Radiance Field Gradient Scaling for Unbiased Near-Camera Training

Anonymous Authors

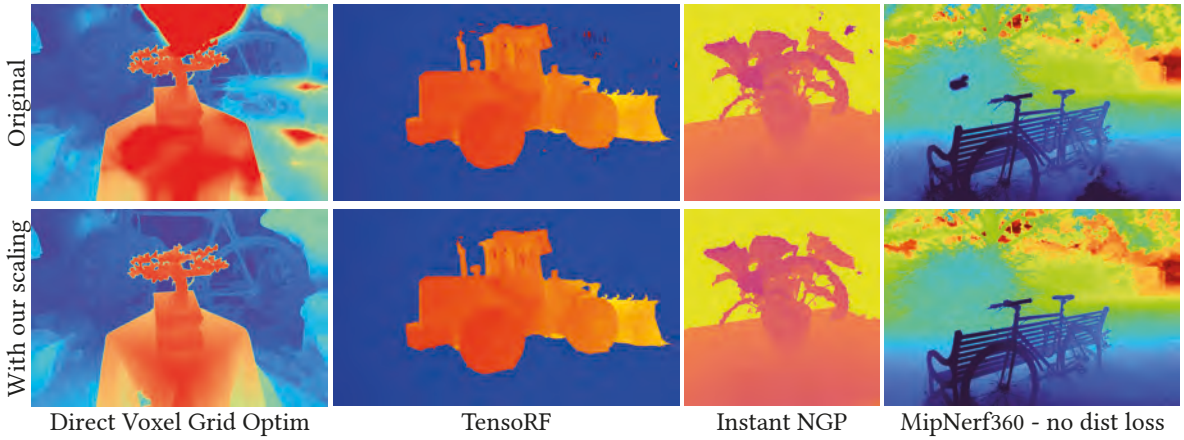


Figure 1: Rendering of different NeRF implementation depths results, without and with our proposed gradient scaling. Most NeRF representations suffer from Background collapse, building density close to the cameras. We propose a simple gradient scaling, which can be easily implemented with any volumetric representation, preventing this build-up. We show examples of DVGO [SSC22a], TensoRF [CXG*22], Instant NGP [MESK22] and MipNerf 360 [BMV*22] (without $\mathcal{L}_{dist}$) without (Original) and with our gradient scaling.

Abstract

NeRF acquisition typically requires careful choice of near planes for the different cameras or suffers from background collapse, creating floating artifacts on the edges of the captured scene. The key insight of this work is that background collapse is caused by a higher density of samples in regions near cameras. As a result of this sampling bias, near-camera volumes receive significantly more gradients, leading to incorrect density buildup. We propose a gradient scaling approach to counter-balance this bias, removing the need for near planes, while preventing background collapse. Our method can be implemented in a few lines, does not induce any significant overhead, and is compatible with most NeRF implementations.

CCS Concepts

• ***Computing methodologies*** → ***Rendering;***

1. Introduction

Neural Radiance Fields (NeRF) [MST*20] introduced a new way to perform 3D reconstruction and rendering of real captured objects and scenes given a set of multi-view images. The NeRF method is based on differentiable volumetric rendering, guiding the presence or absence of density and radiance at every point of the volume. This flexible representation and the impressive reconstruction quality inspired a new line of research, improving the original formulation with respect to a large variety of aspects including reconstruction quality, reconstruction speed, rendering speed, model compression or required memory. Nonetheless, some issues, specifically background collapse and floaters remain present in the reconstruction across most methods, especially for real scenes, pointing towards a more fundamental problem. In this work, we study and propose both an explanation and a simple solution to the problem of background collapse and near-camera floaters.

Background collapse symptoms are very visible floating artifacts appearing close to the training cameras, mistakenly baking some of the background as foreground density. This problem has been identified by previous work [BMV*22] which tackles this issue with an additional term in the loss to force densities to concentrate and be close to a dirac [BMV*22]. While this term indeed results in reduced background collapse it does not explain it and bakes some

priors in the optimization, which might not be suitable for all scenes. Another –often glossed over– detail that plays a role in reducing background collapse is the use of a near plane during ray-marching. It completely prevents gradients from backpropagating towards close camera regions as they are not sampled, but it is scene dependent, has to be manually tuned, and is physically inaccurate. Moreover, for complex capture with varying distances to the subjects, no good scene-wide near plane might exist. On the contrary, we argue that any NeRF method should be able to trace rays directly from the camera without resulting in background collapse or floaters and that the reason we observe such artifacts is mainly due to a bias in the amount of gradient near-camera volume elements receive. We thus propose to scale the gradient during backpropagation to compensate for this bias using a very simple approximation. This scaling allows us to completely remove the need for a near plane while preventing background collapse.

NeRF-based methods mainly differ in their underlying volumetric data structure, which can be as simple as a Multi-Layer Perceptron [BMT*21, BMV*22, MST*20], a voxel hash-grid [MESK22], a tensor decomposition [CXG*22] or even plain voxels [SSC22a]. We show that regardless of the chosen data structure, the reconstructions benefit from our gradient scaling approach.

To summarize our contributions in this work are threefold:

- Identify the root cause of background collapse as a bias in the gradient received by near-camera regions.
- Propose a lightweight gradient scaling solution.
- Demonstrate its effectiveness for several methods with widely varying data structures.

2. Related work

We show that our approach is beneficial to many existing NeRF [MST*20] related methods. We first cover radiance field-based view-synthesis methods before further discussing the problem of background collapse and near-camera bias in the radiance field literature and how it has been addressed so far. For more exhaustive coverage of the fast-paced recent literature, we recommend the recent survey on neural rendering by Tewari et al [TTM*22].

NeRF representations NeRF [MST*20] introduced a different representation for novel view synthesis. Contrary to mesh-based methods that rely on pre-computed geometry and reprojection [OCDD15, HRDB16, HPP*18, RK20, RK21, PMGD21] or point-cloud-based methods [ASK*20, KPLD21, RALB22, KLR*22], radiance field jointly optimize a three-dimensional volumetric density field and a six-dimensional radiance field through differentiable ray-marching and volumetric rendering. This approach has been adapted to use a wide variety of data structures to store the underlying fields providing varying quality, compactness, optimization speed, and rendering speed tradeoffs. The original line of work [MST*20, ZRSK20] used MLPs to represent the scene and was later extended to prevent aliasing [BMT*21], to handle unbounded scenes [BMV*22] and to better represent reflections [VHM*22]. While these methods provide some of the highest-quality results, they are relatively slow to optimize and very slow to render, often requiring several seconds per frame. To act upon these limitations, several works reintroduced some locality in the representation to avoid evaluating a big MLP for each point of space. KiloNeRF [RPLG21] proposed to first train a NeRF before reproducing the optimized field with tiny local MLPs, enabling faster rendering. In a similar spirit, PlenOctrees [YLT*21] bake an octree after training the original NeRF. Hedman et al propose to bake [HSM*21] a pre-trained NeRF to improve rendering speed, however, it does not improve optimization time. Speeding up optimization has been shown to be possible by directly optimizing a grid or voxel-based data structure [STC*22, SSC22a, SSC22b]. In this context of fast optimization, compactness has also been improved using tensor factorization [CXG*22] or hashed-voxels [MESK22] and custom CUDA kernels to enable extremely fast optimization and rendering.

We show that this choice of representation is orthogonal to our contribution and that our gradient scaling can be easily integrated to resolve background collapse by evaluating it on prominent methods with diverse representations.

Floaters, Background collapse and Bias Several works observed and proposed solutions to the problem of floaters and background collapse in radiance field-based methods. Roessle et al. [RBM*22] propose to use depth priors to solve this issue in the context of sparse capture, while in NerfShop, Jambon et al. [JKK*23] acknowledge the problem of floaters and propose an editing method to remove them. MipNeRF360 [BMV*22] proposes a loss that encourages density to concentrate around a single point along the ray, effectively reducing near-camera, semi-transparent radiance. This relatively heavy loss was further made more efficient [SSC22b]. NeRF in the Dark [MHMB*22] also proposes a loss on the weight variance to reduce floaters. FreeNeRF [YPW23] discusses this issue in detail, referring to it as "walls" and "floaters", noting that they are present near the camera and thus propose to penalize density near the camera with an occlusion loss. Using these losses however imposes a prior on the scenes density distribution, which might not be suitable for all content. Further, these methods do not explain the root cause of the phenomenon and why the density builds up close to the camera and not somewhere else. In this work, we provide an explanation and solution to the problem of background collapse and floaters caused by it, by noting that near-camera regions are over-sampled and thus receive more intense and more frequent gradients. A similar sampling problem was identified by Nimier-David et al. [NDMKJ22] in the context of volumetric effect optimization (e.g. smoke) where sampling proportional to the product of transmittance and density leads to patches of density buildup, while only sampling proportional to transmittance significantly reduces artifacts. In our work, we do not modify the sampling but account for its bias in the backpropagation step.

3. Neural Radiance Fields optimization

Most NeRF methods share common components and assumptions to optimize their volumetric representation. Starting from a set of images captured from calibrated

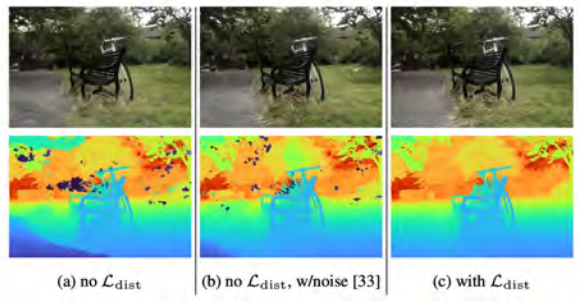


Figure 2: Illustration of the problem of background collapse. $\mathcal{L}_{dist}$ is a loss proposed in MipNeRF360 [BMV*22] to prevent it. (figure credit MipNeRF360 [BMV*22])

cameras, the goal is to optimize an emissive volumetric density to reproduce the appearance of the input pictures when rendered with piece-wise constant volume rendering. The general optimization framework selects pixels in the input training images, generates a ray starting at the camera and towards the chosen pixel, and performs ray-marching by sampling the data structure at discrete positions along the ray to obtain colors and density. The colors and density of these samples are integrated to get a color for each ray cast through that pixel. The aggregation of these colors is finally compared with the original pixel value, resulting in a loss for optimization using stochastic gradient descent.

4. Problematic

Given only a set of calibrated input images, the NeRF reconstruction problem is ill-posed and naive solutions exist, for instance, a planar surface close to each camera reproducing their image content could lead to a zero reconstruction loss. In practice, the nature of the data structures and loss regularizers partially prevent this from happening, but some artifacts often remain. Two of the most prominent are floaters and a phenomenon referred to as *background collapse* where some geometry is reconstructed near the camera. In turn, this incorrectly reconstructed geometry is seen from other viewpoints as floating geometries. In the following section we define the problem (Section 4), analyze the cause of this issue (Section 5) and propose a simple way to solve it (Section 6) .

In Fig. 2 (from MipNeRF360 [BMV*22]) we visualize depth maps in false colors, going from dark blue (close) to red (far). We can see dark blue regions in the depth maps without regularizer loss (a), indicating that some geometry is reconstructed near the camera. In MipNerf360 the authors propose a relatively complex loss to partially solve both floaters and background collapse – both issues being linked as the close camera density will appear as floating geometry from other viewpoints. For a given ray the loss is based on the distance to the camera $\mathbf{s}$ and the weight $\mathbf{w}$ of each interval as:

$$\mathcal{L}_{\text{dist}}(\mathbf{s},\mathbf{w}) = \iint_{-\infty}^{\infty} \mathbf{w}_{\mathbf{s}}(u)\mathbf{w}_{\mathbf{s}}(v)|u-v|d_u d_v \qquad (1)$$

While this loss partially prevents background collapse, it does not explain it and pushes density to be concentrated,

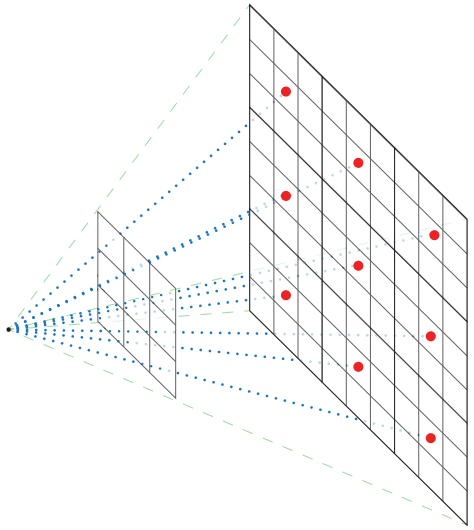


Figure 3: As rays spread from the camera toward the scene, the density of points sampled decreases: all areas were sampled in the first rectangle, while only $\frac{1}{9}$ were in the second (marked with red circles)

which can be problematic when semi-transparent surfaces, represented with partial density, appear in the scene.

Another simpler way to mitigate background collapse is to set a near plane for the rays superior to zero, i.e. the rays do not start from the camera center but a certain distance from it. In practice, this trick is used in most NeRF-related work, but rarely discussed, nor its implications. Using a non-zero near plane has the beneficial effect of preventing any gradient to influence the space contained in the pyramid formed by the camera center and the near plane when optimizing a pixel from this one camera. On the other hand, it prevents reconstruction and rendering in this pyramid, meaning that one should carefully pick the near plane distance. If it is too close to the camera, background collapse may arise, if it is too far, some geometry would be missed during reconstruction and might lead to other artifacts. Indeed, with a near plane positioned too far, the model has to represent the color of the training pixel with density after the near plane. While reasonable values can often be found for the near plane distance, it requires per-scene tuning in the general case. In the case of capture from varying distances from the center of the scene or the various subjects, this near-plane distance may need to be set for every camera independently. Further, when some content is captured very close to the camera, no good value might exist.

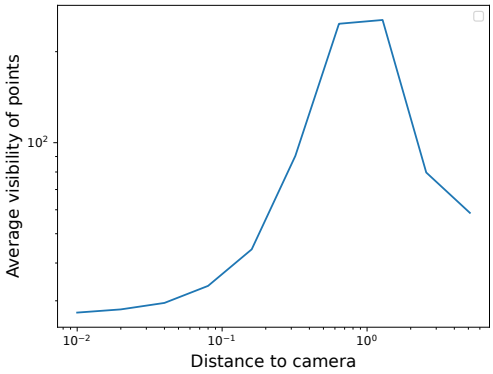
5. Cause

We find that background collapse is primarily caused by a bias in the amount of gradient near-camera volumes receive. As illustrated in Fig. 3, ray casting from a camera is akin to the propagation of light and suffers from a similar quadratic decay. Given a camera and a visible volume element and assuming equally spaced samples along the ray, the density of samples falling in the volume element

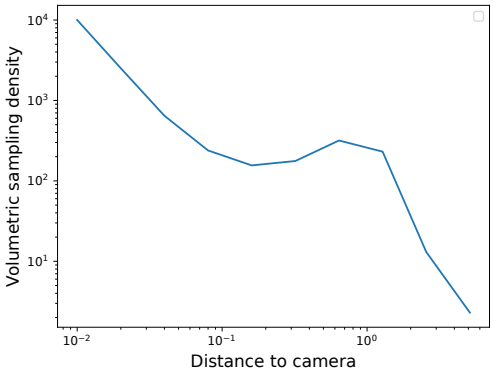
is proportional to the inverse of the square of the distance from that camera. This means that the volume close to the camera is disproportionately more sampled than the rest of it and that near-camera regions receive significantly more gradients per volume element, encouraging a fast build-up of density, and creating floating artifacts.

Indeed, as data structures used to represent radiance fields are generally continuous, a higher sampling rate of volume elements directly translates to stronger and more frequent gradients for the variables used to represent the density and color of the volume. For instance, in the case of a semi-discretized representation, which uses a voxel-like structure [STC*22, MESK22, SSC22a, SSC22b], weights have a local arrangement. In Direct Voxel Grid Optimization (DVGO) [SSC22a] only the eight weights associated with the corners of the voxel containing a sampled point are affected by the backward pass. In these cases, a higher density of sampling directly translates to gradients received more often, and therefore faster updates. This same reasoning can also be applied to the different levels of hash grids in NGP [MESK22]. In the case of MLP-like implicit representations [MST*20, BMT*21, BMV*22], this bias translates to the MLP receiving a lot more signal for near-camera space than elsewhere.

We also note that this bias has the strongest effect early in the training when the low frequencies are not fitted yet. At this early training stage, the gradients are likely to be locally very aligned as they all push toward the same global direction. For instance, if the colors predicted at early iterations for a small volume are varying around grey but the target is red, all points receive approximately the same gradient to change the color to be redder. In such cases, this means that the gradient for a weight influencing a volume element scales linearly with the sampling density of this volume.



(a) Visualisation of the average number of cameras that see a point on a ray as a function of distance to the ray origin. The score is averaged over all cameras in 12 scenes from various methods [BMV*22, CXG*22, SSC22a, MESK22]. Most points near cameras are only seen by a few cameras. Visibility increases until it reaches the average subject distance before decreasing again.



(b) Visualisation of the average volumetric sampling for rays in the Bonsai scene as a function of the distance to the camera. We see that volume units close to the camera are over-sampled despite a low visibility.

Figure 4: Visibility of points on a ray (a) and average volumetric sampling density (b), with respect to distance from a given camera. Both axes use a logarithmic scale.

6. Method

6.1. Sampling in NeRF

Given a pinhole camera c_i, with a view direction $\vec{d}_i$, rays are sampled uniformly for pixels on the image plane. Along those rays, assuming points are sampled linearly, the sampling density at a given point p is given by:

$$\rho_i(p) = \underbrace{v_i(p)}_{\text{Visibility}} \times \underbrace{\frac{|p-c_i|}{\vec{d}_i \cdot (p-c_i)}}_{\text{FOV Sample Variation}} \times \underbrace{\frac{1}{|p-c_i|^2}}_{\text{Distance Decay}} \tag{2}$$

Where $v_i(p)$ is a visibility function (1 if p is in the camera field of view and 0 otherwise). The second term accounts for the lower spatial density of rays on the border while the third accounts for the ray spreading with distance. For reasonable camera FOV, the effect of the second term is negligible in comparison to the distance decay and we can approximate $\frac{|p-c_i|}{\vec{d}_i \cdot (p-c_i)} \approx 1$, giving us:

$$\rho_i(p) = \underbrace{v_i(p)}_{\text{Visibility}} \times \underbrace{\frac{1}{|p-c_i|^2}}_{\text{Distance Decay}} = v_i(p) \times \frac{1}{(\delta_p^i)^2} \tag{3}$$

With δ_p^i the distance between c_i and p.

For a complete scene with n cameras, we can compute the sampling density at a given point p as the sum of density from all cameras:

$$\rho(p) = \sum_{i=0}^{n} v_i(p) \times \frac{1}{(\delta_p^i)^2} \tag{4}$$

The main intuition given by this sum is that for a point, visible and close to a given camera, the sum is dominated by this single camera term, while for points further away

```
class GradientScaler(torch.autograd.Function):
@staticmethod
def forward(ctx, colors, sigmas, ray_dist):
    ctx.save_for_backward(ray_dist)
    return colors, sigmas, ray_dist

@staticmethod
def backward(ctx, grad_output_colors, grad_output_sigmas, grad_output_ray_dist):
    (ray_dist,) = ctx.saved_tensors
    scaling = torch.square(ray_dist).clamp(0, 1)
    return grad_output_colors * scaling.unsqueeze(-1), grad_output_sigmas * scaling,
grad_output_ray_dist
```

Figure 5: PyTorch source code for our gradient scaling operation. `colors`, `sigmas`, `ray_dist` are respectively the per-point color, density, and distance to the cameras. The forward pass is the identity and the backward scales the gradients during back-propagation based on Eq. 6. This module has to be called before integrating the points along the rays.

and at roughly equal distance from the cameras, the visibility term is what plays a significant role. For points near cameras, the inverse squared distance has a very significant impact, while these points tend to only be visible to a few cameras. On the other hand, points around the main subject of the capture tend to be visible by a lot more cameras. This camera visibility phenomenon is illustrated in Fig. 4a. In Fig. 4b we illustrate –on a log scale– the average sampling density along camera rays. We can see that near cameras, the density is decaying quadratically and that despite lower visibility, the volumes close to the camera are disproportionately densely sampled, leading to a disproportionate amount of gradient for these regions.

6.2. Our method

To compensate for the sampling density bias close to cameras, we propose to scale the gradient that per-point characteristics (such as density or color) back-propagate to the NeRF representation (MLP, voxel grid, etc...) during the backward pass. We propose to apply the following gradient scaling:

$$s_{\nabla p} = min(1, (\delta_p^i)^2) \tag{5}$$

i.e we replace ∇p by $\nabla p \times s_{\nabla p}$. Where δ_p^i is the distance between the point and the camera ray origin.

This scaling compensates for the dominating square density close to the cameras while leaving the rest of the gradients unchanged. Note that the scaling for a given point depends on the camera from which the rays are cast.

For this proposed approach we assume that the typical distance between the camera and captured content is in the order of 1 unit of distance. We use this assumption to derive Eq. 6 and did not tune scene scales during our experiments as most scenes respect this assumption. In the case where the scene scale significantly differs from this assumption and the captured content is at an order of σ units of distances, the weighting can be replaced by:

$$s_{\nabla p} = min(1, \frac{(\delta_p^i)^2}{\sigma^2}) \tag{6}$$

6.3. Implementation and performance

Implementing this operation in PyTorch is straightforward using custom `autograd.Function` as shown in the 10 lines of code in Figure 5. We also provide a JAX implementation in supplemental material, directly compatible with the multinerf [MVS*22] codebase. Using this operation, a single call to it can be inserted after the data structure (MLP, hash-grid, voxels, etc) has been sampled, and just before point integration along the ray. This ensures that each point gradient is scaled independently while influencing the weights that control their characteristics (density and color). The code presented in section 6.3, induces an overhead of 100μs in the backward pass for 300k points, which is negligible in the context of $\sim$50ms iterations. It can readily be used in most codebases with minimal adaptations.

7. Evaluations

7.1. Clamped quadratic scaling

Given the nature of the problem, a straightforward candidate could be to scale gradients by $(\delta_p^i)^2$ to compensate for the sampling density quadratic decay. While this indeed solves the near-camera bias problem, it leads to very strong gradients far from the camera, preventing correctly learning surfaces as shown in Figure 6 (middle-row).

As described in equation 4, the sampling density at a given point is the result of the sum of sampling densities from all cameras. This means that the assumption of an inverse quadratic nature of the sampling density for a volume element is mostly valid near each given camera, as sampling from other cameras is negligible in this area. We thus opted not to modify the gradient when reaching the main content of the scene (around a depth of 1) and let the distribution of the camera guide the sampling density in these regions, as illustrated in Fig 4b.

The bias induced by having more cameras seeing a point is a potentially positive bias as it focuses samples in interesting regions as opposed to the near-camera bias which is purely due to the nature of the ray-marching process and produces artifacts. We illustrate in Figure 6 that our gradient scaling approach helps focus the gradient near the

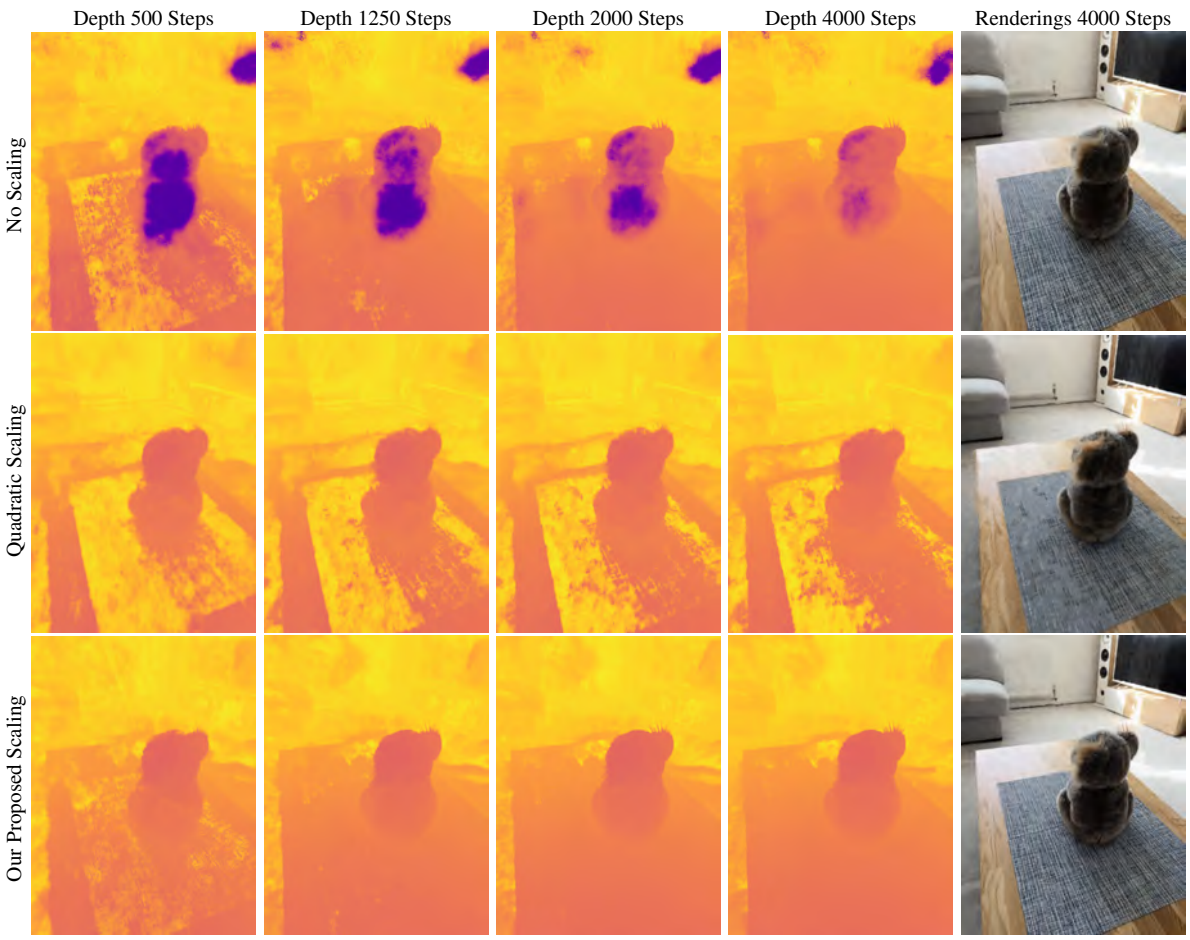


Figure 6: From left to right for each row: Expected Ray depth at 500, 1250, 2000 and 4000 iterations and rendered view at 4000 iterations. Top: No scaling, Middle: Quadratic scaling without clamping, Bottom: our scaling approach. Without Scaling (top row) near camera density arises early on and is never totally removed. With Quadratic Scaling, the table is badly reconstructed with far density and the method only partially recovers. With our Scaling the table is correctly reconstructed early on and no near camera density builds up. The depth is represented here with the plasma color palette, with purple close to the camera and yellow far.

center of the scene, prevents background collapse, and that using a purely quadratic scaling results in worse convergence and reconstruction.

We compare three different approaches of gradient scaling using a custom implementation of NGP [MESK22]. For all of them, we set the near plane to 0. In the top row we can see that without any scaling, density builds up very quickly near the training camera, and while some of it is removed throughout the optimization, some of this close-camera density remains at the end. In the middle row, we can see that scaling purely quadratically leads to a bias toward far density. The table is reconstructed in the background at first and the optimization does not recover from this bad initialization. We present the results of our clamped gradient scaling in the bottom row. We clearly observe the advantage of the method, the geometry of the table is quickly and well reconstructed, and it converges towards a better estimate without background collapse.

7.2. Gradient scaling for various NeRF representation

In this section, we show the effect of adding gradient scaling to various existing methods and show that it removes background collapse effects for all of them. *This is particularly visible in the videos available in Supplemental Materials*. For each method we use their implementation, as much as possible, leading to different color codings of the depth, we define it for each figure.

7.2.1. Direct Voxel Grid Optimization

We use the author's implementation and adjust it to enable our gradient scaling scheme using code similar to Fig 5. In Fig 7 we show a comparison for two scenes with and without gradient scaling. Using our gradient scaling approach completely removes near-camera floaters.

7.2.2. Instant NGP

We use a PyTorch NGP (based on [MESK22]) implementation with the gradient scaling scheme implemented. In Fig 8 we show a comparison for two scenes (Exotic Plant

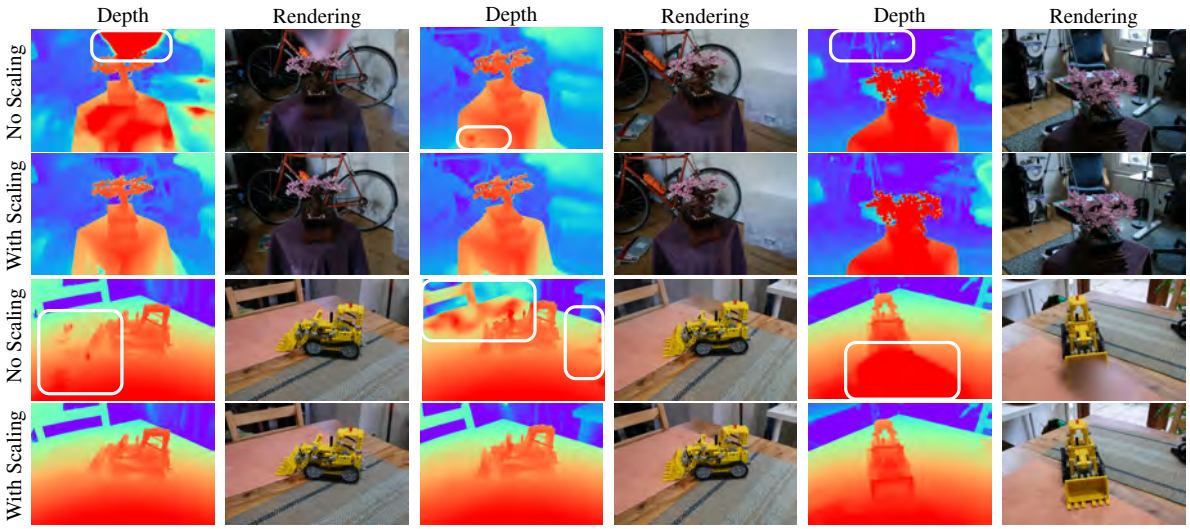


Figure 7: Results of Direct Voxel Grid Optimization [SSC22a] original method (No Scaling) and improved with our proposed gradient scaling. Regions showing significant artifacts are highlighted with white rectangles. We can see that depth is more coherent with our gradient scaling, preventing floaters to appear. The depth is represented here with a warm-cool color palette, with red close to the camera and blue far. This is particularly visible in the top left and bottom right results and in the videos in Supplemental Materials.

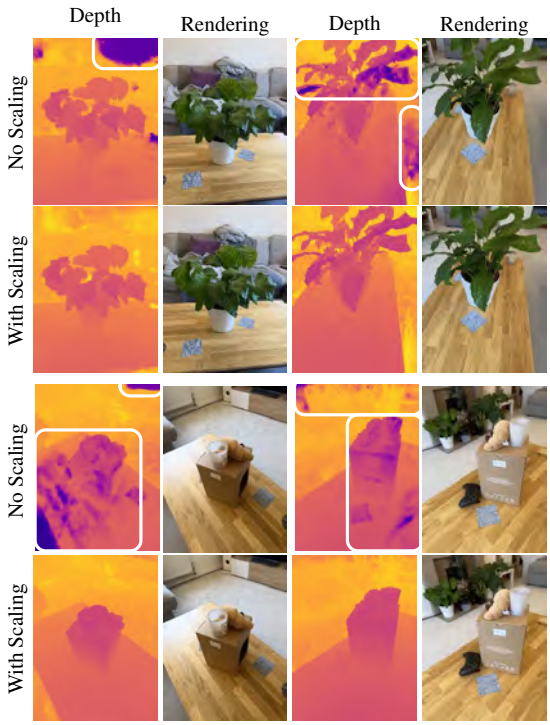


Figure 8: Results of InstantNGP [MESK22] with and without our scaling. **Views shown here are training views**. Regions showing significant artefacts are highlighted with white rectangles. As with other volumetric representations, our proposed scaling helps reconstruct more coherent scenes. This is particularly visible in the videos available in Supplemental Materials. The depth is represented here with the plasma color palette, with purple close to the camera and yellow far.

and Croissant) with and without gradient scaling, this time we visualize training frames and their respective depth after training. This shows the root cause of the problem: density appears very near cameras, perfectly matching the RGB appearance for this given camera, but leading to floaters when seen from novel viewpoints. Using our gradient scaling approach, density does not build up near cameras.

7.2.3. TensoRF

We show in Fig 9 a comparison to TensoRF [CXG*22] and see similar improvement to the other volumetric representations. The results using our scaling show more coherent depth and do not exhibit any floaters.

7.2.4. MipNerf360

We compate to MipNerf360 [BMV*22] in Fig. 10. As MipNerf proposes a loss to limit the problem of background collapse, we show comparisons to different settings. We have three axes we act on, use of the $\mathcal{L}_{\text{dist}}$ proposed by MipNerf360 [BMV*22], setting of the near plane to 0 ($np = 0$) and use of our proposed scaling $s_{\nabla p}$. We can see that combining $\mathcal{L}_{\text{dist}}$ with our scaling $s_{\nabla p}$ yields the best results. As with the other volumetric representations, these effects are most visible in the videos available in Supplemental Materials.

Details MipNerf360 uses a coordinate warping scheme to fit an unbounded space in a bounded box and distributes samples linearly in disparity space to compensate for the contraction of coordinate, using a function $g(x) = 1/x$ to parametrize the rays. Using this same function with a near plane very close to 0 would put most samples just in front of the camera, in our tests when setting the near plane to zero we use $g(x) = 1/(1+x)$. One could also adapt the

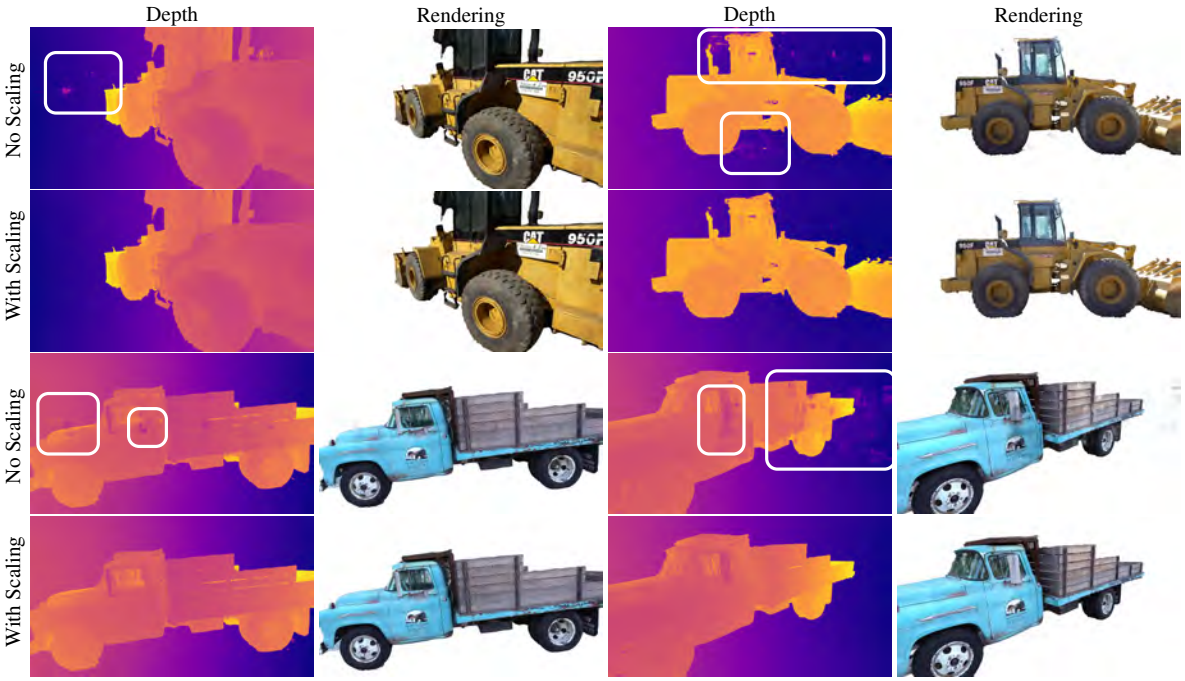


Figure 9: Results of TensoRF [CXG*22] with and without our scaling. Regions showing significant artifacts are highlighted with white rectangles. Here too, our scaling proves efficient in removing floaters due to background-collapse. TensoRF depth visualization attributes blue to red color to the background, the object depth follows the plasma color palette, with purple close to the camera and yellow far.

Method	**Experiment**	**PSNR ↑**	**SSIM ↑**	**LPIPS ↓**
TensoRF	Truck w/o Gradient Scaling	26.92	0.913	0.127
	Truck with Gradient Scaling (Ours)	**27.13**	**0.914**	**0.124**
TensoRF	Caterpillar w/o Gradient Scaling	25.84	0.910	0.136
	Caterpillar with Gradient Scaling (Ours)	**26.19**	**0.912**	**0.133**
DVGO	Kitchen w/o Gradient Scaling	25.84	0.708	0.408
	Kitchen with Gradient Scaling (Ours)	**25.91**	**0.712**	**0.400**
DVGO	Bonsai w/o Gradient Scaling	27.77	0.826	0.404
	Bonsai with Gradient Scaling (Ours)	**28.03**	**0.828**	**0.403**
NGP	Croissant w/o Gradient Scaling	30.94	0.941	0.101
	Croissant with Gradient Scaling (Ours)	**31.22**	**0.943**	**0.097**
NGP	Exotic Plant w/o Gradient Scaling	29.67	0.932	0.112
	Exotic Plant with Gradient Scaling (Ours)	**29.92**	**0.935**	**0.110**
MipNeRF360	Bicycle Original MipNeRF360	**24.79**	**0.685**	**0.223**
	Bicycle No $\mathcal{L}_{\text{dist}}$	24.77	0.680	0.227
	Bicycle No $\mathcal{L}_{\text{dist}}$, $np = 0$, $s_{\nabla p}$	24.65	0.679	0.225
	Bicycle $\mathcal{L}_{\text{dist}}$, $np = 0$, $s_{\nabla p}$ (Ours)	24.75	0.682	**0.223**
MipNeRF360	Stump Original MipNeRF360	26.61	0.743	0.160
	Stump No $\mathcal{L}_{\text{dist}}$	**26.64**	**0.744**	**0.159**
	Stump No $\mathcal{L}_{\text{dist}}$, $np = 0$, $s_{\nabla p}$	26.60	0.741	0.164
	Stump $\mathcal{L}_{\text{dist}}$, $np = 0$, $s_{\nabla p}$ (Ours)	26.62	0.742	0.161

Table 1: We compare PSNR, SSIM, and LPIPS metrics for the evaluated methods([BMV*22,CXG*22,MESK22,SSC22a]) with and without our gradient scaling. Overall our scaling leads to better metrics for all representations, with MipNeRF360 [BMV*22] leading to a similar quality. The floater effect has a relatively low impact on visual metrics, but is perceptually disturbing and clearly visible in the qualitative evaluations and videos in the supplemental material.

scaling and take into account the warping into the spatial sampling bias computation. We found that our method still works as expected without this adjustment, probably because the central part of the scene's coordinates, which are visible by most cameras, are unaffected.

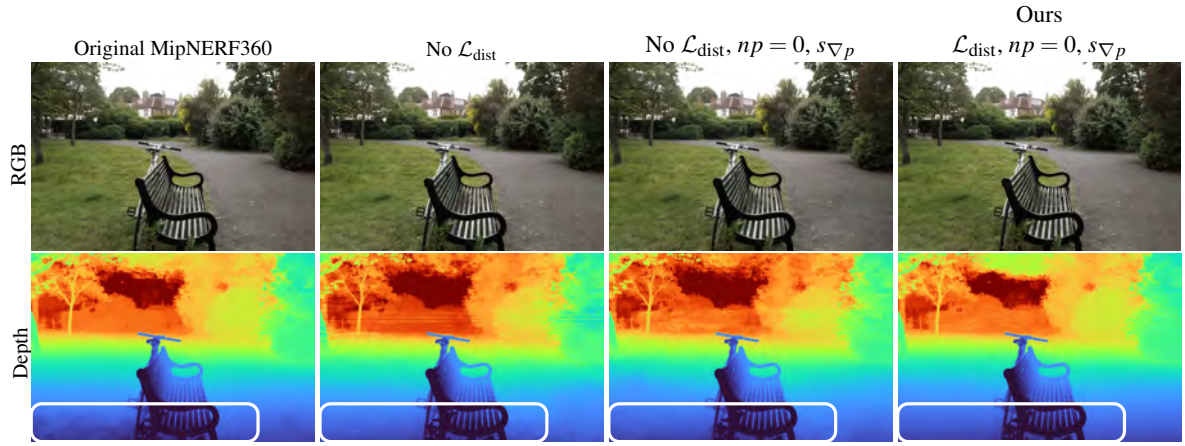


Figure 10: Comparison to MipNERF360 [BMV*22] with different design choice, with and without $\mathcal{L}_{\text{dist}}$, with a near plane at 0.2 or $np = 0$ and with or without our proposed gradient scaling $s_{\nabla p}$. We see here artifacts close to the camera near the floor. Our results combining $\mathcal{L}_{\text{dist}}$ and $s_{\nabla p}$, shows the best reconstruction. The depth is represented with deep blue close to the camera and deep red far. As with other methods, this improvement is particularly visible in the videos available in Supplemental Materials.

7.3. Quantitative evaluation and discussions

We report PSNR, SSIM, and LPIPS [ZIE*18] metrics for the evaluated methods with and without our gradient scaling in Tab. 1 and for the different variants of MipNeRF360. Overall our method leads to better metrics for most tests except for MipNeRF360. While the quantitative difference is small, it mainly depends on the visibility of the floaters in test views, which might be a relatively sparse event depending on the dataset. Further, setting the near plane to zero allocates slightly fewer samples to the center of the scene which can have a slightly negative impact on reconstruction when there is no near-camera geometry. Floaters are more likely to be visible when test cameras are slightly further away from the scene center than training cameras. We find that the qualitative evaluation test paths, shown in Supplemental Material, better show the improvement provided by our approach.

8. Conclusion

We present a simple, yet efficient, gradient scaling approach, removing the need for near-plane setting in NeRF-like methods while preventing background collapse. Our method is computationally efficient and solves a bias existing in most published approaches. This is particularly important in capture scenarios where objects are arbitrarily close or at varying distances from the cameras. Our scaling is directly applicable to most NeRF-like representations and can be easily integrated with a few lines of code.

References

[ASK*20] ALIEV K.-A., SEVASTOPOLSKY A., KOLOS M., ULYANOV D., LEMPITSKY V.: Neural point-based graphics. arXiv:1906.08240v3. 2

[BMT*21] BARRON J. T., MILDENHALL B., TANCIK M., HEDMAN P., MARTIN-BRUALLA R., SRINIVASAN P. P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields, 2021. arXiv:2103.13415. 2, 4

[BMV*22] BARRON J. T., MILDENHALL B., VERBIN D., SRINIVASAN P. P., HEDMAN P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022), 5460–5469. 1, 2, 3, 4, 7, 8, 9

[CXG*22] CHEN A., XU Z., GEIGER A., YU J., SU H.: Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision (ECCV)* (2022). 1, 2, 4, 7, 8

[HPP*18] HEDMAN P., PHILIP J., PRICE T., FRAHM J.-M., DRETTAKIS G., BROSTOW G.: Deep blending for free-viewpoint image-based rendering. 257:1–257:15. 2

[HRDB16] HEDMAN P., RITSCHEL T., DRETTAKIS G., BROSTOW G.: Scalable Inside-Out Image-Based Rendering. 231:1–231:11. 2

[HSM*21] HEDMAN P., SRINIVASAN P. P., MILDENHALL B., BARRON J. T., DEBEVEC P.: Baking neural radiance fields for real-time view synthesis. *ICCV* (2021). 2

[JKK*23] JAMBON C., KERBL B., KOPANAS G., DIOLATZIS S., LEIMKÜHLER T., DRETTAKIS G.: Nerfshop: Interactive editing of neural radiance fields. *Proceedings of the ACM on Computer Graphics and Interactive Techniques 6*, 1 (May 2023). URL: https://repo-sam.inria.fr/fungraph/nerfshop/. 2

[KLR*22] KOPANAS G., LEIMKÜHLER T., RAINER G., JAMBON C., DRETTAKIS G.: Neural point catacaustics for novel-view synthesis of reflections. *ACM Transactions on Graphics 41*, 6 (2022), Article–201. 2

[KPLD21] KOPANAS G., PHILIP J., LEIMKÜHLER T., DRETTAKIS G.: Point-based neural rendering with per-view optimization. In *Computer Graphics Forum* (2021), vol. 40, Wiley Online Library, pp. 29–43. 2

[MESK22] MÜLLER T., EVANS A., SCHIED C., KELLER A.: Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph. 41*, 4 (July 2022), 102:1–102:15. URL: https://doi.org/10.1145/3528223.3530127, doi:10.1145/3528223.3530127. 1, 2, 4, 6, 7, 8

[MHMB*22] MILDENHALL B., HEDMAN P., MARTIN-BRUALLA R., SRINIVASAN P. P., BARRON J. T.: NeRF in the dark: High dynamic range view synthesis from noisy raw images. *CVPR* (2022). 2

[MST*20] MILDENHALL B., SRINIVASAN P. P., TANCIK M., BARRON J. T., RAMAMOORTHI R., NG R.: Nerf: Repre-

senting scenes as neural radiance fields for view synthesis. In *ECCV* (2020). 1, 2, 4

[MVS*22] MILDENHALL B., VERBIN D., SRINIVASAN P. P., HEDMAN P., MARTIN-BRUALLA R., BARRON J. T.: MultiNeRF: A Code Release for Mip-NeRF 360, Ref-NeRF, and RawNeRF, 2022. URL: https://github.com/google-research/multinerf. 5

[NDMKJ22] NIMIER-DAVID M., MÜLLER T., KELLER A., JAKOB W.: Unbiased inverse volume rendering with differential trackers. *ACM Trans. Graph. 41*, 4 (July 2022), 44:1–44:20. URL: https://doi.org/10.1145/3528223.3530073, doi:10.1145/3528223.3530073. 2

[OCDD15] ORTIZ-CAYON R., DJELOUAH A., DRETTAKIS G.: A bayesian approach for selective image-based rendering using superpixels. In *International Conference on 3D Vision (3DV)* (2015), IEEE. URL: http://www-sop.inria.fr/reves/Basilic/2015/ODD15. 2

[PMGD21] PHILIP J., MORGENTHALER S., GHARBI M., DRETTAKIS G.: Free-viewpoint indoor neural relighting from multi-view stereo. *ACM Transactions on Graphics* (2021). URL: http://www-sop.inria.fr/reves/Basilic/2021/PMGD21. 2

[RALB22] RAKHIMOV R., ARDELEAN A.-T., LEMPITSKY V., BURNAEV E.: Npbg++: Accelerating neural point-based graphics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2022), pp. 15969–15979. 2

[RBM*22] ROESSLE B., BARRON J. T., MILDENHALL B., SRINIVASAN P. P., NIESSNER M.: Dense depth priors for neural radiance fields from sparse input views. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2022). 2

[RK20] RIEGLER G., KOLTUN V.: Free view synthesis. In *European Conference on Computer Vision* (2020). 2

[RK21] RIEGLER G., KOLTUN V.: Stable view synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2021). 2

[RPLG21] REISER C., PENG S., LIAO Y., GEIGER A.: Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps, 2021. arXiv:2103.13744. 2

[SSC22a] SUN C., SUN M., CHEN H.: Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR* (2022). 1, 2, 4, 7, 8

[SSC22b] SUN C., SUN M., CHEN H.: Improved direct voxel grid optimization for radiance fields reconstruction. *arxiv cs.GR 2206.05085* (2022). 2, 4

[STC*22] SARA FRIDOVICH-KEIL AND ALEX YU, TANCIK M., CHEN Q., RECHT B., KANAZAWA A.: Plenoxels: Radiance fields without neural networks. In *CVPR* (2022). 2, 4

[TTM*22] TEWARI A., THIES J., MILDENHALL B., SRINIVASAN P., TRETSCHK E., YIFAN W., LASSNER C., SITZMANN V., MARTIN-BRUALLA R., LOMBARDI S., SIMON T., THEOBALT C., NIESSNER M., BARRON J. T., WETZSTEIN G., ZOLLHÖFER M., GOLYANIK V.: Advances in neural rendering. *Computer Graphics Forum 41*, 2 (2022), 703–735. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14507, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14507, doi:https://doi.org/10.1111/cgf.14507. 2

[VHM*22] VERBIN D., HEDMAN P., MILDENHALL B., ZICKLER T., BARRON J. T., SRINIVASAN P. P.: Ref-NeRF: Structured view-dependent appearance for neural radiance fields. *CVPR* (2022). 2

[YLT*21] YU A., LI R., TANCIK M., LI H., NG R., KANAZAWA A.: Plenoctrees for real-time rendering of neural radiance fields. In *arXiv* (2021). 2

[YPW23] YANG J., PAVONE M., WANG Y.: Freenerf: Improving few-shot neural rendering with free frequency regularization. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2023). 2

[ZIE*18] ZHANG R., ISOLA P., EFROS A. A., SHECHTMAN E., WANG O.: The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR* (2018). 9

[ZRSK20] ZHANG K., RIEGLER G., SNAVELY N., KOLTUN V.: NERF++: Analyzing and improving neural radiance fields. *https://arxiv.org/abs/2010.07492* (2020). 2