Multiscale NeRF

Jiayin Meng

UC San Diego

Abstract

Neural Radiance Fields (NeRF) has demonstrated strong performance in scene reconstruction and novel view synthesis. However, one limitation of vanilla NeRF is the aliasing artifacts when rendering images at different spatial resolutions. Mip-NeRF addresses this issue by integrating positional encoding over a conical frustum instead of sampling along a single ray, enabling scale-aware rendering.

In this project, we investigate a simple alternative: explicitly conditioning the NeRF network on approximate pixel footprint while retaining the original architecture in vanilla NeRF. The goal is to isolate the role of explicit scale information and evaluate whether it alone can mitigate aliasing effects.

We first reimplement a full coarse-to-fine NeRF pipeline in PyTorch and then introduce the proposed scale-conditioned modification, evaluating it under a multiscale rendering setup. Our results show that the proposed approach achieves performance comparable to Mip-NeRF in terms of PSNR and SSIM. These observations provide insight into the role of explicit scale information in mitigating aliasing in neural radiance fields.

1. Introduction

Neural Radiance Fields (NeRF) [2] has emerged as a highly effective approach for learning implicit 3D scene representations from 2D images and synthesizing photorealistic novel views. NeRF represents a scene using a fully-connected neural network that maps a 3D spatial position and a 2D viewing direction to volume density and view-dependent color. Combined with volume rendering, this representation enables high-quality novel view synthesis purely from a collection of RGB images, without requiring explicit 3D geometry.

However, NeRF's rendering procedure has a fundamental limitation in multiscale settings. NeRF casts a single infinitesimal ray per pixel using a fixed positional encoding, which is entirely agnostic to the spatial footprint of the pixel being rendered. When training and test images observe scene content at different resolutions, NeRF renderings exhibit aliasing artifacts, with blurring in close-up shots and aliasing in low resolution views. Mip-NeRF [1] addresses this by casting cones instead of rays and introducing integrated positional encoding (IPE) to featurize the sampled volume. However, Mip-NeRF simultaneously changes both the sampling strategy and the feature representation, making it unclear whether the improvement comes from the integrated positional encoding itself or from the explicit availability of scale information.

In this work, we investigate this question by proposing scale-conditioned NeRF: retaining the full vanilla NeRF architecture and introducing a single modification of appending an encoding of the approximate pixel footprint to the MLP input. We explore two variants: appending the raw scalar directly, and applying positional encoding to the footprint before appending, with the latter yielding better performance. Our contributions are as follows:

- A PyTorch reimplementation of the full NeRF pipeline, including coarse-to-fine hierarchical sampling.
- A scale-conditioned NeRF variant that explicitly conditions the MLP on pixel footprint size while preserving the original architecture, with two variants of scale input: raw scalar and positional-encoded scalar.
- A systematic evaluation on a multiscale benchmark comparing our approach against vanilla NeRF and Mip-NeRF in terms of PSNR, SSIM, and LPIPS.

2. Related Work

We briefly review the two methods most relevant to this work: NeRF and Mip-NeRF.

2.1. Neural Radiance Fields

NeRF [2] represents a scene as a continuous volumetric function parameterized by a fully-connected neural network. Given a 3D position and a 2D viewing direction as input, the network outputs volume density and view-dependent color. Novel views are synthesized by casting rays through the scene, querying the network at sampled points along each ray, and compositing the resulting colors and densities via volume rendering. To enable the net-

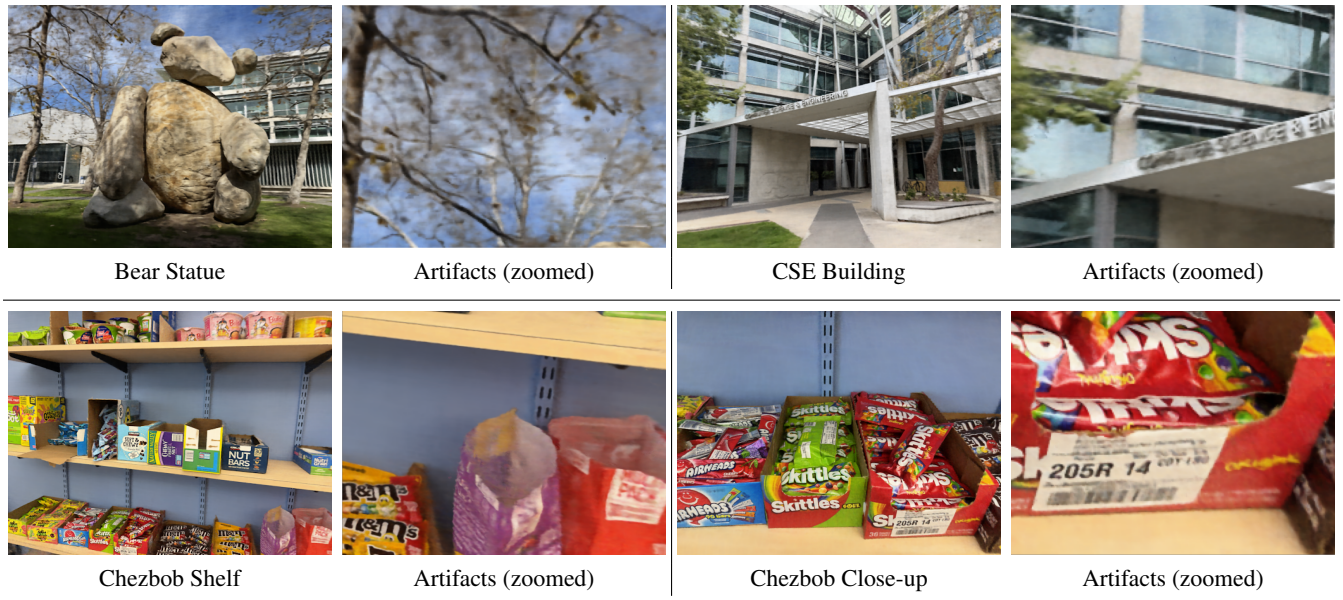

Figure 1. Rendered novel views from our vanilla NeRF reimplementation on four custom scenes. Outdoor scenes (top row) exhibit blurry fine structures and grey discolored artifacts due to large depth ranges and sparse view coverage. Indoor scenes (bottom row) produce significantly cleaner renderings with metrics comparable to standard benchmarks.

work to represent high-frequency scene content, input coordinates are transformed using positional encoding (PE), which maps each coordinate to a higher-dimensional space using sinusoidal functions at multiple frequencies. To improve sampling efficiency, NeRF adopts a coarse-to-fine hierarchical sampling strategy, where a coarse network first estimates the distribution of scene content along each ray, and a fine network then concentrates samples in regions of higher density.

2.2. Mip-NeRF

Mip-NeRF [1] extends NeRF to handle multiscale rendering by addressing its inherent aliasing problem. The key observation is that NeRF's point sampling along a ray is agnostic to the pixel's spatial footprint, causing the same positional encoding features to be produced regardless of the scale at which the scene is observed. Mip-NeRF addresses this by casting a cone per pixel instead of a ray, and dividing the cone into a series of conical frustums. Each frustum is approximated as a multivariate Gaussian, and an integrated positional encoding (IPE) is computed as the expected value of the positional encoding over that Gaussian. This allows the network to reason about both the position and size of each sampled region, enabling scale-aware rendering. Additionally, Mip-NeRF replaces NeRF's separate coarse and fine networks with a single shared MLP, reducing model size by 50% while improving accuracy. On a multiscale variant of the Blender benchmark, Mip-NeRF reduces average error rates relative to NeRF by 60%.

3. Method

We reimplement the full NeRF training pipeline in PyTorch, including coarse-to-fine hierarchical sampling. We briefly summarize the key components below, then describe our proposed scale-conditioned modification.

3.1. NeRF Preliminaries

NeRF represents a scene as a continuous volumetric function parameterized by a fully-connected neural network f_Θ. To render a pixel, a ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ is cast from the camera origin $\mathbf{o}$ along direction $\mathbf{d}$. A set of N points $\{t_k\}$ are sampled along the ray, and each 3D position $\mathbf{x}_k = \mathbf{r}(t_k)$ is transformed via positional encoding (PE):

$$\gamma(\mathbf{x}) = \left[\sin(2^l\pi\mathbf{x}),\ \cos(2^l\pi\mathbf{x})\right]_{l=0}^{L-1} \tag{1}$$

where L is the number of frequency bands. The encoded position, along with the encoded viewing direction $\mathbf{d}$, is passed to the MLP to predict volume density σ_k and color $\mathbf{c}_k$. The final pixel color is computed via volume rendering:

$$\hat{C}(\mathbf{r}) = \sum_{k=1}^{N} T_k \left(1 - \exp(-\sigma_k\delta_k)\right)\mathbf{c}_k \tag{2}$$

where $\delta_k = t_{k+1} - t_k$ is the distance between adjacent samples and T_k is the accumulated transmittance. The model is trained by minimizing the mean squared error between rendered and ground-truth pixel colors. To improve sample efficiency, NeRF employs a coarse-to-fine hierarchical

	PSNR ↑					SSIM ↑					LPIPS ↓				
	Full	1/2	1/4	1/8	Avg	Full	1/2	1/4	1/8	Avg	Full	1/2	1/4	1/8	Avg
Vanilla NeRF	30.8547	30.1156	25.6829	21.9518	27.1512	0.9487	0.9591	0.9294	0.8736	0.9277	0.0702	0.0384	0.0416	0.0830	0.0583
Mip-NeRF	30.8472	31.5347	30.5388	27.7668	30.1719	0.9497	0.9660	0.9695	0.9557	0.9602	0.0300	0.0191	0.0218	0.0360	0.0267
Ours w/o PE	31.2059	30.8776	29.3848	27.8158	29.8210	0.9523	0.9628	0.9590	0.9469	0.9552	0.0651	0.0376	0.0330	0.0508	0.0466
Ours w/ PE	**31.2549**	31.1686	29.9764	**28.4482**	**30.2120**	**0.9525**	0.9633	0.9620	0.9536	0.9579	0.0645	0.0385	0.0328	**0.0328**	0.0432

Table 1. Quantitative results on the *lego* scene.

	PSNR ↑					SSIM ↑					LPIPS ↓				
	Full	1/2	1/4	1/8	Avg	Full	1/2	1/4	1/8	Avg	Full	1/2	1/4	1/8	Avg
Vanilla NeRF	27.6708	28.5113	25.6960	22.2561	26.0336	0.8391	0.8700	0.8756	0.8431	0.8570	0.2116	0.1401	0.1040	0.1052	0.1402
Mip-NeRF	27.5340	29.2235	29.5700	27.8157	28.5358	0.8491	0.8819	0.9148	0.9330	0.8947	0.1561	0.0888	0.0547	0.0360	0.0839
Ours w/o PE	27.8615	28.9600	28.6008	27.6083	28.2577	0.8415	0.8707	0.8891	0.8897	0.8727	0.2067	0.1389	0.0961	0.0857	0.1318
Ours w/ PE	**27.9822**	**29.2450**	29.4786	**28.7852**	**28.8728**	0.8424	0.8722	0.8947	0.9044	0.8784	0.2058	0.1393	0.0984	0.0811	0.1312

Table 2. Quantitative results on the *ship* scene.

sampling strategy with two MLPs. The coarse network estimates the scene density distribution, and the fine network concentrates samples in regions of higher density.

3.2. Scale-Conditioned NeRF

The core observation motivating our approach is that NeRF's positional encoding is entirely agnostic to the scale at which the scene is observed. Two rays sampling the same 3D position at different resolutions produce identical PE features, providing no information to the network about the pixel's spatial footprint.

We propose a simple modification: explicitly appending an approximate pixel footprint as an additional input to the MLP. For a sample point at depth z along a ray with focal length f, we define the pixel footprint as:

$$s = \frac{z}{f} \tag{3}$$

This approximates the world-space size of a pixel at that depth, providing the network with a scale-aware signal at each sample point.

We explore two variants of scale conditioning. In the **raw scalar** variant, s is directly concatenated to the PE features of the 3D position before being passed to the MLP. In the **positional-encoded scalar** variant, s is first transformed with a separate PE with $L = 4$ frequency bands before concatenation:

$$\tilde{s} = \gamma_s(s) = \left[\sin(2^l \pi s),\ \cos(2^l \pi s)\right]_{l=0}^{3} \tag{4}$$

The encoded footprint $\tilde{s}$ is then concatenated with the encoded position $\gamma(\mathbf{x})$ before being passed to the MLP, while the rest of the architecture remains unchanged. In our experiments, the positional-encoded variant yields better performance and is used as our primary model.

3.3. Multiscale Training Setup

To evaluate scale-aware rendering, we construct a multiscale variant of the Blender dataset following Mip-NeRF [1]. Each training image is downsampled by factors of 2, 4, and 8, yielding images at four resolutions: full, $1/2$, $1/4$, and $1/8$. Camera intrinsics are adjusted accordingly for each resolution. During training, rays are sampled randomly across all training images regardless of resolution. Each ray carries the focal length of its corresponding image, which is used to compute the pixel footprint at each sample point as described above.

We evaluate our scale-conditioned NeRF and the vanilla NeRF baseline on the multiscale dataset across all four resolutions. For reference, we also report Mip-NeRF results obtained by running the official implementation via Nerfstudio on the full-resolution Blender dataset. Performance is reported in terms of PSNR, SSIM, and LPIPS.

4. Results

4.1. Vanilla NeRF on Custom Scenes

We validate our NeRF reimplementation on four self-collected real-world scenes: a bear statue, the entrance of the CSE building, a Chezbob vending shelf, and a closer view of the Chezbob shelf. Images were captured with a phone camera and downsampled by a factor of 8 prior to training to make training feasible on available hardware. All models are trained for 200k iterations. Qualitative results are shown in Figure 1.

The indoor scenes (Chezbob shelf and its close-up view) achieve metrics comparable to those reported for LLFF datasets in the original NeRF paper, demonstrating that our reimplementation is correct. The outdoor scenes (bear statue and CSE building entrance), however, yield lower metrics, which we attribute to several factors. First, these

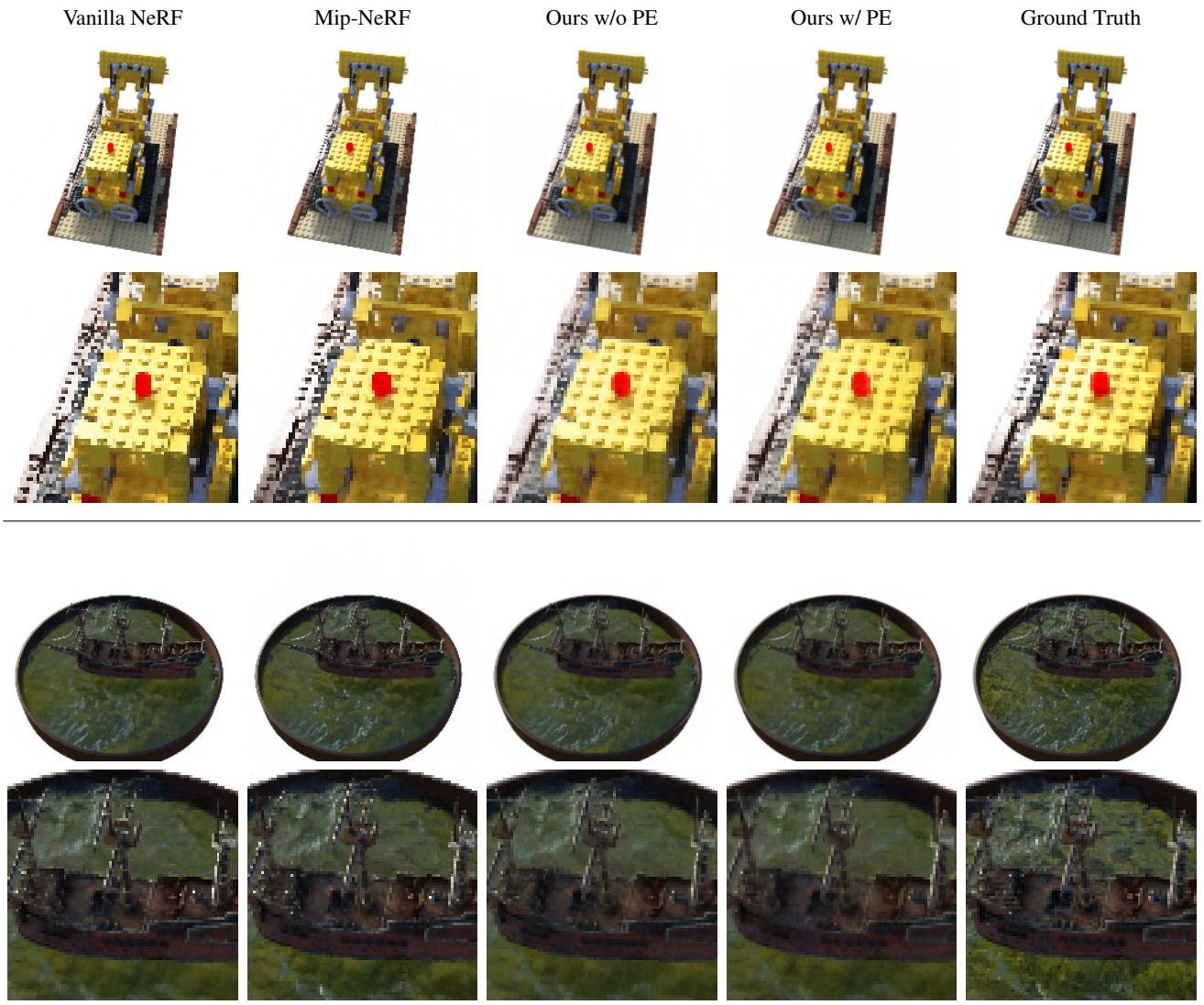

Figure 2. Qualitative comparisons at $1/4$ resolution on the lego (top) and ship (bottom) scenes. Each scene shows a full rendered view (odd rows) and a zoomed-in region (even rows) to highlight fine detail differences across methods.

are unbounded scenes with large depth ranges. Since NeRF uses normalized device coordinates (NDC) for forward-facing real scenes, fewer sample points are allocated to distant regions, leading to blurry reconstructions of fine structures such as tree branches and the sky. Second, training at $1/8$ resolution makes it generally difficult to reconstruct thin structures. Across all four scenes, the camera paths include more occlusions and scene boundaries compared to the structured capture paths used in official benchmarks, resulting in fewer images covering certain regions. Additionally, renderings from novel viewpoints that deviate significantly from the training views exhibit grey or discolored regions, consistent with the known limitations of vanilla NeRF under sparse view coverage.

4.2. Multiscale Evaluation

We evaluate all methods on two scenes from the Blender dataset, lego and ship, under the multiscale setting described in Section 3.3. All models are trained for 100k iterations. Quantitative results are reported in Tables 1 and 2. Qualitative comparisons at $1/4$ and $1/8$ resolution are shown in Figures 2 and 3.

Quantitative Results. As shown in Tables 1 and 2, our scale-conditioned model with positional encoding (Ours w/ PE) achieves performance closely comparable to Mip-NeRF in both scenes in terms of PSNR and SSIM. On the lego scene (see Tables 1), Ours w/ PE achieves an average PSNR of 30.21 dB compared to 30.17 dB for Mip-NeRF.

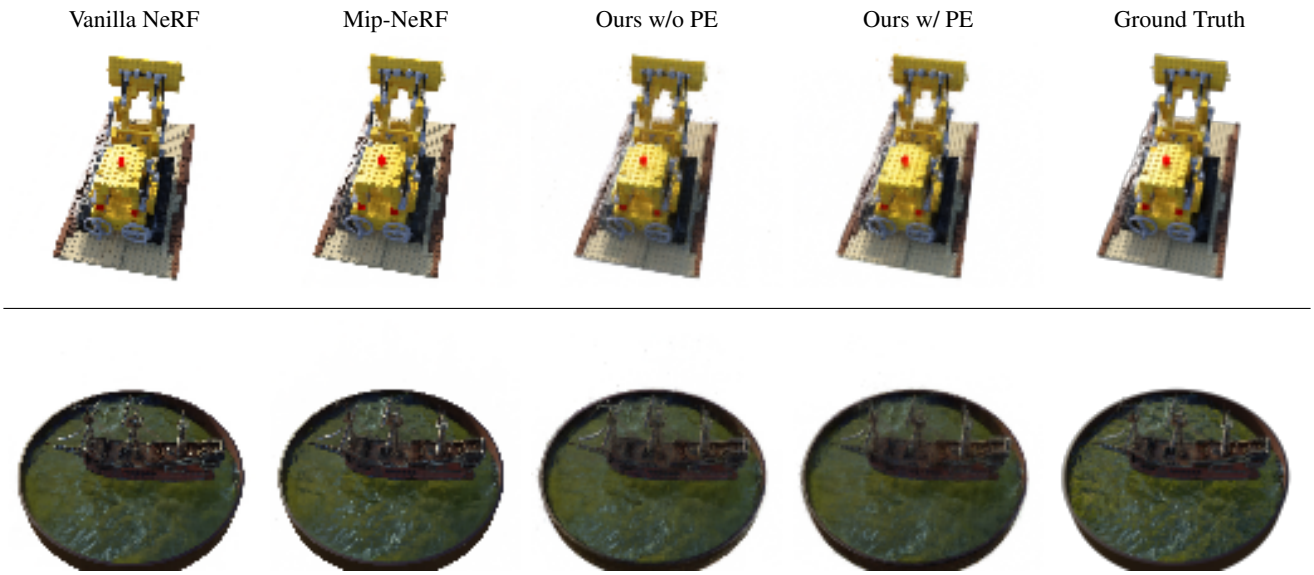

Figure 3. Qualitative comparisons at $1/8$ resolution on the lego (top) and ship (bottom) scenes.

On the ship scene (see Tables 2), Ours w/ PE achieves an average PSNR of 28.87 dB compared to 28.54 dB for Mip-NeRF.

A small gap remains in LPIPS, where Mip-NeRF still holds an advantage on both scenes, suggesting that Mip-NeRF's integrated positional encoding produces perceptually higher-quality renderings despite similar PSNR and SSIM scores.

Comparing the two variants of our method, adding positional encoding to the scale input (Ours w/ PE vs. Ours w/o PE) does not yield a large improvement on any single metric, but consistently improves performance slightly across PSNR, SSIM, and LPIPS on both scenes, suggesting that encoding the scale signal more expressively is generally beneficial.

Notably, Mip-NeRF is evaluated on the full-resolution single-scale dataset via Nerfstudio, while our models are trained and evaluated on the multiscale dataset, making direct comparison approximate.

Qualitative Results. Figures 2 and 3 show qualitative comparisons at $1/4$ and $1/8$ resolution on the lego and ship scenes. At $1/4$ resolution, Vanilla NeRF produces renderings where fine details such as colored regions deviate noticeably from the ground truth. Our scale-conditioned model (Ours w/ PE) better reproduces these details, though the overall renderings appear slightly smoother and less sharp compared to the ground truth. We hypothesize that explicitly providing scale information as input may encourage the network to apply implicit smoothing, similar to a low-pass filter, which reduces aliasing but also slightly blurs fine details. Applying positional encoding to the scale input (Ours w/ PE) produces slightly cleaner results compared to the raw scalar variant (Ours w/o PE). At $1/8$ resolution, as shown in Figure 3, the smoothing effect of our model becomes more apparent.

Full rendered view Zoomed-in region

Figure 4. An example of grey pixel artifacts observed in our scale-conditioned model (Ours w/ PE) on the lego scene at $1/8$ resolution. The zoomed-in region reveals spurious grey pixels generated against the white background, which are not present in the ground truth.

Figure 4 shows an additional artifact observed in our model: spurious grey pixels appearing against white background regions. This artifact likely contributes to the small remaining gap in LPIPS between our model and Mip-NeRF, as LPIPS is sensitive to perceptual differences.

4.3. Discussion

Our results suggest that explicitly providing scale information to the NeRF MLP is beneficial: our scale-conditioned model consistently outperforms Vanilla NeRF across both

scenes and all resolutions in terms of PSNR, SSIM, and LPIPS, and achieves performance closely comparable to Mip-NeRF in terms of PSNR and SSIM. However, a gap in LPIPS remains, and our model introduces additional artifacts at low resolutions, including spurious grey pixels in background regions. This suggests that while explicit scale conditioning is a useful inductive bias, it is not sufficient to fully replicate the perceptual quality achieved by Mip-NeRF's integrated positional encoding.

It is also worth noting that in our experiments, Mip-NeRF is evaluated on the full-resolution single-scale Blender dataset, yet still achieves the best performance at lower resolutions on several metrics. If Mip-NeRF had been trained and evaluated on the multiscale dataset, its performance would likely be even stronger. This suggests that the integrated positional encoding in Mip-NeRF plays an important role beyond simply providing scale information. It fundamentally changes how the network featurizes the sampled volume, enabling more accurate and perceptually consistent rendering across scales. Explicit scale conditioning, as explored in this work, offers a simpler alternative that achieves competitive quantitative performance, but the two approaches are not equivalent in terms of rendering quality.

5. Conclusion

In this work, we investigated whether explicitly providing scale information to the NeRF MLP, without modifying the positional encoding or network architecture, is sufficient to mitigate aliasing in multiscale rendering. We reimplemented the full NeRF pipeline in PyTorch and proposed a scale-conditioned variant that appends an encoding of the approximate pixel footprint to the MLP input. Experiments on the multiscale Blender dataset show that our approach consistently outperforms Vanilla NeRF and achieves PSNR and SSIM comparable to Mip-NeRF, demonstrating that explicit scale conditioning is an effective and simple inductive bias for scale-aware rendering. However, a remaining gap in LPIPS and the presence of spurious artifacts at low resolutions suggest that Mip-NeRF's integrated positional encoding contributes beyond scale information alone. It fundamentally improves how the network featurizes the sampled volume. These findings indicate that explicit scale conditioning and integrated positional encoding are complementary rather than equivalent: scale information alone is beneficial, but the way scale is encoded into the network's feature representation plays an equally important role in achieving high-quality multiscale neural rendering.

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] 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.