic	ship	mean
FreeNeRF [22]	PSNR ↑	26.72	18.16	18.46	27.18	24.32	21.63	25.64	20.23	22.77
	SSIM ↑	0.916	0.827	0.840	0.929	0.887	0.853	0.942	0.729	0.865
	LPIPS ↓	0.071	0.176	0.161	0.096	0.132	0.202	0.066	0.290	0.149
ZeroRF [18]	PSNR ↑	27.62	20.88	22.21	29.93	26.26	21.41	27.40	22.13	24.73
	SSIM ↑	0.926	0.869	0.898	0.949	0.913	0.849	0.954	0.756	0.889
	LPIPS	0.074	0.131	0.100	0.075	0.085	0.132	0.050	0.256	0.113
Ours	PSNR ↑	26.62	19.30	19.43	28.84	27.09	21.46	25.78	22.89	23.93
	SSIM ↑	0.918	0.838	0.860	0.939	0.915	0.856	0.942	0.767	0.879
	LPIPS ↓	0.095	0.182	0.124	0.108	0.103	0.141	0.072	0.261	0.136