RobustNeRF: Ignoring Distractors with Robust Losses[4]

Sara Sabour[1,2] Suhani Vora[1] Daniel Duckworth[1] Ivan Krasin[1]
David J. Fleet[1,2] Andrea Tagliasacchi[1,2,3]
[1]Google Research, Brain Team [2]University of Toronto [3]Simon Fraser University

Abstract

Neural radiance fields (NeRF) excel at synthesizing new views given multi-view, calibrated images of a static scene. When scenes include distractors, which are not persistent during image capture (moving objects, lighting variations, shadows), artifacts appear as view-dependent effects or 'floaters'. To cope with distractors, we advocate a form of robust estimation for NeRF training, modeling distractors in training data as outliers of an optimization problem. Our method successfully removes outliers from a scene and improves upon our baselines, on synthetic and real-world scenes. Our technique is simple to incorporate in modern NeRF frameworks, with few hyper-parameters. It does not assume a priori knowledge of the types of distractors, and is instead focused on the optimization problem rather than pre-processing or modeling transient objects. More results at https://robustnerf.github.io/public.

1. Introduction

The ability to understand the structure of a *static* 3D scene from 2D images alone is a fundamental problem is computer vision [44]. It finds applications in AR/VR for mapping virtual environments [6, 36, 61], in autonomous robotics for action planning [1], and in photogrammetry to create digital copies of real-world objects [34].

Neural fields [55] have recently revolutionized this classical task, by storing 3D representations within the weights of a neural network [39]. These representations are optimized by back-propagating image differences. When the fields store view-dependent *radiance* and volumetric rendering is employed [21], we can capture 3D scenes with photo-realistic accuracy, and we refer to the generated representation as Neural Radiance Fields, or NeRF [25]).

Training of NeRF models generally requires a large collection of images equipped with accurate camera calibration, which can often be recovered via structure-from-motion [37]. Behind its simplicity, NeRF hides several assumptions. As models are typically trained to minimize error in RGB color space, it is of paramount importance that images are photometrically consistent – two photos taken from the same vantage point should be *identical* up to noise. Unless one employs a method explicitly accounting for it [35], one should manually hold a camera's focus, exposure, white-balance, and ISO fixed.

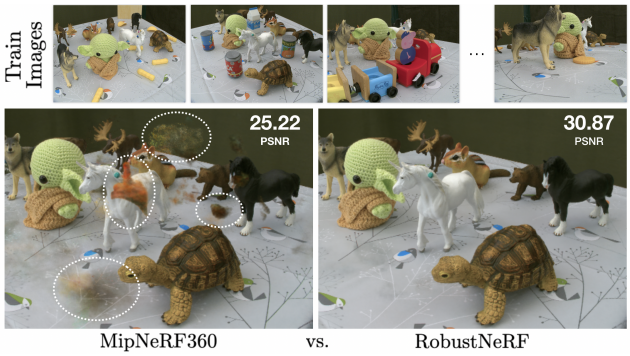

Figure 1. NeRF assumes photometric consistency in the observed images of a scene. Violations of this assumption, as with the images in the top row, yield reconstructed scenes with inconsistent content in the form of "floaters" (highlighted with ellipses). We introduce a *simple* technique that produces *clean* reconstruction by automatically *ignoring distractors without explicit supervision*.

However, properly configuring one's camera is not all that is required to capture high-quality NeRFs – it is also important to avoid *distractors*: anything that isn't persistent throughout the entire capture session. Distractors come in many shapes and forms, from the hard-shadows cast by the operators as they explore the scene to a pet or child casually walking within the camera's field of view. Distractors are tedious to *remove* manually, as this would require pixel-by-pixel labeling. They are also tedious to *detect*, as typical NeRF scenes are trained from hundreds of input images, and the types of distractors are not known a priori. If distractors are *ignored*, the quality of the reconstruction scene suffers significantly; see Figure 1.

In a typical capture session, it is difficult to to capture multiple images of the same scene from the same viewpoint, rendering distractors challenging to model mathematically. As such, while view-dependent effects are what give NeRF their realistic look, *how can the model tell the difference* between a distractor and a view-dependent effect?

[4]Work done at Google Research.

Despite the challenges, the research community has devised several approaches to overcome this issue:

- If distractors are known to belong to a specific class (e.g., people), one can remove them with a pre-trained semantic segmentation model [35, 43] – this process does *not generalize* to "unexpected" distractors such as shadows.
- One can model distractors as per-image *transient* phenomena, and control the balance of transient/persistent modeling [23] – however, it is *difficult to tune* the losses that control this Pareto-optimal objective.
- One can model data in time (i.e., high-framerate video) and decompose the scene into static and dynamic (i.e., distractor) components [53] – but this clearly only applies to *video* rather than photo collection captures.

Conversely, we approach the problem of distractors by modeling them as *outliers* in NeRF optimization.

We analyze the aforementioned techniques through the lens of robust estimation, allowing us to understand their behavior, and to design a method that is not only simpler to implement but also more effective (see Figure 1). As a result, we obtain a method that is straightforward to implement, requires minimal-to-no hyper-parameter tuning, and achieves state-of-the-art performance. We evaluate our method:

- quantitatively, in terms of reconstruction with synthetically, yet photo-realistically, rendered data;
- qualitatively on publicly available datasets (often fine-tuned to work effectively with previous methods);
- on a new collection of natural and synthetic scenes, including those autonomously acquired by a robot, allowing us to demonstrate the sensitivity of previous methods to hyper-parameter tuning.

2. Related Work

We briefly review the basics and notation of Neural Radiance Fields. We then describe recent progress in NeRF research, paying particular attention to techniques for modeling of static/dynamic scenes.

Neural Radiance Fields. A neural radiance field (NeRF) is a continuous volumetric representation of a 3D scene, stored within the parameters of a neural network $\boldsymbol{\theta}$. The representation maps a position $\mathbf{x}$ and view direction $\mathbf{d}$ to a *view-dependent* RGB color and *view-independent* density:

$$\left.\begin{matrix}\mathbf{c}(\mathbf{x},\mathbf{d})\\ \sigma(\mathbf{x})\end{matrix}\right\} f(\mathbf{x},\mathbf{d};\boldsymbol{\theta}) \tag{1}$$

This representation is trained from a collection, $\{(\mathbf{C}_i, \mathbf{T}_i)\}$, of images $\mathbf{C}_i$ with corresponding calibration parameters $\mathbf{T}_i$ (camera extrinsics and intrinsics).

During training the calibration information is employed to convert each pixel of the image into a ray $\mathbf{r}{=}(\mathbf{o},\mathbf{d})$, and rays are drawn randomly from input images to form a training mini-batch ($\mathbf{r}{\sim}\mathbf{C}_i$). The parameters $\boldsymbol{\theta}$ are optimized to correctly predict the colors of the pixels in the batch via the L2 photometric-reconstruction loss:

$$\mathcal{L}_{\text{rgb}}(\boldsymbol{\theta}) = \sum_i \mathbb{E}_{\mathbf{r}\sim\mathbf{C}_i}\left[\mathcal{L}^{\mathbf{r},i}_{\text{rgb}}(\boldsymbol{\theta})\right] \tag{2}$$

$$\mathcal{L}^{\mathbf{r},i}_{\text{rgb}}(\boldsymbol{\theta}) = ||\mathbf{C}(\mathbf{r};\boldsymbol{\theta}) - \mathbf{C}_i(\mathbf{r})||^2_2 \tag{3}$$

Parameterizing the ray as $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$, the NeRF model image $\mathbf{C}(\mathbf{r};\boldsymbol{\theta})$ is generated pixel-by-pixel volumetric rendering based on $\sigma(\cdot)$ and $\mathbf{c}(\cdot)$ (e.g., see [25, 42]).

Recent progress on NeRF models. NeRF models have recently been extended in several ways. A major thread has been the speedup of training [15, 27] and inference [6, 13], enabling today's models to be trained in minutes [27], and rendered on mobile in real-time [6]. While initially restricted to forward-facing scenes, researchers quickly found ways to model real-world 360° scenes [4, 59], and to reduce the required number of images, via sensor fusion [35] or hand-designed priors [28]. We can now deal with image artifacts such as motion blur [22], exposure [24], and lens distortion [14]. And the requirement of (precise) camera calibrations is quickly being relaxed with the introduction of techniques for local camera refinement [8, 19], or direct inference [58]. While a NeRF typically represents geometry via volumetric density, there exist models custom-tailored to predict surfaces [29, 51], which can be extended to use predicted normals to significantly improve reconstruction quality [50, 57]. Given high-quality normals [47], inferring the (rendering) structure of a scene becomes a possibility [5]. We also note recent papers about additional applications to generalization [56], semantic understanding [48], generative modeling [33], robotics [1], and text-to-3D [31].

Modeling non-static scenes. For unstructured scenes like those considered here, the community has focused on reconstructing both static and non-static elements from video. The most direct approach, treating time as an auxiliary input, leads to cloudy geometry and a lack of fine detail [11, 54]. Directly optimizing per-frame latent codes as an auxiliary input has proved more effective [17, 30, 53]. The most widely-adopted approach is to fit a time-conditioned deformation field mapping 3D points between pairs of frames [18, 49] or to a canonical coordinate frame [9, 10, 20, 32, 45]. Given how sparsely space-time is sampled, all methods require careful regularization, optimization, or additional training signals to achieve acceptable results.

Relatively little attention has been given to *removing* non-static elements. One common approach is to segment and ignore pixels which are likely to be distractors [35, 43]. While this eliminates larger objects, it fails to account for secondary effects like shadows. Prior attempts to model distractors as outliers still leave residual cloudy geometry [23].

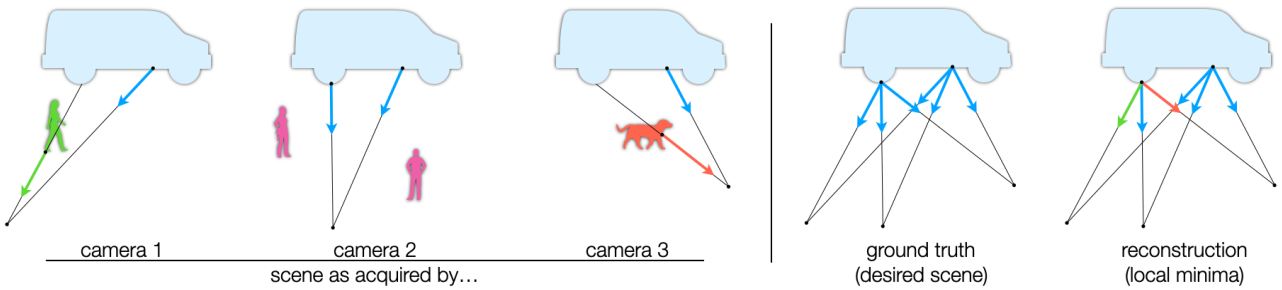

Figure 2. **Ambiguity –** A simple 2D scene where a static object (blue) is captured by three cameras. During the first and third capture the scene is not photo-consistent as a distractor was within the field of view. Not photo-consistent portions of the scene can end up being encoded as view-dependent effects – even when we assume ground truth geometry.

3. Method

The classical NeRF training losses (3) are effective for capturing scenes that are photometrically consistent, leading to the photo-realistic novel-view synthesis that we are now accustomed to seeing in recent research. However, "*what happens when there are elements of the scene that are not persistent throughout the entire capture session?*" Simple examples of such scenes include those in which an object is only present in some fraction of the observed images, or may not remain in the same position in all observed images. For example, Figure 2 depicts a 2D scene comprising a persistent object (the truck), along with several transient objects (e.g., people and a dog). While rays in blue from the three cameras intersect the truck, the green and orange rays from cameras 1 and 3 intersect transient objects. For video capture and spatio-temporal NeRF models, the persistent objects comprise the "static" portion of the scene, while the rest would be called the "dynamic".

3.1. Sensitivity to outliers

For Lambertian scenes, photo-consistent structure is view independent, as scene radiance only depends on the incident light [16]. For such scenes, view-dependent NeRF models like (1), trained by minimizing (3), admit local optima in which transient objects are explained by view-dependent terms. Figure 2 depicts this, with the outgoing color corresponding to the memorized color of the outlier – i.e. view-dependent radiance. Such models exploit the view-dependent capacity of the model to over-fit observations, effectively memorizing the transient objects. One can alter the model to remove dependence on $\mathbf{d}$, but the L2 loss remains problematic as least-squares (LS) estimators are sensitive to outliers, or heavy-tailed noise distributions.

Under more natural conditions, dropping the Lambertian assumption, the problem becomes more complex as *both* non-Lambertian reflectance phenomena and outliers can be explained as view-dependent radiance. While we want the models to capture photo-consistent view-dependent radiance, outliers and other transient phenomena should ideally be ignored. And in such cases, optimization with an L2 loss (3) yields significant errors in reconstruction; see Figure 1. Problems like these are pervasive in NeRF model fitting, especially in uncontrolled environments with complex reflectance, non-rigidity, or independently moving objects.

3.2. Robustness to outliers

Robustness via semantic segmentation. One way to reduce outlier contamination during NeRF model optimization is to rely on an *oracle* $\mathbf{S}$ that specifies whether a given pixel $\mathbf{r}$ from image i is an outlier, and should therefore be excluded from the empirical loss, replacing (3) with:

$$\mathcal{L}^{\mathbf{r},i}_{\text{oracle}}(\boldsymbol{\theta}) = \mathbf{S}_i(\mathbf{r}) \cdot ||\mathbf{C}(\mathbf{r};\boldsymbol{\theta}) - \mathbf{C}_i(\mathbf{r})||_2^2 \tag{4}$$

In practice, a *pre-trained* (semantic) segmentation network $\mathcal{S}$ might serve as an oracle, $\mathbf{S}_i{=}\mathcal{S}(\mathbf{C}_i)$. E.g., Nerf-in-the-wild [23] employed a semantic segmenter to remove pixels occupied by people, as they are outliers in the context of photo-tourism. Urban Radiance Fields [35] segmented out sky pixels, while LOL-NeRF [33] ignored pixels not belonging to faces. The obvious problem with this approach is the need for an oracle to detect arbitrary distractors.

Robust estimators. Another way to reduce sensitivity to outliers is to replace the conventional L2 loss (3) with a *robust loss* (e.g., [2, 41]), so that photometrically-inconsistent observations can be down-weighted during optimization. Given a robust kernel $\kappa(\cdot)$, we rewrite our training loss as:

$$\mathcal{L}^{\mathbf{r},i}_{\text{robust}}(\boldsymbol{\theta}) = \kappa(||\mathbf{C}(\mathbf{r};\boldsymbol{\theta}) - \mathbf{C}_i(\mathbf{r})||_2) \tag{5}$$

where $\kappa(\cdot)$ is positive and monotonically increasing. Mip-NeRF [3], for example, employs an L1 loss $\kappa(\epsilon){=}|\epsilon|$, which provides some degree of robustness to outliers during NeRF training. Given our analysis, a valid question is whether we can straightforwardly employ a robust kernel to approach our problem, and if so, given the large variety of robust kernels [2], which is the kernel of choice.

Unfortunately, as discussed above, outliers and non-Lambertian effects can *both* be modelled as view-dependent effects (see Figure 3). As a consequence, with simple application of robust estimators it can be difficult to separate signal from noise. Figure 4 shows examples in which outliers

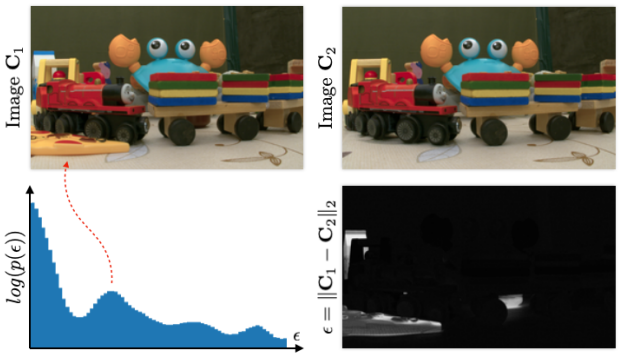

Figure 3. **Histograms** – Robust estimators perform well when the distribution of residuals agrees with the one implied by the estimator (e.g., Gaussian for L2, Laplacian for L1). Here we visualize the ground-truth distribution of residuals (bottom-left), which is hardly a good match with any simple parametric distribution.

are removed, but fine-grained texture and view-dependent details are also lost, or conversely, fine-grained details are preserved, but outliers cause artifacts in the reconstructed scene. One can also observe mixtures of these cases in which details are not captured well, nor are outliers fully removed. We find that this behaviour occurs consistently for many different robust estimators and parameter settings.

Training time can also be problematic. The robust estimator gradient w.r.t. model parameters can be expressed using the chain rule as

$$\left.\frac{\partial \kappa(\epsilon(\boldsymbol{\theta}))}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}^{(t)}} = \left.\frac{\partial \kappa(\epsilon)}{\partial \epsilon}\right|_{\epsilon(\boldsymbol{\theta}^{(t)})} \cdot \left.\frac{\partial \epsilon(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}^{(t)}} \tag{6}$$

The second factor is the classical NeRF gradient. The first factor is the kernel gradient evaluated at the *current* error residual $\epsilon(\boldsymbol{\theta}^{(t)})$. During training, large residuals can *equivalently* come from high-frequency details that have not yet been learnt, or they may arise from outliers (see Figure 4 (bottom)). This explain why robust optimization, implemented as (5), should not be expected to decouple high-frequency details from outliers. Further, when *strongly* robust kernels are employed, like redescending estimators, this also explains the loss of visual fidelity. That is, because the gradient of (large) residuals get down-weighted by the (small) gradients of the kernel, *slowing down* the learning of these fine-grained details (see Figure 4 (top)).

3.3. Robustness via Trimmed Least Squares

In what follows we advocate a form of iteratively reweighted least-squares (IRLS) with a Trimmed least squares (LS) loss for NeRF model fitting.

Iteratively Reweighted least Squares. IRLS is a widely used method for robust estimation that involves solving a sequence of weighted LS problems, the weights of which are adapted to reduce the influence of outliers. To that end,

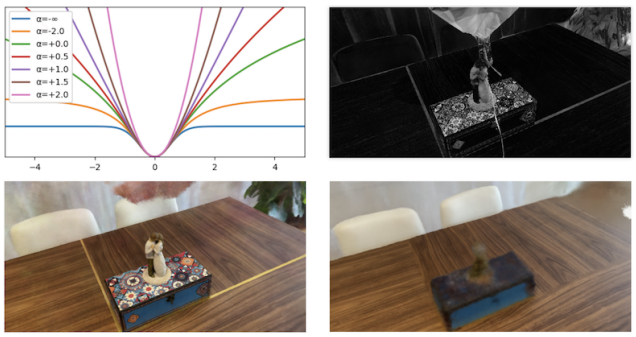

Figure 4. **Kernels –** (top-left) Family of robust kernels [2], including L2 (α=2), Charbonnier (α=1) and Geman-McClure (α=−2). (top-right) Mid-training, residual magnitudes are similar for distractors and fine-grained details, and pixels with large residuals are learned more slowly, as the gradient of re-descending kernels flattens out. (bottom-right) A too aggressive Geman-McClure in down-weighting large residuals removes both outliers and high-frequency detail. (bottom-left) A less aggressive Geman-McClure does not effectively remove outliers.

at iteration t, one can write the loss as

$$\begin{aligned}\mathcal{L}^{\mathbf{r},i}_{\text{robust}}(\boldsymbol{\theta}^{(t)}) &= \omega(\boldsymbol{\epsilon}^{(t-1)}(\mathbf{r})) \cdot ||\mathbf{C}(\mathbf{r};\boldsymbol{\theta}^{(t)}) - \mathbf{C}_i(\mathbf{r})||_2^2 \\ \boldsymbol{\epsilon}^{(t-1)}(\mathbf{r}) &= ||\mathbf{C}(\mathbf{r};\boldsymbol{\theta}^{(t-1)}) - \mathbf{C}_i(\mathbf{r})||_2 \end{aligned} \tag{7}$$

For weight functions given by $\omega(\epsilon)=\epsilon^{-1}\cdot\partial\kappa(\epsilon)/\partial\epsilon$ one can show that, under suitable conditions, the iteration converges to a local minima of (5) (see [41, Sec. 3]).

This framework admits a broad family of losses, including maximum likelihood estimators for heavy-tailed noise processes. Examples in Figure 4 include the Charbonnier loss (smoothed L1), and more aggressive redescending estimators such as the Lorentzian or Geman-McClure [2]. The objective in (4) can also be viewed as a weighted LS objective, the binary weights of which are provided by an oracle. And, as discussed at length below, one can also view several recent methods like NeRFW [23] and D^{2}NeRF [53] through the lens of IRLS and weighted LS.

Nevertheless, choosing a suitable weight function $\omega(\epsilon)$ for NeRF optimization is non-trivial, due in large part to the intrinsic ambiguity between view-dependent radiance phenomena and outliers. One might try to solve this problem by learning a neural weight function [40], although generating enough annotated training data might be prohibitive. Instead, the approach taken below is to exploit inductive biases in the structure of outliers, combined with the simplicity of a robust, trimmed LS estimator.

Trimmed Robust Kernels. Our goal is to develop a weight function for use in iteratively weighted LS optimization that is simple and captures useful inductive biases for NeRF optimization. For simplicity we opt for a binary weight function with intuitive parameters that adapts naturally through model fitting so that fine-grained image details that are not

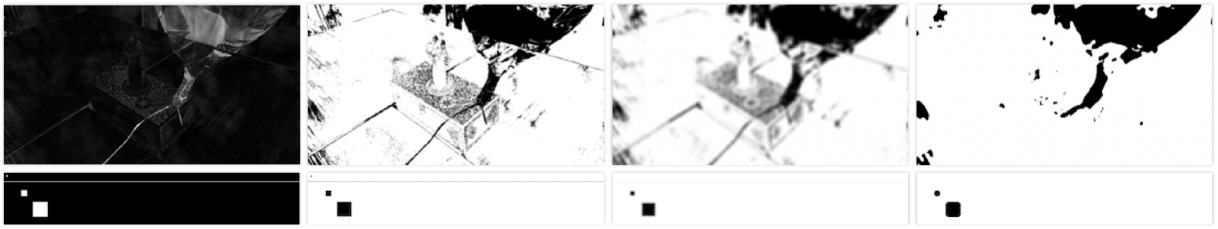

Figure 5. **Algorithm** – We visualize our weight function computed by residuals on two examples: (top) the residuals of a (mid-training) NeRF rendered from a *training* viewpoint, (bottom) a toy residual image containing residual of small spatial extent (dot, line) and residuals of large spatial extent (squares). Notice residuals with large magnitude but small spatial extent (texture of the box, dot, line) are included in the optimization, while weaker residuals with larger spatial extent are excluded. Note that while we operate on patches, we visualize the weight function on the whole image to facilitate visualization.

outliers can be learned quickly. It is also important to capture the structured nature of typical outliers, contrary to the typical i.i.d. assumption in most robust estimator formulations. To this end, the weight function should capture spatial smoothness of the outlier process, recognizing that objects typically have continuous local support, and hence outliers are expected to occupy large, connected regions of an image (e.g., the silhouette of a person to be segmented out from a photo-tourism dataset).

Surprisingly, a relatively simple weight function embodies these properties and performs extremely well in practice. The weight function is based on so-called *trimmed estimators* that are used in trimmed least-squares, like that used in trimmed ICP [7]. We first *sort* residuals, and assume that residuals below a certain percentile are inliers. Picking the 50% percentile for convenience (i.e., median), we define

$$\tilde{\omega}(\mathbf{r}) = \epsilon(\mathbf{r}) \leq \mathcal{T}_\epsilon \,, \quad \mathcal{T}_\epsilon = \text{Median}_\mathbf{r}\{\epsilon(\mathbf{r})\} \,. \tag{8}$$

To capture spatial smoothness of outliers we spatially diffuse inlier/outlier labels ω with a 3×3 box kernel $\mathcal{B}_{3\times3}$. Formally, we define

$$\mathcal{W}(\mathbf{r}) = (\tilde{\omega}(\mathbf{r}) \circledast \mathcal{B}_{3\times3}) \geq \mathcal{T}_\circledast \,, \quad \mathcal{T}_\circledast = 0.5 \,. \tag{9}$$

This helps to avoid classifying high-frequency details as outliers, allowing them to be captured by the NeRF model during optimization (see Figure 5).

While the trimmed weight function (9) improves the robustness of model fitting, it sometimes misclassifies fine-grained image details early in training where the NeRF model first captures coarse-grained structure. These localized texture elements may emerge but only after very long training times. We find that stronger inductive bias to spatially coherence allows fine-grained details to be learned more quickly. To that end, we aggregate the detection of outliers on 16×16 neighborhoods; i.e., we label entire 8×8 patches as outliers or inliers based on the behavior of $\mathcal{W}$ in the 16×16 neighborhood of the patch . Denoting the $N\times N$ neighborhood of pixels around $\mathbf{r}$ as $\mathcal{R}_N(\mathbf{r})$, we define

$$\omega(\mathcal{R}_8(\mathbf{r})) = \mathbb{E}_{\mathbf{s}\sim\mathcal{R}_{16}(\mathbf{r})}\left[\mathcal{W}(\mathbf{s})\right] \geq \mathcal{T}_\mathcal{R} \,, \quad \mathcal{T}_\mathcal{R} = 0.6 \,. \tag{10}$$

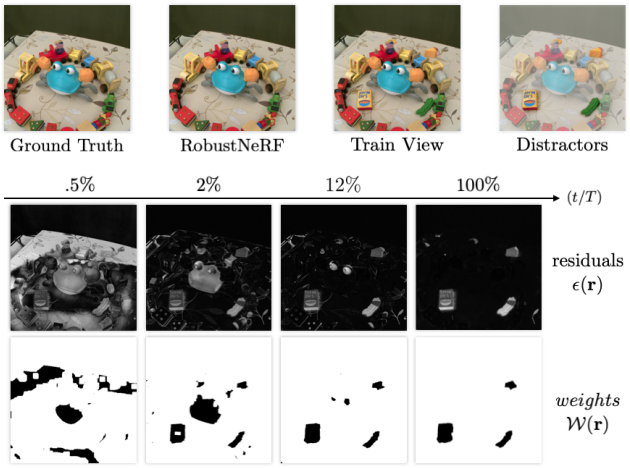

Figure 6. **Residuals –** For the dataset shown in the top row, we visualize the dynamics of the RobustNeRF training residuals, which show how over time the estimated distractor weights go from being random ((t/T)=0.5%) to identify distractor pixels ((t/T)=100%) without any explicit supervision.

The final weight function is the union of the three masks in Eqns. 8 -10. This robust weight function evolves during optimization, as one expects with IRLS where the weights are a function of the residuals at the previous iteration. That is, the labeling of pixels as inliers/outliers *changes* during training, and settles around masks similar to the one an oracle would provide as training converges (see Figure 6).

4. Experiments

We implement our robust loss function in the MultiNeRF codebase [26] and apply it to mip-NeRF 360 [4]. We dub this method "RobustNeRF". To evaluate RobustNeRF, we compare against baselines on several scenes containing different types of distractors. Where possible, we quantitatively compare reconstructions to held-out, distraction-free images; we report three metrics, averaged across held-out frames, namely, PSNR, SSIM [52], and LPIPS [60].

We compare different methods on two collections of scenes, i.e., those provided by the authors of D^{2}NeRF, and

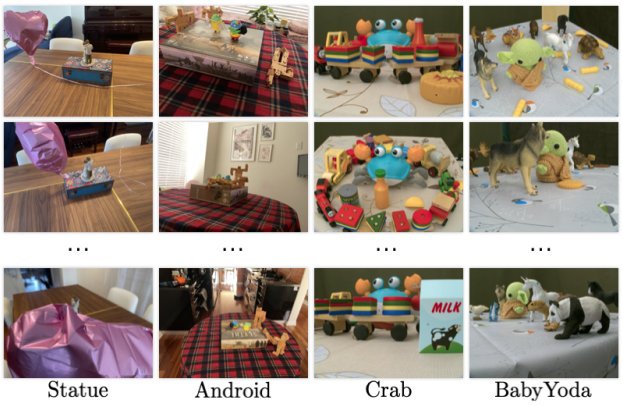

Figure 7. **Dataset –** Sample training images showing the distractors in each scene. Statue and Android were acquired manually, and the others with a robotic arm. In the robotic setting we have pixel-perfect alignment of distractor vs. distractor-free images.

novel datasets described below. We also present a series of illustrative experiments on synthetic scenes, shedding light on RobustNeRF's efficacy and inner workings.

4.1. Baselines

We compare RobustNeRF to variants of mip-NeRF 360 optimized with different loss functions (L_2, L_1, and Charbonnier). These variants serve as natural baselines for models with limited or no robustness to outliers.We also compare to D^{2}NeRF, a recent method for reconstructing dynamic scenes from monocular *video*. Unlike our method, D^{2}NeRF is designed to *reconstruct* distractors rather than discard them. While D^{2}NeRF is presented as a method for monocular video, it does not presuppose temporal continuity, and can be directly applied to unordered images. We omit additional comparisons to NeRF-W as its performance falls short of D^{2}NeRF [53]. For more details on model training, see the supplementary material.

4.2. Datasets – Figure 7

In addition to scenes from D^{2}NeRF, we introduce a set of natural and synthetic scenes. They facilitate the evaluation of RobustNeRF's effectiveness on illustrative use cases, and they enable empirical analysis under controlled conditions.

Natural scenes. We capture seven natural scenes exemplifying different types of distractors. Scenes are captured in three settings, on the street, in an apartment and in a robotics lab. Distractor objects are moved, or are allowed to move, between frames to simulate capture over extended periods of time. We vary the number of unique distractors from 1 (Statue) to 150 (BabyYoda), and their movements. Unlike prior work on monocular video, frames are captured without a clear temporal ordering (see Figure 7). The other three (i.e., Street1, Street2, and Gloss) include view-dependence effects, the results of which are shown in the supplementary material. We also capture additional frames *without distractors* to enable quantitative evaluations. Camera poses are estimated using COLMAP [38]. A full description of each scene in the supplementary material.

Synthetic scenes. To further evaluate RobustNeRF, we generate synthetic scenes using the Kubric dataset generator [12]. Each scene is constructed by placing a set of simple geometries in an empty, texture-less room. In each scene, a subset of objects remain fixed while the other objects (i.e., distractors) change position from frame to frame. By varying the number of objects, their size, and the way they move, we control the level of distraction in each scene. We use these scenes to examine RobustNeRF's sensitivity to its hyperparameters, see supplementary material.

4.3. Evaluation

We evaluate RobustNeRF on its ability to *ignore* distractors while accurately reconstructing the static elements of a scene. We train RobustNeRF, D^{2}NeRF, and variants of mip-NeRF 360 on scenes where distraction-free frames are available. Models are *trained* on frames with distractors and *evaluated* on distractor-free frames.

Comparison to mip-NeRF 360 – Figure 8. On natural scenes, RobustNeRF generally outperforms variants of mip-NeRF 360 by 1.3 to 4.7 dB in PSNR. As L_2, L_1, and Charbonnier losses weigh all pixels equally, the model is forced to represent, rather than ignore, distractors as "clouds" with view-dependent appearance. We find clouds to be most apparent when distractors remain stationary for multiple frames. In contrast, RobustNeRF's loss isolates distractor pixels and assigns them a weight of zero (see Figure 6). To establish an upper bound on reconstruction accuracy, we train mip-NeRF 360 with Charbonnier loss on distraction-free versions of each scene, the images for which are taken from (approximately) the same viewpoints. Reassuringly, RobustNeRF when trained on distraction-free frames, achieves nearly identical accuracy; see Figure 11.

While RobustNeRF consistently outperforms mip-NeRF 360, the gap is smaller in the Apartment scenes (Statue, Android) than the Robotics Lab scenes (Crab, BabyYoda). This can be explained by challenging background geometry, errors in camera parameter estimation, and imperceptible changes to scene appearance. For further discussion, see the supplementary material.

Comparison to D^{2}NeRF – Figure 9. Quantitatively, RobustNeRF matches or outperforms D^{2}NeRF by as much as 12 dB PSNR depending on the number of unique outlier objects in the capture. Results on D^{2}NeRF real scenes are provided in the supplementary material for qualitative comparison. In Statue and Android, 1 and 3 non-rigid objects are moved around the scene, respectively. D^{2}NeRF is able to model these objects and thus separate them from the scenes' static content. In the remaining scenes, a much

	Statue			Android			Crab			BabyYoda		
	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑
mip-NeRF 360 (L_2)	0.36	0.66	19.09	0.40	0.65	19.35	0.27	0.77	25.73	0.31	0.75	22.97
mip-NeRF 360 (L_1)	0.30	0.72	19.55	0.40	0.66	19.38	0.22	0.79	26.69	0.22	0.80	26.15
mip-NeRF 360 (Ch.)	0.30	0.73	19.64	0.40	0.66	19.53	0.21	0.80	27.72	0.23	0.80	25.22
D^{2}NeRF	0.48	0.49	19.09	0.43	0.57	20.61	0.42	0.68	21.18	0.44	0.65	17.32
RobustNeRF	**0.28**	**0.75**	**20.89**	**0.31**	**0.65**	**21.72**	**0.21**	**0.81**	**30.75**	**0.20**	**0.83**	**30.87**
mip-NeRF 360 (clean)	0.19	0.80	23.57	0.31	0.71	23.10	0.16	0.84	32.55	0.16	0.84	32.63

Figure 8. **Evaluation on Natural Scenes –** RobustNeRF outperforms baselines and D^{2}NeRF [53] on novel view synthesis with real-world captures. The table provides a quantitative comparison of RobustNeRF, D^{2}NeRF and mip-NeRF 360 using different reconstruction losses. The last row reports mip-NeRF 360 trained on a distractor-free version of each dataset, giving an upperbound for RobustNeRF performance. We also visualize samples from each scene rendered with each of the methods. See Supplementary Material for more samples.

larger pool of 100 to 150 unique, non-static objects are used – too many for D^{2}NeRF to model effectively. As a result, "cloud" artifacts appear in its static representation, similar to those produced by mip-NeRF 360. In contrast, RobustNeRF identifies non-static content as outliers and omits it during reconstruction. Although both methods use a similar number of parameters, D^{2}NeRF's peak memory usage is 2.3x higher than RobustNeRF and 37x higher when normalizing for batch size. This is a direct consequence of model architecture: D^{2}NeRF is tailored to simultaneously modeling static and dynamic content and thus merits higher complexity. To remain comparable, we limit image resolution to 0.2 megapixels for all experiments.

Ablations – Figure 10. We ablate elements of the RobustNeRF loss on the crab scene, comparing to an upper bound on the reconstruction accuracy of mip-NeRF 360 trained on distractor-free (clean) images from identical viewpoints. Our trimmed estimator (8) successfully eliminates distractors at the expense of high frequency texture and a lower PSNR. With smoothing (9), fine details are recovered, at the cost of longer training times. With the spatial window (10), RobustNeRF training time is on-par with mip-NeRF 360. We also ablate patch size and the trimming threshold (see Supplementary Material); we find that RobustNeRF is insensitive to trimming threshold, and that reducing the patch size offsets the gains from smoothing and patching.

	Car			Cars			**Bag**			Chairs			Pillow		
	LPIPS↓	MS-SSIM↑	PSNR↑	LPIPS↓	MS-SSIM↑	PSNR↑	LPIPS↓	MS-SSIM↑	PSNR↑	LPIPS↓	MS-SSIM↑	PSNR↑	LPIPS↓	MS-SSIM↑	PSNR↑
NeRF-W [23]	.218	.814	24.23	.243	.873	24.51	.139	.791	20.65	.150	.681	23.77	.088	.935	28.24
NSFF [18]	.200	.806	24.90	.620	.376	10.29	.108	.892	25.62	.682	.284	12.82	.782	.343	4.55
NeuralDiff [46]	.065	.952	31.89	.098	.921	25.93	.117	.910	29.02	.112	.722	24.42	.565	.652	20.09
D^{2}NeRF [53]	.062	.975	34.27	.090	.953	26.27	.076	.979	34.14	.095	.707	24.63	.076	.979	36.58
RobustNeRF	**.013**	**.988**	**37.73**	**.063**	**.957**	**26.31**	**.006**	**.995**	**41.82**	**.007**	**.992**	**41.23**	**.018**	**.990**	**38.95**

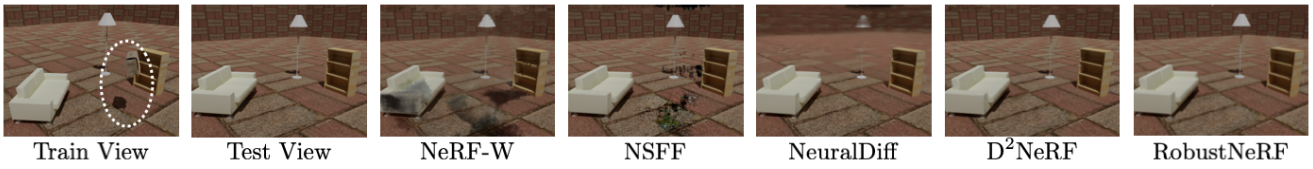

Figure 9. **Evaluations on D^{2}NeRF Synthetic Scenes –** Quantitative and qualitative evaluations on the Kubric synthetic dataset introduced by D^{2}NeRF, consisting of 200 training frames (with distractor) and 100 novel views for evaluation (without distractor).

	LPIPS↓	SSIM↑	PSNR↑	Updates to PSNR=30
mip-NeRF 360 (L_2)	0.31	0.75	22.97	–
+ robust (8)	0.39	0.60	18.21	–
+ smoothing (9)	0.22	0.81	30.01	250K
+ patching (10)	**0.21**	**0.81**	**30.75**	70K
oracle (clean)	0.16	0.84	32.55	25K

Figure 10. **Ablations –** Blindly trimming the loss causes details to be lost. Smoothing recovers fine-grained detail, while patch-based evaluation speeds up training and adds more detail. Patching enables the model to reach PSNR of 30, almost 4× faster.

Sensitivity – Figure 11. We find that RobustNeRF is remarkably robust to the amount of clutter in a dataset. We define an image as "cluttered" if it contains some number of distractor pixels. The figure shows how the reconstruction accuracy of RobustNeRF and mip-NeRF 360 depends on the fraction of training images with distractors, keeping the training set size constant. As the fraction increases, mip-NeRF 360's accuracy steadily drops from 33 to 25 dB, while RobustNeRF's remains steadily above 31 dB throughout. In the distraction-free regime, we find that RobustNeRF mildly under-performs mip-NeRF 360, both in reconstruction quality and the time needed for training. This follows from the statistical inefficiency induced by the trimmed estimator (8), for which a percentage of pixels will be discarded even if they do not correspond to distractors.

5. Conclusions

We address a central problem in training NeRF models, namely, optimization in the presence of distractors, such as transient or moving objects and photometric phenomena that are not persistent throughout the capture session.

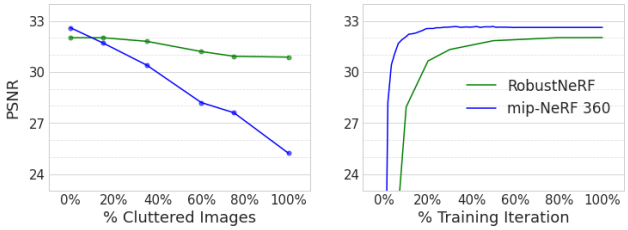

Figure 11. **Sensitivity and Limitations –** (left) Reconstruction accuracy for BabyYoda as we increase the fraction of train images with distractors. (right) Accuracy vs training time on *clean* BabyYoda images (distractor-free).

Viewed through the lens of robust estimation, we formulate training as a form of iteratively re-weighted least squares, with a variant of trimmed LS, and an inductive bias on the smoothness of the outlier process. RobustNeRF is surprisingly simple, yet effective on a wide range of datasets. RobustNeRF is shown to outperform recent state-of-the-art methods [4, 53], qualitatively and quantitatively, on a suite of synthetic datasets, common benchmark datasets, and new datasets captured by a robot, allowing fine-grained control over distractors for comparison with previous methods. While our experiments explore robust estimation in the context of mip-NeRF 360, the RobustNeRF loss can be incorporated within other NeRF models.

Limitations. While RobustNeRF performs well on scenes with distractors, the loss entails some statistical inefficiency. On clean data, this yields somewhat poorer reconstructions, often taking longer to train (see Figure 11). Future work will consider very small distractors, which may require adaptation of the spatial support used for outlier/inlier decisions. It would also be interesting to learn a neural weight function, further improving RobustNeRF; active learning may be useful in this context. Finally, it would be interesting to include our robust loss in other NeRF frameworks.

Acknowledgements We thank Pete Florence and Konstantinos Rematas for helpful feedback, and Tianhao Wu for help with D^{2}NeRF experiments.

References

[1] Michal Adamkiewicz, Timothy Chen, Adam Caccavale, Rachel Gardner, Preston Culbertson, Jeannette Bohg, and Mac Schwager. Vision-only robot navigation in a neural radiance world. *IEEE Robotics and Automation Letters*, 2022. 1, 2

[2] Jonathan T. Barron. A general and adaptive robust loss function. *Proc. CVPR*, 2019. 3, 4

[3] 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. 3

[4] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proc. CVPR*, 2022. 2, 5, 8

[5] Mark Boss, Andreas Engelhardt, Abhishek Kar, Yuanzhen Li, Deqing Sun, Jonathan T. Barron, Hendrik P.A. Lensch, and Varun Jampani. SAMURAI: Shape And Material from Unconstrained Real-world Arbitrary Image collections. In *Proc. NeurIPS*, 2022. 2

[6] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. *arXiv preprint arXiv:2208.00277*, 2022. 1, 2

[7] Dmitry Chetverikov, Dmitry Svirko, Dmitry Stepanov, and Pavel Krsek. The trimmed iterative closest point algorithm. In *International Conference on Pattern Recognition*, 2002. 5

[8] Shin-Fang Chng, Sameera Ramasinghe, Jamie Sherrah, and Simon Lucey. Garf: Gaussian activated radiance fields for high fidelity reconstruction and pose estimation. *arXiv e-prints*, 2022. 2

[9] Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B Tenenbaum, and Jiajun Wu. Neural radiance flow for 4d view synthesis and video processing. In *Proc. ICCV*. IEEE Computer Society, 2021. 2

[10] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. Fast dynamic radiance fields with time-aware neural voxels. *arXiv preprint arXiv:2205.15285*, 2022. 2

[11] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. In *Proc. ICCV*, 2021. 2

[12] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, Thomas Kipf, Abhijit Kundu, Dmitry Lagun, Issam Laradji, Hsueh-Ti (Derek) Liu, Henning Meyer, Yishu Miao, Derek Nowrouzezahrai, Cengiz Oztireli, Etienne Pot, Noha Radwan, Daniel Rebain, Sara Sabour, Mehdi S. M. Sajjadi, Matan Sela, Vincent Sitzmann, Austin Stone, Deqing Sun, Suhani Vora, Ziyu Wang, Tianhao Wu, Kwang Moo Yi, Fangcheng Zhong, and Andrea Tagliasacchi. Kubric: a scalable dataset generator. In *Proc. CVPR*, 2022. 6

[13] Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. *Proc. ICCV*, 2021. 2

[14] Yoonwoo Jeong, Seokjun Ahn, Christopher Choy, Anima Anandkumar, Minsu Cho, and Jaesik Park. Self-calibrating neural radiance fields. In *Proc. ICCV*, 2021. 2

[15] Animesh Karnewar, Tobias Ritschel, Oliver Wang, and Niloy Mitra. Relu fields: The little non-linearity that could. *TOG (Proc. SIGGRAPH)*, 2022. 2

[16] KN Kutulakos and SM Seitz. A theory of shape by space carving. *IJCV*, 2000. 3

[17] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, et al. Neural 3d video synthesis from multi-view video. In *Proc. CVPR*, 2022. 2

[18] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proc. CVPR*, 2021. 2, 8

[19] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. In *Proc. CVPR*, 2021. 2

[20] Jia-Wei Liu, Yan-Pei Cao, Weijia Mao, Wenqiao Zhang, David Junhao Zhang, Jussi Keppo, Ying Shan, Xiaohu Qie, and Mike Zheng Shou. Devrf: Fast deformable voxel radiance fields for dynamic scenes. *arXiv preprint arXiv:2205.15723*, 2022. 2

[21] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM TOG*, 2019. 1

[22] Li Ma, Xiaoyu Li, Jing Liao, Qi Zhang, Xuan Wang, Jue Wang, and Pedro V. Sander. Deblur-nerf: Neural radiance fields from blurry images. In *Proc. CVPR*, 2022. 2

[23] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proc. CVPR*, 2021. 2, 3, 4, 8

[24] Ben Mildenhall, Peter Hedman, Ricardo Martin-Brualla, Pratul P. Srinivasan, and Jonathan T. Barron. NeRF in the dark: High dynamic range view synthesis from noisy raw images. In *Proc. CVPR*, 2021. 2

[25] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *Proc. ECCV*, 2020. 1, 2

[26] Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, Peter Hedman, Ricardo Martin-Brualla, and Jonathan T. Barron. Multinerf: a code release for Mip-NeRF 360, Ref-NeRF, and RawNeRF, 2022. 5

[27] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *TOG (Proc. SIGGRAPH)*, 2022. 2

[28] Michael Niemeyer, Jonathan T. Barron, Ben Mildenhall, Mehdi S. M. Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *Proc. CVPR*, 2022. 2

[29] Michael Oechsle, Songyou Peng, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *Proc. ICCV*, 2021. 2

[30] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M Seitz. Hypernerf: a higher-dimensional representation for topologically varying neural radiance fields. *ACM Transactions on Graphics (TOG)*, 2021. 2

[31] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint 2209.14988*, 2022. 2

[32] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proc. CVPR*, 2021. 2

[33] Daniel Rebain, Mark Matthews, Kwang Moo Yi, Dmitry Lagun, and Andrea Tagliasacchi. LOLNeRF: Learn from One Look. In *Proc. CVPR*, 2022. 2, 3

[34] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *Proc. ICCV*, 2021. 1

[35] Konstantinos Rematas, Andrew Liu, Pratul P. Srinivasan, Jonathan T. Barron, Andrea Tagliasacchi, Tom Funkhouser, and Vittorio Ferrari. Urban radiance fields. *Proc. CVPR*, 2022. 1, 2, 3

[36] Antoni Rosinol, John J Leonard, and Luca Carlone. Nerf-slam: Real-time dense monocular slam with neural radiance fields. *arXiv preprint arXiv:2210.13641*, 2022. 1

[37] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proc. CVPR*, 2016. 1

[38] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *Proc. ECCV*, 2016. 6

[39] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Proc. NeurIPs*, 2019. 1

[40] Weiwei Sun, Wei Jiang, Andrea Tagliasacchi, Eduard Trulls, and Kwang Moo Yi. ACNe: Attentive Context Normalization for Robust Permutation-Equivariant Learning. *Proc. CVPR*, 2020. 4

[41] Andrea Tagliasacchi and Hao Li. Modern techniques and applications for real-time non-rigid registration. In *Proc. SIGGRAPH Asia (Technical Course Notes)*, 2016. 3, 4

[42] Andrea Tagliasacchi and Ben Mildenhall. Volume Rendering Digest (for NeRF), 2022. 2

[43] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. In *Proc. CVPR*, 2022. 2

[44] Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edgar Tretschk, W Yifan, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, et al. Advances in neural rendering. In *Computer Graphics Forum*, 2022. 1

[45] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *Proc. ICCV*, 2021. 2

[46] Vadim Tschernezki, Diane Larlus, and Andrea Vedaldi. NeuralDiff: Segmenting 3D objects that move in egocentric videos. In *Proc. 3DV*, 2021. 8

[47] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. Ref-NeRF: Structured view-dependent appearance for neural radiance fields. *Proc. CVPR*, 2022. 2

[48] Suhani Vora, Noha Radwan, Klaus Greff, Henning Meyer, Kyle Genova, Mehdi SM Sajjadi, Etienne Pot, Andrea Tagliasacchi, and Daniel Duckworth. Nesf: Neural semantic fields for generalizable semantic segmentation of 3d scenes. *TMLR*, 2021. 2

[49] Chaoyang Wang, Ben Eckart, Simon Lucey, and Orazio Gallo. Neural trajectory fields for dynamic novel view synthesis. *arXiv preprint arXiv:2105.05994*, 2021. 2

[50] Jiepeng Wang, Peng Wang, Xiaoxiao Long, Christian Theobalt, Taku Komura, Lingjie Liu, and Wenping Wang. Neuris: Neural reconstruction of indoor scenes using normal priors. *arXiv preprint*, 2022. 2

[51] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *Proc. NeurIPS*, 2021. 2

[52] Z. Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 2004. 5

[53] Tianhao Wu, Fangcheng Zhong, Forrester Cole, Andrea Tagliasacchi, and Cengiz Oztireli. D2nerf: Self-supervised decoupling of dynamic and static objects from a monocular video. In *Proc. NeurIPS*, 2022. 2, 4, 6, 7, 8

[54] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *Proc. CVPR*, 2021. 2

[55] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. *Comput. Graph. Forum*, 2022. 1

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

[57] Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction. In *Proc. NeurIPS*, 2022. 2

[58] Jason Y Zhang, Deva Ramanan, and Shubham Tulsiani. Relpose: Predicting probabilistic relative rotation for single objects in the wild. In *Proc. ECCV*, 2022. 2

[59] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 2

[60] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proc. CVPR*, 2018. 5

[61] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. In *Proc. CVPR*, 2022. 1