A1. More Technical Details

We summarize the detailed procedures of Aug-NeRF in the Algorithm 1.

Algorithm 1 The training pipeline of Aug-NeRF. For simplicity, we assume batch size is 1.

Initialize: Training view images $\mathcal{I} = \{\boldsymbol{I}_i \in \mathbb{R}^M\}_{i=1}^N$ and their associated camera poses $\mathcal{P} = \{\boldsymbol{\phi}_i \in \mathbb{R}^{3\times4}\}_{i=1}^N$. Define neural radiance field $F_\Theta(\boldsymbol{p}, \boldsymbol{\theta}) = (g \circ f(\boldsymbol{p}, \boldsymbol{\theta}), h \circ f(\boldsymbol{p})) : (\boldsymbol{p}, \boldsymbol{\theta}) \mapsto (\boldsymbol{c}, \sigma)$ as in Sec. 4.2.

Cast rays for each pixel in each $\boldsymbol{I}_i$ via inverse projection with respect to $\boldsymbol{\phi}_i$, and obtain a set of rays $\mathcal{R} = \{(\boldsymbol{o}_i, \boldsymbol{d}_i, \boldsymbol{\theta}_i, \widehat{\boldsymbol{C}}_i)\}_{i=1}^{NM}$.
while until convergence **do**
  Randomly pick a ray $(\boldsymbol{o}_i, \boldsymbol{d}_i, \boldsymbol{\theta}_i, \widehat{\boldsymbol{C}}_i) \in \mathcal{R}$
  Generate adversarial perturbations δ_t, $\boldsymbol{\delta}_{xyz}$, $\boldsymbol{\delta}_\theta$, $\boldsymbol{\delta}_f$, $\boldsymbol{\delta}_c$, δ_σ by solving Eqn. 4 using PGD (Eqn. 5) within the corresponding search space in Sec. 4.2.
  # Sample points along rays.
  (Coarse) Sample $K/2$ depth intervals t_k along the rays uniformly
  (Fine) Sample $K/2$ depth intervals t_k via proportional to coarse sampled densities [29].
  for $k \in \{1, \cdots, K\}$ **do**
    $t_k^\dagger = t_k + \delta_{t,k}, \boldsymbol{\theta}_i^\dagger = \boldsymbol{\theta}_i + \boldsymbol{\delta}_\theta$.
    $\boldsymbol{p}_k = \boldsymbol{o}_i + t_k \boldsymbol{d}_i, \boldsymbol{p}_k^\dagger = \boldsymbol{o}_i + t_k^\dagger \boldsymbol{d}_i + \boldsymbol{\delta}_{xyz}$
    $(\boldsymbol{c}_k, \sigma_k) = g \circ f(\boldsymbol{p}_k, \boldsymbol{\theta}_i), h \circ f(\boldsymbol{p}_k)$
    $\boldsymbol{c}_k^\dagger = g(f(\boldsymbol{p}_k^\dagger) + \boldsymbol{\delta}_f, \boldsymbol{\theta}_i^\dagger) + \boldsymbol{\delta}_c$
    $\sigma_k^\dagger = h(f(\boldsymbol{p}_k^\dagger + \boldsymbol{\delta}_f)) + \boldsymbol{\delta}_\sigma$
  end for
  # Volumetric rendering.
  $(\Delta t_k, \Delta t_k^\dagger) = t_k - t_{k-1}, t_k^\dagger - t_{k-1}^\dagger$
  $T(k) = \exp\left(-\sum_{l=1}^{k-1} \sigma_l \Delta t_l\right)$
  $T^\dagger(k) = \exp\left(-\sum_{l=1}^{k-1} \sigma_l^\dagger \Delta t_l^\dagger\right)$
  $\boldsymbol{C}_i = \sum_{k=1}^K T(k)(1 - \exp(-\sigma_k \Delta t_k))\boldsymbol{c}_k$
  $\boldsymbol{C}_i^\dagger = \sum_{k=1}^K T^\dagger(k)(1 - \exp(-\sigma_k^\dagger \Delta t_k^\dagger))\boldsymbol{c}_k^\dagger$
  # Train network.
  $\mathcal{L} = \|\boldsymbol{C}_i - \widehat{\boldsymbol{C}}_i\|_2^2 + \lambda \|\boldsymbol{C}_i^\dagger - \widehat{\boldsymbol{C}}_i\|_2^2$
  Update network parameter Θ via $\nabla_\Theta \mathcal{L}$.
end while

A2. More Experiment Results

Qualitative results on NeRF-Synthetic 360° dataset. We present the constructed test views in Fig. A8 and the learned depth maps in Fig. A9. As shown in Fig. A8, we find that the vanilla NeRF fails to capture the fine-grained details of objects, such as the "*ship net*", while Aug-NeRF demonstrates substantially improved visual qualities.

In the meantime, from the depth maps in Fig. A9, NeRF baseline suffers from severe noises. On the contrary, Aug-NeRF enjoys much more smooth depth maps, which suggests that our proposed triple-level robust augmentations indeed enhance the NeRF's continuity and generate smooth geometry representations.

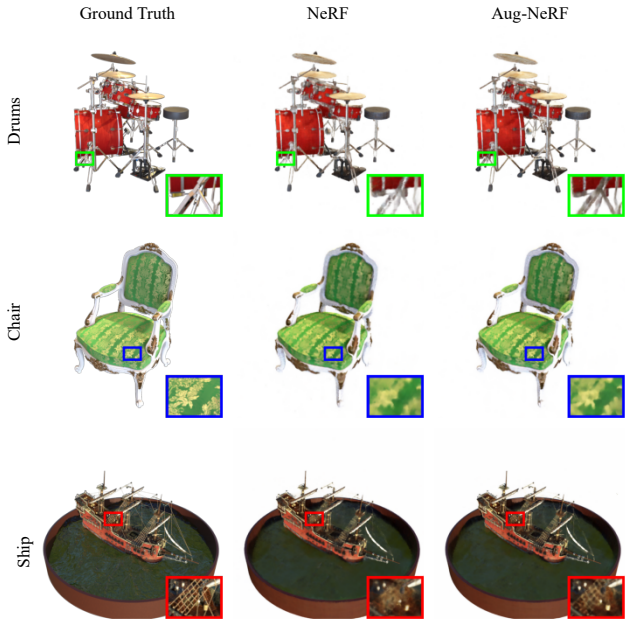

Figure A8. Comparisons on test-set views for scenes from the NeRF-Synthetic 360° dataset generated with a physics-based renderer [29].

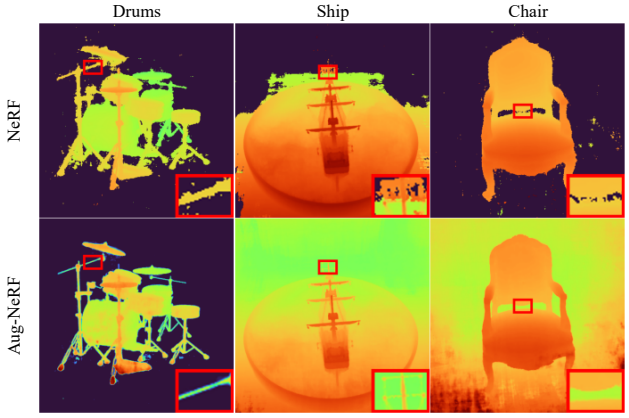

Figure A9. Comparisons of learned depth maps from NeRF and our Aug-NeRF on scenes from NeRF-Synthetic 360° dataset.

Benefits in overfitting vs underfitting cases. We take the scene "chair" as an example, and investigate three combinations of different model sizes and data scales as below Tab. A5: (i) big NeRF (512) on small images ($\frac{1}{2}$ Res.); (ii) normal NeRF (256) on small images ($\frac{1}{2}$ Res.); (iii) small NeRF (128) on large images (Full Res.). We show that in all settings of overfitting / normal case / underfitting, our

proposed augmentations are consistently beneficial.

Table A5. Performance of Aug-NeRF on different backbone and data size combinations.

Setting	Model	PSNR ↑	SSIM ↑	LPIPS ↓	Average ↓
Big NeRF (512)	NeRF	33.56	0.968	0.043	0.015
+ $\frac{1}{2}$ Res.	Aug-NeRF (Ours)	**34.26**	**0.973**	**0.038**	**0.013**
Big NeRF (256)	NeRF	33.27	0.968	0.045	0.016
+ $\frac{1}{2}$ Res.	Aug-NeRF (Ours)	**33.92**	**0.971**	**0.038**	**0.013**
Small NeRF (128)	NeRF	33.00	0.967	0.046	0.016
+ Full Res.	Aug-NeRF (Ours)	**33.86**	**0.970**	**0.041**	**0.014**

Geometry extraction. To obtain geometric visualization in Fig. 5, we first query the network F_Θ with a regular lattice defined over $[-1, 1]^3$, and export a discretized density field volume. The absolute voxel size is 2/512. Then we employ marching cube algorithm [22] provided in UCSF Chimera[2] to extract the surface. We set the threshold to 25 and 1 for chairs and drums, respectively. The step size is chosen as 1. In order to numerically assess the quality of reconstructed geometries, we introduce Chamfer Distance (CD) to measure the difference between reconstructed geometries and ground-truth models:

$$d_{CD} = \frac{1}{|\mathcal{S}_1|}\sum_{\boldsymbol{x}\in\mathcal{S}_1}\min_{\boldsymbol{y}\in\mathcal{S}_2}\|\boldsymbol{x}-\boldsymbol{y}\|_2 + \frac{1}{|\mathcal{S}_2|}\sum_{\boldsymbol{x}\in\mathcal{S}_2}\min_{\boldsymbol{y}\in\mathcal{S}_1}\|\boldsymbol{x}-\boldsymbol{y}\|_2,$$

where $\mathcal{S}_1$ and $\mathcal{S}_2$ are point sets sampled from the extracted surfaces and ground-truth models, respectively. On scene chair, our AugNeRF achieves 1.04×10^{-2} CD which is 29.25% lower than vanilla NeRF (1.47×10^{-2}).

Different types of noise and inaccurate camera poses. As shown in Tab. 3 and Fig. 6, we experiment on two kinds of corruptions, i.e., Gaussian and Shot noises. In this paragraph, we add extra results of training with Pepper noise and inaccurate camera poses are collected in below Tab. A6. The results consistently demonstrate the superiority of Aug-NeRF. We note that our main goal is to endow NeRF with smoothness-aware geometry reconstruction, enhanced generalization to synthesizing unseen views, while the improved tolerance of noisy supervisions is a by-product bonus.

Table A6. Additional results of Aug-NeRF trained on images corrupted by pepper noise and inaccurate camera poses.

Noise Type	"*fern*"	PSNR ↑	SSIM ↑	LPIPS ↓	Average ↓
Pepper Noise	NeRF	19.01	0.401	0.546	0.174
	Aug-NeRF (Ours)	**19.96**	**0.568**	**0.403**	**0.138**
Inaccurate Pose	NeRF	12.31	0.253	0.725	0.333
	Aug-NeRF (Ours)	**13.54**	**0.365**	**0.811**	**0.306**

Implementation of explicit regularization. We investigate three types of explicit regularizations: ℓ_1 sparsity, total variation (TV), and Laplacian. The TV regularization is defined as:

$$R_{\text{TV}}(\Theta) = \int_{[-1,1]^3} |\nabla_{\boldsymbol{x}}\sigma_\Theta(\boldsymbol{x})| d\boldsymbol{x},$$

where σ_Θ denotes the density branch of the function F_Θ. However, evaluating this integral is implausible. Instead, we discretize the integral interval into regular grids and conducting quadrature rule for estimating TV regularization:

$$R_{\text{TV}}(\Theta) = \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}|\nabla_{\boldsymbol{x}}\sigma_\Theta(\delta i, \delta j, \delta k)|\delta^3,$$

where we utilize auto-differentiation provided in PyTorch Library to calculate $\nabla_{\boldsymbol{x}}\sigma_\Theta$. Similarly, we can approximate ℓ_1 sparsity and Laplacian regularization by:

$$\begin{aligned} R_{\ell_1}(\Theta) &= \int_{[-1,1]^3} |\sigma_\Theta(\boldsymbol{x})| d\boldsymbol{x} \\ &\approx \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}|\sigma_\Theta(\delta i, \delta j, \delta k)|\delta^3 \\ R_{\text{Lap}}(\Theta) &= \int_{[-1,1]^3} |\Delta_{\boldsymbol{x}}\sigma_\Theta(\boldsymbol{x})| d\boldsymbol{x} \\ &\approx \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}|\Delta_{\boldsymbol{x}}\sigma_\Theta(\delta i, \delta j, \delta k)|\delta^3 \end{aligned}$$

where $\Delta = \text{div}\cdot\nabla$ denotes the Laplacian operator.

[2] https://www.cgl.ucsf.edu/chimera/