RobustNeRF: Ignoring Distractors with Robust Losses[4] (Supplementary Material)

6.1. Dataset Description

To investigate RobustNeRF and its baselines, we capture and generate a collection of natural and synthetic scenes. With the goal of reconstructing the *static* elements of a scene, we capture frames both with and without distractors present. We describe the details of the capture below.

6.1.1 Natural Scenes

We introduce four natural scenes, two captured in an apartment setting, and two in a robotics lab. See Figure 1 for key details.

	# Clut.	# Clean	# Extra	Paired?	Res.	Setting
Android	122	122	10	No	4032x3024	Apartment
Statue	255	132	19	No	4032x3024	Apartment
Crab	109	109	194	Yes	3456x3456	Robotics Lab
BabyYoda	109	109	202	Yes	3456x3456	Robotics Lab

Figure 1. **Natural Scenes** – Key facts about natural scenes introduced in this work. Includes number of paired photos with (# Clut.) and without (# Clean) distractors. Extra photos (# Extra) do not contain distractors and are taken from unpaired camera poses.

Apartment (Statue & Android). To mimic a casual home scenario, we capture two tabletop scenes in an apartment using a commodity smartphone. Both captures focus on one or more objects on a table top, with photos taken from different viewpoints from a hemisphere of directions around the objects of interest. A subset of objects on the table move from photo to photo as described below. The photos within each scene do not have a clear temporal order.

The capture setup is as follows. We employ an iPhone 12 mini and use ProCamera v15 to control camera exposure settings. We use a fixed shutter speed of 1/60, 0.0 exposure bias, and a fixed ISO of 80 or 200 for the Statue and Android scenes, respectively. We use the iPhone's standard wide lens with an aperture of f/1.6 and resolution of 4032x3024. A tripod is used to reduce the effects of the rolling shutter.

The Android dataset comprises 122 cluttered photos and 10 clean photos (i.e., with no distractors). This scene depicts two Android robot figures standing on a board game box, which in turn is sitting on a table with a patterned table cloth. We pose three small wooden robots atop the table in various ways in each cluttered photo to serve as distractors.

For the Statue scene, we capture 255 cluttered photos and 19 clean photos. The scene depicts a small statue on top of a highly-detailed decorative box on a wooden kitchen table. To simulate a somewhat persistent distractor, we float a balloon over the table which, throughout the capture, naturally changes its position slightly with each photo. Unlike the Android scene, where distractors move to entirely new poses in each frame, the balloon frequently inhabits the same volume of space for multiple photos. The decorative box and kitchen table both exhibit fine grained texture details.

We run COLMAP's [**?**] Structure-from-Motion pipeline using the `SIMPLE_RADIAL` camera model. While COLMAP's camera parameter estimates are only approximate, we find that they are sufficient for training NeRF models with remarkable detail.

The apartment scenes are considerably more challenging to reconstruct than the robotics lab scenes (described below). An accurate NeRF reconstruction must model not only the static, foreground content but also the scene's background. Unlike the foreground, each object in the background is partially over- or underexposed and appears in a limited number of photos. We further found it challenging to maintain a controlled, static scene during capture. As a result, some objects in the background move by a small, unintended amount between photos (e.g., see Figure 3).

Robotics Lab (Crab & BabyYoda). In an effort to control confounding factors in data acquisition, we capture two scenes in a Robotics Lab setting. In these scenes, we employ a robotic arm to randomly position a camera within 1/4 of the hemisphere over a table. The table is placed in a closed booth with constant, indoor lighting. A series of toys are placed on the table, a subset of which are glued to the table's surface to prevent them from moving. Between photos, distractor toys on the table are removed and/or new distractor toys are introduced.

For capture, we use a Blackfly S GigE camera with a TECHSPEC 8.5mm C Series fixed length lens. Photos are center-cropped from their original resolution of 5472x3648 to 3456x3456 to eliminate lens distortion. We capture 12-bit raw photos with an aperture of f/8 and exposure time of 650 ms. Raw photos are automatically color-calibrated afterwards according to a reference color palette.

In each scene, we capture 109 pairs of photos from identical camera poses, one with distractors present and another without. This results in a large number of unique distractors which are challenging to model directly. This further allows us to investigate the counterfactual: What if distractors were *not* present? We further capture an additional $\sim$200 photos from random viewpoints, not aligned with those for training and without distractors, for the purposes of evaluation. In total, because the placement of objects is done manually, one capture session often takes several hours.

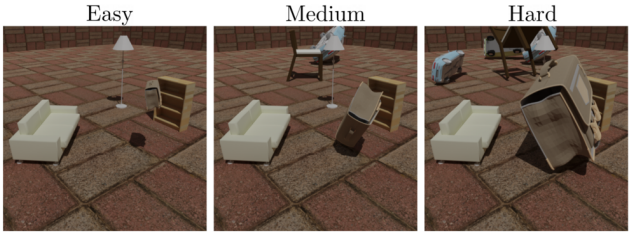

Figure 2. **Synthetic Kubric Scenes** – Example Kubric synthetic images for three datasets with different ratio of outlier pixels. The sofa, lamp, and bookcase are static objects in all three setups. The easy setup has 1 small distractor, the medium setup has 3 medium distractors, and the hard setup has 6 large distractors.

6.1.2 Synthetic Scenes

We generate three Kubric [**?**] scenes similar to the D^{2}NeRF synthetic scenes with different difficulty levels: easy, medium, and hard. These datasets are used to ablate our method with control on the proportion of outlier occupancy (see Sec. 6.3.3).

Each dataset contains 200 cluttered images for training and 100 clean images for evaluation. In all three scenes the static objects include a sofa, a lamp and a bookshelf. Figure 2 shows one example image from the training set for each dataset. The easy scene contains only one small distractor object (a bag). This dataset is similar to Kubric Bag dataset of D^{2}NeRF. The medium scene has three distractors (a bag, a chair, and a car) which are larger in size and hence the outlier occupancy is $4\times$ the outlier occupancy of the easy scene. The hard scene has six large distractors (a bag, a chair, and four cars). They occupy on average $10\times$ more pixels than the easy setup, covering roughly half of each image.

6.2. Training Details

While camera parameters are estimated on the full-resolution imagery, we downsample images by 8x for each natural scene dataset. While mip-NeRF 360 and RobustNeRF are capable of training on high resolution photos, we limit the resolution to accommodate D^{2}NeRF. Unless otherwise stated, we train on all available cluttered images, and evaluate on a holdout set; i.e., 10 images for Android; 19 for the Statue dataset; 194 for Crab; and 202 for the BabyYoda dataset (see Figure 1).

RobustNeRF. We implement RobustNeRF by incorporating our proposed loss function into the MultiNeRF codebase [**?**], replacing mip-NeRF 360's [**?**] reconstruction loss. All other terms in the loss function, such as regularizers, are included as originally published in mip-NeRF 360.

We train RobustNeRF for 250,000 steps with the Adam optimizer, using a batch size of 64 image patches randomly sampled from training images. Each pixel within a 16x16 patch contributes to the loss function, except those identified as outliers (see **??** for a visualization). The learning rate is exponentially decayed from 0.002 to 0.00002 over the course of training with a warmup period of 512 steps.

Our model architecture comprises a proposal Multilayer Perceptron (MLP) with 4 hidden layers and 256 units per layer, and a NeRF MLP with 8 hidden layers, each with 1024 units. We assign each training image a 4-dimensional GLO vector to account for unintended appearance variation. Unless otherwise stated, we use the robust loss hyperparameters given in the main body of the paper. All models are trained on 16 TPUv3 chips over the course of 9 hours.

mip-NeRF 360 [?]. We use the reference implementation of mip-NeRF 360 from the MultiNeRF codebase. Similar to RobustNeRF, we train each variant of mip-NeRF 360 with the Adam optimizer, using the same number of steps, batch size, and learning rate schedule. mip-NeRF 360 uses a random sample of 16384 rays per minibatch. Proposal and NeRF MLP depth and width are identical to those for RobustNeRF. Training hardware and duration are also the same as RobustNeRF.

D^{2}NeRF [?]. We use the reference implementation of D^{2}NeRF [**?**] provided by the authors. Model architecture, hierarchical volume sampling density, and learning rate are the same as published in [**?**]. As in the original work, we train the model for 100,000 iterations with a batch size of 1024 rays, though over the course of 3 hours. Due to hardware availability, we employ four NVIDIA V100 GPUs in place of the A100 GPUs used in the original work.

Images are kept in the order of provided by the file system (i.e., ordered by position information alphanumerically). However, this image order is not guaranteed to represent a continuous path in space since the images were not captured along a continuous path, but rather at random locations. Below we discuss the effects of random ordering versus ordering the views along a heuristically identified path.

D^{2}NeRF training is controlled by five key hyperparameters, namely, skewness (k), which encourages a binarization loss to favor static explanations, and four regularization weights that scale the skewed binarization loss (λ_s), ray regularization loss (λ_r), static regularization loss (λ_{σ^s}), and the view-correlated shadow field loss (λ_ρ). A hyperparameter search is performed in D^{2}NeRF for 16 real world scenes to identify combinations best suited for each scene, and four primary configurations of these parameters are identified as optimal. In particular, the first configuration (i.e., $k = 1.75$, $\lambda_s = 1e^{-4} \rightarrow 1e^{-2}$, $\lambda_r = 1e^{-3}$, $\lambda_{\sigma^s} = 0$, and $\lambda_\rho = 1e^{-1}$) was reported to be most effective across the largest number of scenes real world (10 of 16). We additionally conduct a tuning experiment (see Figure 5) and confirm the first configuration as best suited. We apply this configuration in all additional D^{2}NeRF experiments.

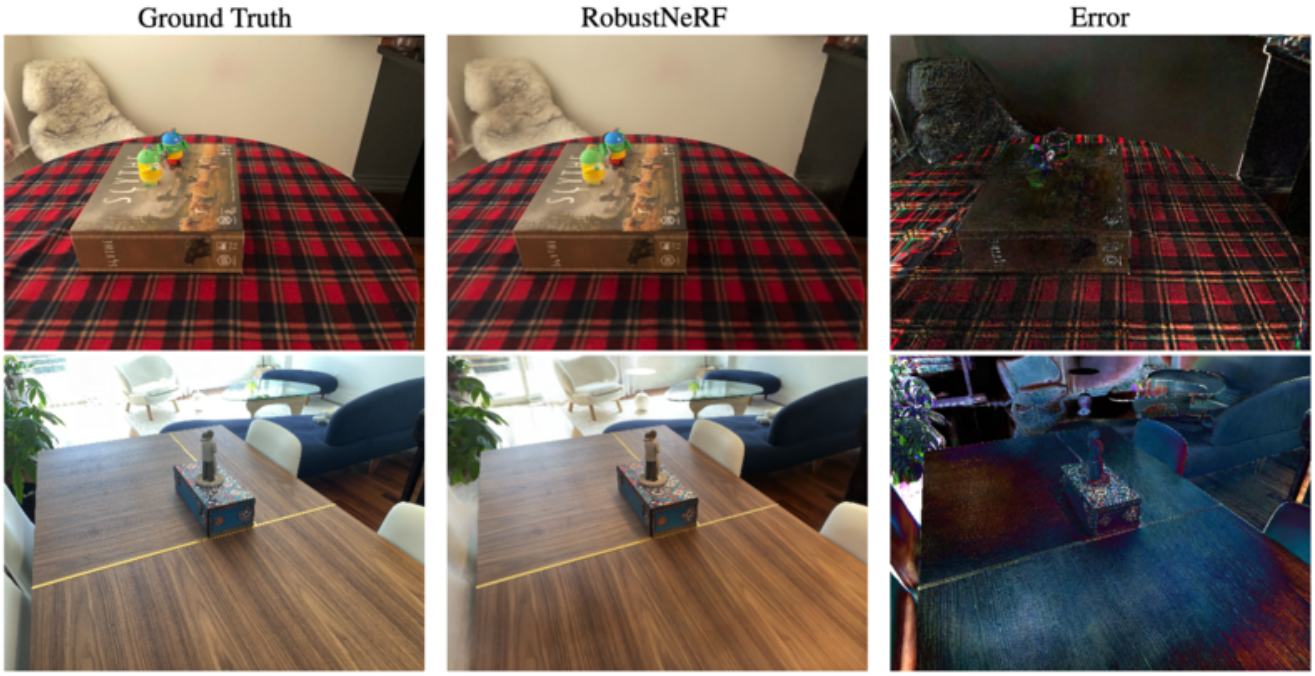

Figure 3. **Challenges in Apartment Scenes –** Each row, from left to right, shows a ground truth photo, a RobustNeRF render, and the difference between the two. Best viewed in PDF. (Top) Note the fold in the table cloth in ground truth image and the lack of fine-grained detail on the covered chair in the background. The table cloth moved during capture, and the background was not captured thoroughly enough for a high-fidelity reconstruction. (Bottom) The ground truth image for the Statue dataset exhibits overexposure and color calibration issues, and hence do not exactly match the RobustNeRF render.

6.3. Experiments

6.3.1 Comparison to mip-NeRF 360

In experiments on natural scenes, as reported in **??**, the performance gap between mip-NeRF 360 (Ch.) and RobustNeRF is markedly higher for the two scenes captured in the robotics lab (i.e., Crab, BabyYoda), compared to those in the apartment (i.e., Statue, Android). We attribute this to the difficulty in reconstructing the apartment scenes, regardless of the presence of distractors. This statement is supported by metrics for reconstruction quality of a mip-NeRF 360 model trained on clean, distractor-free photos. In particular, while mip-NeRF 360 achieves over 32 dB PSNR on Crab and BabyYoda scenes, its PSNR is nearly 10 dB lower on Statue and Android.

Upon closer inspection of the photos and our reconstructions, we identified several reasons for this. First, the apartment scenes contain non-trivial background content with 3D structure. As the background was not the focus of these captures, background content is poorly reconstructed by all models considered. Second, background content illuminated by sunlight is overexposed in some test images (see Figure 3). While this challenge has already been addressed by RawNeRF [**?**], we do not address it here as it is not a focus of this work. Lastly, we find that some static objects were unintentionally moved during our capture. The most challenging form of this is the movement of a table cloth prominently featured in the Android scene which lead to perturbed camera parameter estimates (e.g., see Figure 3).

	Crab			BabyYoda		
Order 1	0.43	0.66	20.19	0.44	0.66	18.17
Order 2	0.42	0.68	20.95	0.44	0.66	17.13

Figure 4. **Effect of Image Order on D^{2}NeRF** – As this model is based on space-time NeRFs [**?**], to make it compatible with our setting we create a 'temporal' indexing of the photos. Here, we visualize: (left) with our heuristic ordering; (right) with another random order. We observe similar distractor-related artifacts in both cases.

	Statue			Crab		
Config 1	0.48	0.49	19.09	0.42	0.68	21.18
Config 2	0.49	0.48	18.20	0.51	0.59	17.02
Config 3	0.51	0.47	18.28	0.46	0.63	19.01
Config 4	0.49	0.48	18.18	0.49	0.58	16.77

Config #	k	λ_s	λ_r	λ_{σ^s}	λ_ρ
Config 1	1.75	1e-4 $\rightarrow$ 1e-2	1e-3	0	1e-1
Config 2	3	1e-4 $\Rightarrow$ 1	1e-3	0	1e-1
Config 3	2.75	1e-5 $\Rightarrow$ 1	1e-3	0	-
Config 4	2.875	5e-4 $\Rightarrow$ 1	0	0	-

Figure 5. **D^{2}NeRF HParam Tuning** – The performance of D^{2}NeRF is heavily influenced by the choice of hyperparameters. In particular, optimal choices of hyperparameters are noted to be strongly influenced by the amount of object and camera motion, as well as video length. We tune by applying four recommended configurations, and identify the first as optimal across the Statue and Crab datasets. Please note that $\rightarrow$ indicates linear increase in value and $\Rightarrow$ indicates exponential increase in value.

6.3.2 Comparison to D^{2}NeRF

Unlike RobustNeRF, D^{2}NeRF makes use of a time signal in the form of provided appearance and warp IDs to generate a code as additional input to the HyperNeRF model. This explicitly models dynamic content alongside the static component of the scene. Two assumptions of D^{2}NeRF are broken in our datasets: 1) the objects sporadically appear (by design); and 2) the views are not necessarily captured in a video-like order. Sporadic object appearance is central to our task, so we do not ablate this property. However, we do evaluate the effect of heuristically reordering camera views according to z position and radial angle of the robotic arm, thereby producing an image order for an imagined "continuous" path. As a control, we pseudorandomly scramble the view order, and train D^{2}NeRF in both settings. The results for BabyYoda and Crab can be seen in Figure 4. We observe no consistent discernable improvement in performance as a result of view reordering and hypothesize that the major hurdle for D^{2}NeRF is rather the modeling of sporadic artifacts.

We also evaluate the effect of applying the four hyperparameter configurations provided by D^{2}NeRF [**?**]. We observe, as expected, that the first configuration performs best across our datasets. Due to limited access to appropriate compute architecture for D^{2}NeRF, we were not able to tune every scene, but selected configuration 1 for all experiments as it performed best in 10/16 real world scenes for D^{2}NeRF as well as tuning experiments on two of our example datasets as see in in Figure 5.

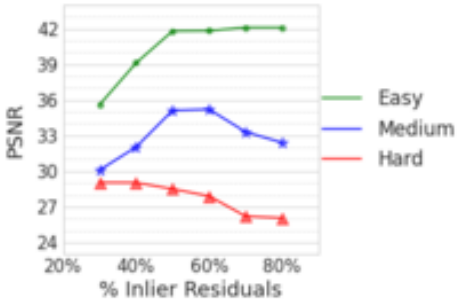

Figure 6. **Sensitivity to $\mathcal{T}_\epsilon$ –** RobustNeRF's reconstruction quality as a function of $\mathcal{T}_\epsilon$ on scenes with different inlier/outlier proportions. Overestimating $\mathcal{T}_\epsilon$ increases training time without affecting final reconstruction accuracy.

Neigh./Patch	4/2	8/4	16/2	16/4	16/8
$\mathcal{T}_\mathcal{R} = 0.6$	18.3	23.08	30.22	30.35	30.75
$\mathcal{T}_\mathcal{R} = 0.8$	28.28	30.7	30.72	30.69	30.72

Figure 7. **Sensitivity to hyper-parameters.** PSNR on distractor-free frames on the Crab dataset as a function of RobustNeRF's neighborhood size, patch size, and $\mathcal{T}_\mathcal{R}$.

6.3.3 Sensitivity to Hyperparameters

We find that the choice of thresholds and filter sizes, described in Section **??**, suffices for a wide range of datasets. As long as the threshold $\mathcal{T}_\epsilon$ is greater than the proportion of outlier pixels in a dataset, RobustNeRF will reliably identify and ignore outlier pixels; see Figure 6. Easy has less than 10% outlier pixels so any $\mathcal{T}_\epsilon$ less than 80% works. In the medium case at least a $\mathcal{T}_\epsilon$ of 60% is required to remove the outliers. In the hard case 44% of pixels are on average occupied so any $\mathcal{T}_\epsilon$ above 50% has worse results. Training with less than 50% of the loss slows down training significantly. Therefore, we observe that after the 250k iterations the model has not converged yet. On average training with 30% of loss requires twice the number of training iterations to catch up. In contrast, D^{2}NeRF requires careful, manual hyperparameter tuning for each scene (e.g., see Figure 5) for several hyperparameters. In our experiments, we found that a single setting of neighborhood and patch sizes works well across all scenes. We present model performance as a function of both hyperparameters on Crab in Figure 7. Larger neighborhood sizes are better regularizers, and we are bounded by the amount of device memory available.

6.3.4 View-dependent effects

We experimentally observed that RobustNeRF performs similarly to mip-NeRF 360 in reconstructing scenes with non-Lambertian materials, semi-transparent objects, and soft shadows. These phenomena are present in the Statue scene (tabletop is *glossy*), and the toys in the Crab and BabyYoda scenes which cast soft *shadows*.

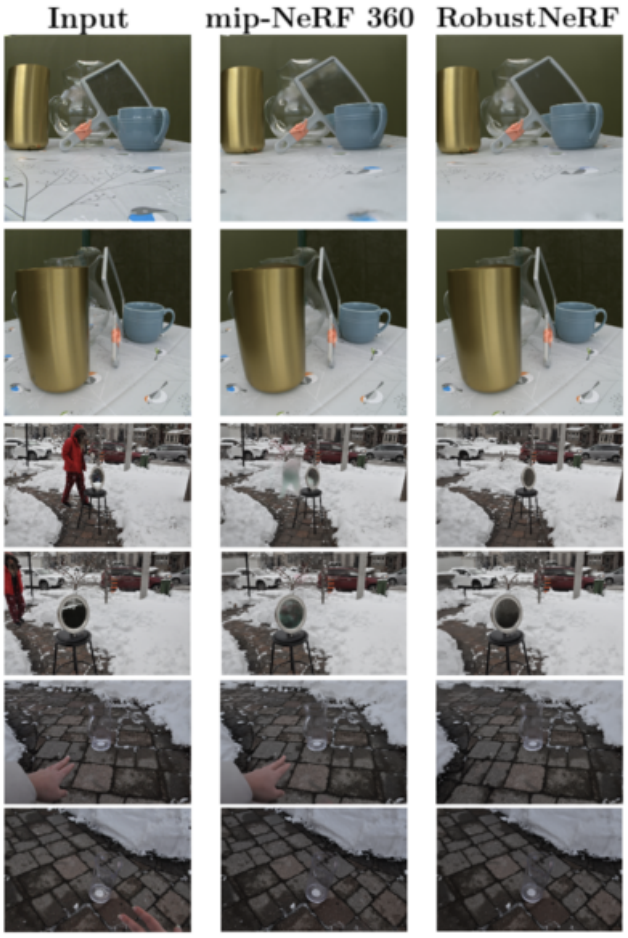

Figure 8. **Qualitative results on scenes with view-dependent effects.** RobustNeRF naturally captures view-dependent effects in scenes with (3rd-6th rows) and without (1st and 2nd row) distractors.

To further emphasize these qualities, we include results for additional scenes with glass, metallic, and reflective objects in Figure 8. The first scene is captured with our Robotic rig, similar to Crab and BabyYoda scenes. It contains a mirror, a shiny cylinder, a transparent vase and a glossy ceramic mug. The other two datasets are captured in the wild. One is with a mirror while pedestrians are moving (as distractions). The last scene contains a transparent pitcher as the object of interest, while the photographer's body parts appear in the photos as the distractors.

6.3.5 More Qualitative Results

We render images from different NerF models from more viewpoints from each of our datasets to further expand the comparison with baselines, D^{2}NeRF, and RobustNeRF. Looking at Figures 10 through 13 one can see that D^{2}NeRF is only able to remove the outliers when there is a single distractor object (Statue dataset) and it fails on the other three

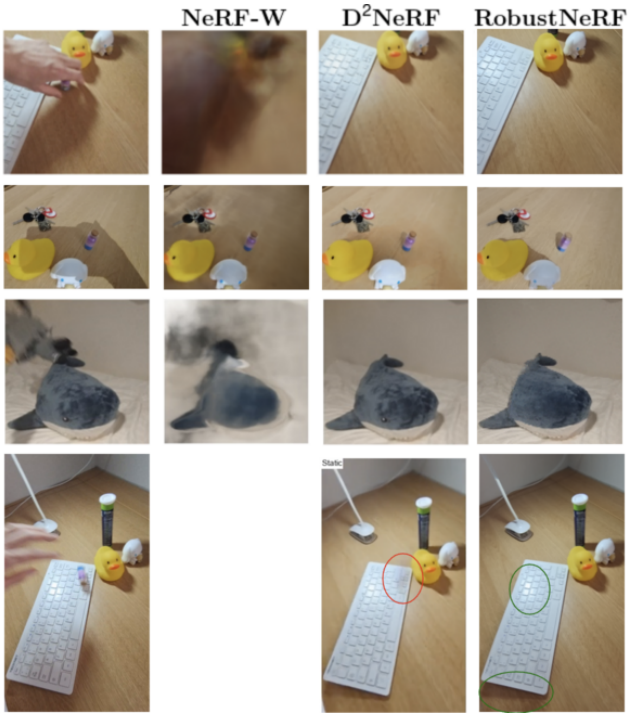

Figure 9. **Qualitative results on D^{2}NeRF Pick scene.** Renders of static model components. Results for NeRF-W and D^{2}NeRF are provided by [**?**]. Note how RobustNeRF naturally captures specular reflections and shadows (green, right).

datasets. The Android dataset has three wooden robots with articulated joints as distractors, and even in this setup where the texture of the distractor objects are similar to one another, D^{2}NeRF fails to fully remove the outliers. In comparison, RobustNeRF is able to remove the outliers irrespective of their number and diversity.

For all four datasets mip-NeRF 360, with either L1, L2, or Charbonnier loss, fails to detect the outliers; one can see 'clouds' or even distinct floaters for these methods. The worst performing loss is L2, as expected. L1 and Charbonnier behave similarly in terms of outlier removal. Changing the loss to RobustNeRF eliminates the floaters and artifacts in all datasets. Video renderings for these scenes are also included in the zipfile with the supplementary material. The floaters in mip-NeRF 360 are easier to resolve in the videos.

We have also experimented on the D^{2}NeRF natural scenes in [**?**]. The qualitative samples are shown in Figure 9. We find that RobustNeRF produces plausible, distraction-free models.

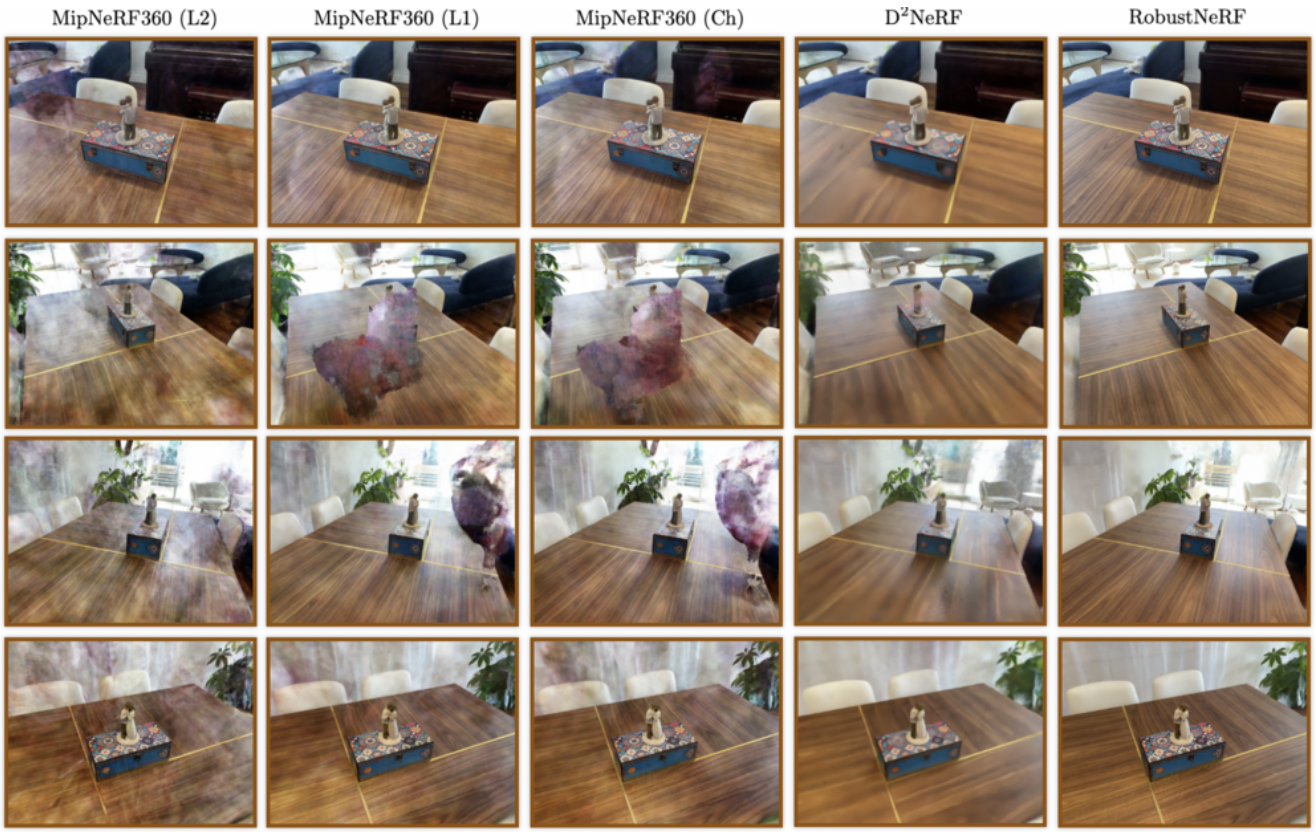

Figure 10. **Statue –** Qualitative results on Statue. It is helpful to zoom in to see details.

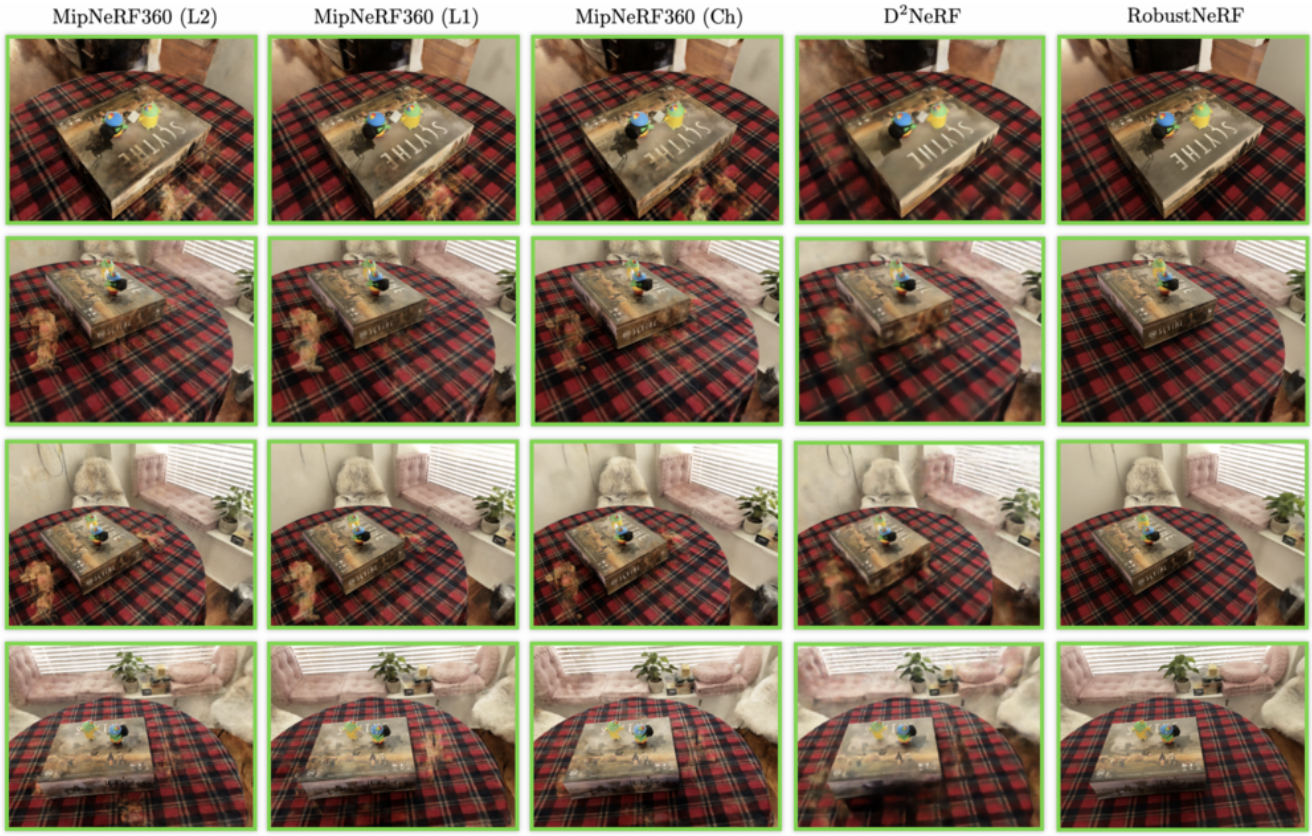

Figure 11. **Android –** Qualitative results on Android. It is helpful to zoom in to see details.

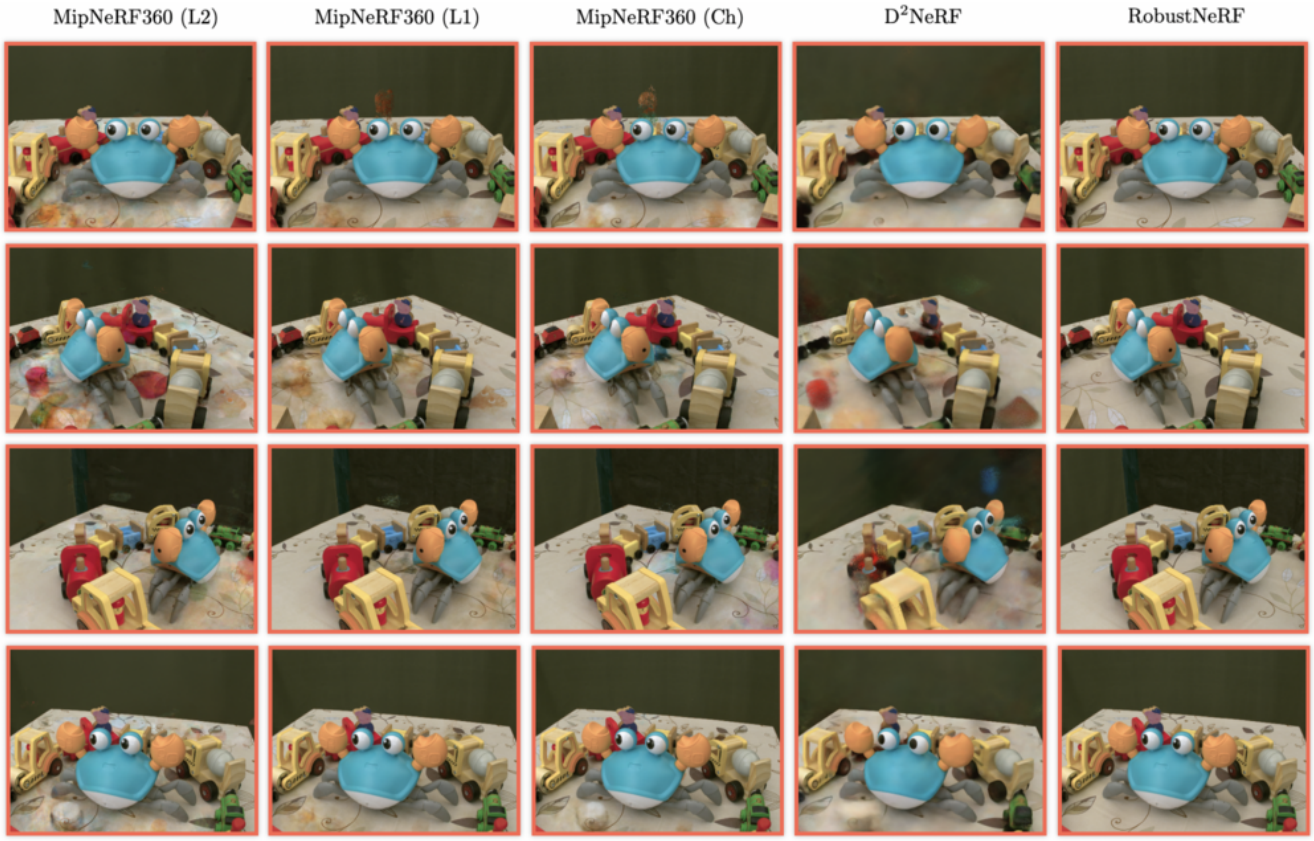

Figure 12. **Crab –** Qualitative results on Crab. It is helpful to zoom in to see details.

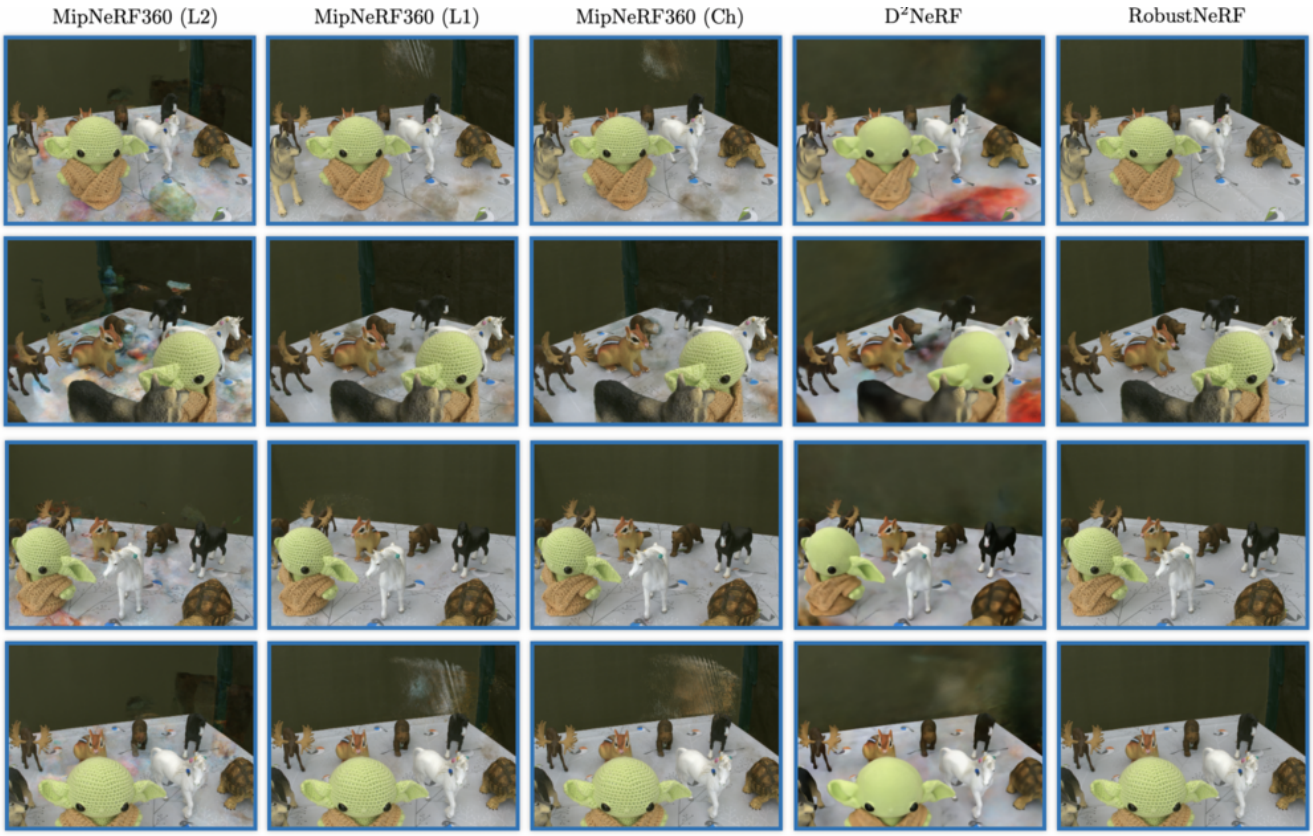

Figure 13. **BabyYoda –** Qualitative results on BabyYoda. It is helpful to zoom in to see details.