Bayes' Rays: Uncertainty Quantification for Neural Radiance Fields

Lily Goli[1] Cody Reading[2] Silvia Sellán[1] Alec Jacobson[1,4] Andrea Tagliasacchi[1,2,3]
University of Toronto[1] Simon Fraser University[2] Google DeepMind[3] Adobe Research[4]

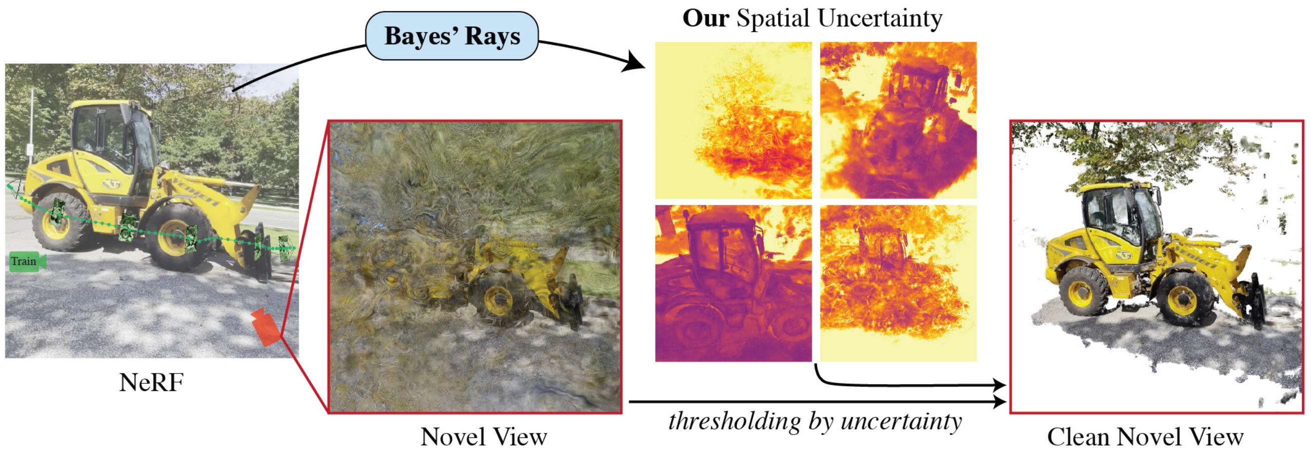


Figure 1. We introduce Bayes' Rays, a post-hoc algorithm to estimate the spatial uncertainty of any pre-trained NeRF of any arbitrary architecture. Our method requires no additional training and can be used to clean up NeRF artifacts caused by occluded or incomplete data.

Abstract

Neural Radiance Fields (NeRFs) have shown promise in applications like view synthesis and depth estimation, but learning from multiview images faces inherent uncertainties. Current methods to quantify them are either heuristic or computationally demanding. We introduce Bayes' Rays, a post-hoc framework to evaluate uncertainty in any pre-trained NeRF without modifying the training process. Our method establishes a volumetric uncertainty field using spatial perturbations and a Bayesian Laplace approximation. We derive our algorithm statistically and show its superior performance in key metrics and applications. More results available at: `https://bayesrays.github.io`

1. Introduction

Neural Radiance Fields (NeRFs) are a class of learned volumetric implicit scene representations that have exploded in popularity due to their success in applications like novel view synthesis and depth estimation. The process of learning a NeRF from a discrete set of multiview images is plagued with uncertainty: even in perfect experimental conditions, occlusions and missing views will limit the epistemic knowledge that the model can acquire about the scene.

Studying the epistemic uncertainty in NeRF is fundamental for tasks like outlier detection [23] and next-best-view planning [32] that expand NeRF's performance and application domain to critical areas like autonomous driving [12]. However, quantifying the uncertainty contained in a NeRF model is a relatively new area of study, with existing methods proposing either heuristic proxies without theoretical guarantees [16, 58] or probabilistic techniques that require costly computational power [47] and/or elaborate changes to the conventional NeRF training pipeline [1, 42].

We draw inspiration from triangulation problems in classic photogrammetry [45], where uncertainty is often modeled through distributions of feature point positions in image space that are then projected onto 3D (see Figure 2). Intuitively, this distribution measures how much a feature's position can be perturbed while maintaining multi-view consistency. We apply a similar intuition to NeRF, identifying

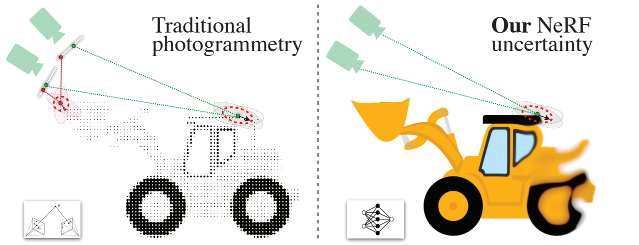


Figure 2. Inspired by uncertainty quantification in classic photogrammetry (left), we find epistemic uncertainty in NeRF (right).

the regions of the radiance field that can be spatially perturbed with minimal impact on the reconstruction loss.

We propose Bayes' Rays, a post-hoc framework for quantifying the uncertainty of any arbitrary pre-trained NeRF. Without requiring any changes to the training pipeline and regardless of the architecture in use in the given NeRF (see Figure 3), our method simulates spatially parametrized perturbations of the radiance field and uses a Bayesian Laplace approximation to produce a volumetric *uncertainty field*, which can be rendered like an additional color channel.

In this work, we show that our calculated uncertainties are not only statistically meaningful but also outperform previous works on key metrics like correlation to reconstructed depth error. Furthermore, they provide a framework for critical applications like removing *floater* artifacts from NeRF (see Figure 1), matching or improving the state-of-the-art. In summary, our main contributions are:

- We introduce a plug-and-play probabilistic method to quantify the uncertainty of any pre-trained Neural Radiance Field independently of its architecture, and without needing training images or costly extra training.
- In little over a minute, we compute a spatial uncertainty field that can be rendered as an additional color channel.
- We propose a direct application of our uncertainty field in removing artifacts from pre-trained NeRFs in real time.

2. Related Work

Uncertainty Quantification studies the distribution of the responses of a system conditioned on a set of measurable input variables [43]. As a field of statistics, it has grown over many decades out of the need to measure the accuracy of scientific predictions in areas like physics [15], chemistry [40] or meteorology [31].

Uncertainty in Computer Vision. Closer to our application, estimating the uncertainty of Computer Vision systems has been a subject of study even long before the Deep Learning revolution; for example, in Structure from Motion and Bundle Adjustment [45, Section 11.4][51]. In these problems from classic photogrammetry, scene geometry and camera parameters are jointly optimized in a process filled with uncertainty [46, 54], often modeled through 2D image-space Gaussians projected to 3D [9, 27, 45, 51] (see Figure 2).

Uncertainty in Deep Learning. The process of fitting a neural network to observed data typically contains two fundamentally different sources of uncertainty [2, 19]. *Aleatoric* uncertainty refers to the inherent randomness contained in the data (e.g., due to instrument error or uncontrolled influences) and is often captured by fitting not just one function but a parametric distribution (e.g., Gaussians) to the data [5, 19, 26]. On the other hand, *epistemic* uncertainty quantifies the lack of knowledge that a model has over the system it is trying to replicate; for example, due to missing data. Commonly, this is achieved through a Bayesian framework that estimates the posterior distribution of the model given observed data. The most straightforward-yet-expensive way of achieving this is via the use of *Deep Ensembles* that quantify the differences in optimized parameters after training many identical, yet differently initialized, networks on the same data [21, 57]. A popular alternative to ensembles are variational Bayesian Neural Networks, which model each network parameter with an independent distribution that is sampled at each training iteration and adjusted through a KL loss to approximate the true model posterior [10, 29]. This significant change to the training pipeline comes at a computational cost, which recent works have proposed circumventing through the use of post-hoc *Laplace Approximations* (see [13, 38], or [50] for a Computer Vision application) that only estimate the network weight posterior near their already-trained value using derivative information.

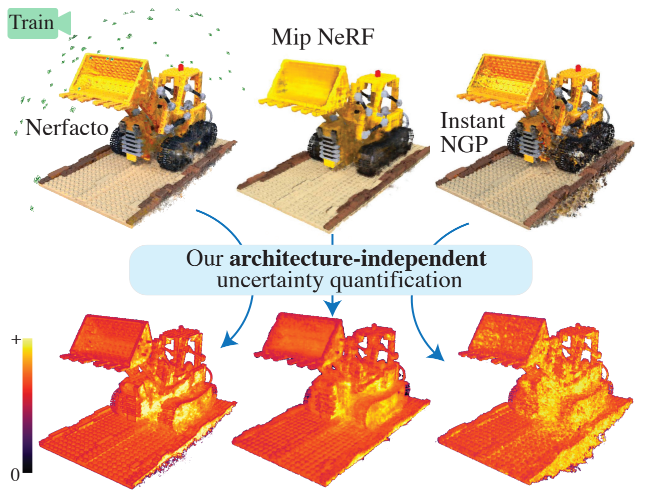


Figure 3. Bayes' Rays applied to different NeRF architectures. "Lego" [25] is only trained with cameras on its left hemisphere.

Uncertainty in Neural Radiance Fields. NeRFs [25] represent 3D scenes through a neural volumetric encoding that is optimized to match the images produced from multiple camera views. Aleatoric uncertainty presents itself in this process through the presence of transient objects in the scene or changes in lighting and camera specifications. These phenomena are quantified by the pioneering work NeRF-W [24] and subsequent follow-ups [18, 32, 36] through a combination of standard aleatoric Deep Learning techniques [2] and a learned appearance latent embedding [11].

Distinctly, we concern ourselves with the epistemic uncertainty of Neural Radiance Fields, the source of which is often missing data due to occlusions, ambiguities and limited camera views. Many of the general Deep Learning techniques to quantify this uncertainty have been applied to NeRF with limited success. Works like [47] propose uncertainty estimation through ensemble learning, which can be time and memory consuming. Shen et al. [42] and its follow-up [1] model the problem through variational inference and KL divergence optimization in a way that is similar in principle, yet shown to be superior, to standard variational Bayesian neural networks. All these methods require intricate changes to the NeRF training pipeline. In contrast, we introduce Bayes' Rays, the first framework that allows the use of Laplace approximations for NeRF uncertainty quantification, avoids variational optimization and can thus be applied on any pretrained NeRF of any arbitrary pipeline.

Away from the traditional Deep Learning uncertainty quantification frameworks, other works propose using NeRF-specific proxies for uncertainty. For example, Zhan et al. [58] propose computing the uncertainty as entropy of ray termination in NeRF model. While high entropy can be a good indicator of uncertainty in modeling solid objects, such assumption can fail while using density regularizers like the distortion loss proposed in [8]. Hoffman et al. [16] suggest quantifying uncertainty as the variance in scenes produced by a generative model when conditioned on partial observations, relying heavily on the specific model's priors.

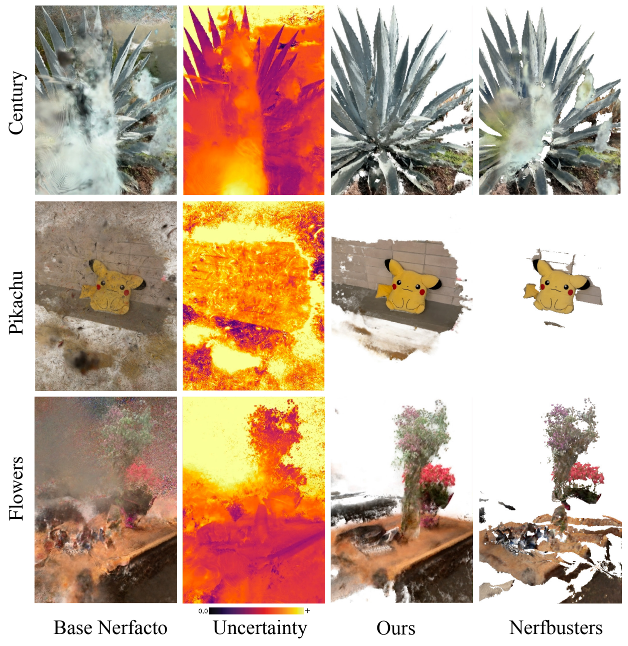
Other Related Works. As we show in Section 4, our uncertainty quantification relies on the sensitivity of any trained NeRF to perturbations, a concept recently explored in an unpublished preprint by Yan et al. [55] for applications outside NeRF and through a continual learning framework. To make our method computationally tractable and architecture-independent, we introduce a spatial deformation field similar to the one suggested by [33, 34], but interpret it as a re-parametrization of the NeRF model to quantify its uncertainty via Laplace approximation. One of the many uses we propose for our output spatial uncertainty field is to avoid common spatial NeRF artifacts. Instead of changing the optimization at training time by adding regularizers [8, 30, 35, 37], we propose removing them from any pre-trained NeRF in a post-processing step, a task that has been tackled recently by diffusion-based works like Nerfbusters [52]. As we show in Section 5, our algorithm matches or improves the performance of Nerfbusters in this specific application while being more general and requiring no additional training.

3. Background

We propose a general framework for applying Laplace approximations to quantify the epistemic uncertainty of any pre-trained NeRF. We will briefly review both these concepts before exploring the difficulties one encounters when trying to combine them naively, thus motivating our perturbation-based approach described in Section 4.

3.1. Neural Radiance Fields

Conventional NeRFs [25] learn to map each point in 3D space to a view-dependent radiance and a view-independent



Method	PSNR ↑	SSIM ↑	LPIPS ↓	Coverage ↑	Time ↓
Nerfacto (base)	16.83	0.52	0.39	0.89	-
Nerfbusters	17.99	**0.60**	**0.25**	0.63	20 min
Bayes′Rays-0.9	17.66	0.56	0.34	**0.83**	
Bayes′Rays-0.4	17.78	0.57	0.31	0.78	1.6 min
Bayes′Rays-best	**18.27**	**0.60**	0.27	0.70	

Figure 4. We propose cleaning up a learned scene by thresholding the computed uncertainty, matching or surpassing the state of the art at a much lower computational cost. Processing time computed on an NVIDIA RTX 6000; with no rendering overhead.

density value:

$$\mathbf{c}_{\boldsymbol{\phi}}(\mathbf{x}, \mathbf{d}),\ \tau_{\boldsymbol{\phi}}(\mathbf{x}) = \mathcal{R}(\mathbf{x}, \mathbf{d}; \boldsymbol{\phi}) \tag{1}$$

where $\boldsymbol{\phi}$ represent the learnable parameters in the neural field. The color of each pixel in an image can then be rendered through compositing the density and color of a series of points $\{t_i\}$ along the ray $\mathbf{r} = \mathbf{o_r} + t \cdot \mathbf{d_r}$, using volume rendering [48]:

$$\mathbf{C}_{\boldsymbol{\phi}}(\mathbf{r}) = \sum_i \exp\left(-\sum_{j<i} \tau_j \delta_j\right)(1 - \exp(-\tau_i \delta_i))\,\mathbf{c}_i, \tag{2}$$

where δ_i denotes the distance between each pair of successive points. The network parameters $\boldsymbol{\phi}$ are optimized to minimize reconstruction loss defined as the squared distance between the predicted color $\mathbf{C}(\mathbf{r})$ and ground truth $\mathbf{C}^{\text{gt}}$ for each ray $\mathbf{r}$ sampled from image $\mathbf{I}_n$ of training set images $\mathcal{I} = \{\mathbf{I}\}_{n=0}^{\mathbf{N}}$. From a Bayesian perspective, this is equivalent to assuming a Gaussian likelihood $p(\mathbf{C}_n^{\text{gt}}) \sim \mathcal{N}(\mathbf{C}_{\boldsymbol{\phi}}, s^2)$ and finding $\boldsymbol{\phi}^*$, the mode of the posterior distribution:

$$\boldsymbol{\phi}^* = \arg\max_{\boldsymbol{\phi}} p(\boldsymbol{\phi}|\mathcal{I}) \tag{3}$$

which, by Bayes' rule, is the same as minimizing the negative log-likelihood, with s^2 commonly set to $\frac{1}{2}$:

$$\boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\phi}} \; \mathbb{E}_i \, \mathbb{E}_{\mathbf{r}\sim\mathbf{I}_n} \|\mathbf{C}_{\boldsymbol{\phi}}(\mathbf{r}) - \mathbf{C}_n^{\text{gt}}(\mathbf{r})\|_2^2 \tag{4}$$

3.2. Neural Laplace Approximations

A common strategy to quantify the epistemic uncertainty of any neural network trained on some data $\mathcal{I}$ is to study the posterior distribution of the network parameters $\boldsymbol{\theta}$ conditioned on the data, $p(\boldsymbol{\theta}|\mathcal{I})$. In contrast to variational Bayesian neural networks, which propose using Bayes' rule and a variational optimization to estimate this distribution, Laplace approximations [13, 38] rely on simply training the network by any conventional means until convergence; i.e., on obtaining the likeliest network weights $\boldsymbol{\theta}^*$ – the mode of $p(\boldsymbol{\theta}|\mathcal{I})$. Then, the posterior is approximated by a multivariate Gaussian distribution centered at the obtained mode $p(\boldsymbol{\theta}|\mathcal{I}) \sim \mathcal{N}(\boldsymbol{\theta}^*, \boldsymbol{\Sigma})$. The covariance $\boldsymbol{\Sigma}$ of this distribution is then computed via a second-order Taylor expansion of the negative log-likelihood $h(\boldsymbol{\theta}) = -\log p(\boldsymbol{\theta}|\mathcal{I})$ about $\boldsymbol{\theta}^*$:

$$h(\boldsymbol{\theta}) \approx h(\boldsymbol{\theta}^*) + \frac{1}{2}(\boldsymbol{\theta}-\boldsymbol{\theta}^*)^\top \mathbf{H}(\boldsymbol{\theta}^*)\,(\boldsymbol{\theta}-\boldsymbol{\theta}^*)\,, \tag{5}$$

where first order terms are discarded since $\boldsymbol{\theta}^*$ is a maximum of $h(\boldsymbol{\theta})$ and $\mathbf{H}(\boldsymbol{\theta}^*)$ is the Hessian matrix of second derivatives of $h(\boldsymbol{\theta})$ evaluated at $\boldsymbol{\theta}^*$. Identifying the terms in (5) with the usual log squared exponential Gaussian likelihood of $\mathcal{N}(\boldsymbol{\theta}^*, \boldsymbol{\Sigma})$, one obtains:

$$\boldsymbol{\Sigma} = -\mathbf{H}(\boldsymbol{\theta}^*)^{-1} \tag{6}$$

Unfortunately, a naive application of this framework to NeRF by identifying $\boldsymbol{\theta}$ with $\boldsymbol{\phi}$ is impracticable, as it would have three potentially fatal flaws:

- First, as we show in Section 4.4, the parameters of the NeRF are strongly correlated with each other, making it difficult to accurately estimate the posterior distribution with any guarantees without computing, (and storing) all the entries in $\mathbf{H}$, a (potentially fully, at least block-) dense matrix with dimensions matching the number of network weights, before carrying out a costly inversion step.
- Secondly, even if one perfectly computed $\boldsymbol{\Sigma}$, the parameter correlations and network non-linearities would make it such that transferring this distribution to any geometrically meaningful one like a distribution over densities or pixel values would require repeatedly and expensively drawing samples from the full $\mathcal{N}(\boldsymbol{\phi}^*, \boldsymbol{\Sigma})$.
- Finally, beyond computational constraints, estimating uncertainty directly on the NeRF parameters would require our algorithm to have knowledge of (and, potentially, dependence on) the specific internal NeRF architecture used.

Below, we solve all of these problems by introducing a parametric *perturbation field* on which to perform the Laplace approximation. Our algorithm is completely independent of the specific NeRF architecture used and can guarantee minimal correlations between parameters, allowing us to calculate a meaningful spatial uncertainty field without the need to draw any distribution samples.

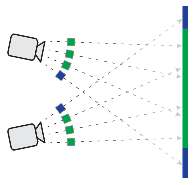
4. Method

As input, we take a pre-trained radiance field $\mathcal{R}$ with radiance function $\mathbf{c}$, density τ and optimized parameters $\boldsymbol{\phi}^*$, as well as ground truth camera parameters $\{\mathbf{T}_n\}$ corresponding to the $\mathbf{N}$ training images. Our method makes no assumption about the specific architecture of $\mathcal{R}$ and is designed for any arbitrary framework that produces a learned density $\tau_{\boldsymbol{\phi}^*}$ and radiance $\mathbf{c}_{\boldsymbol{\phi}^*}$, which we treat as differentiable black boxes.

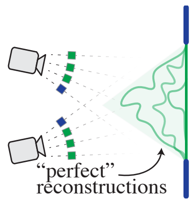
We begin by noting that, while the neural network weights $\boldsymbol{\phi}$ may serve as a useful parametrization of $\mathcal{R}$ during training, a Laplace approximation can be formulated on any reparametrization $\mathcal{R}_{\boldsymbol{\theta}}$ for any parameter set $\boldsymbol{\theta} \in \Theta$, as long as one knows the mode of the distribution $p(\boldsymbol{\theta}|\mathcal{I})$.

We follow by drawing inspiration from the key insight behind NeRFs: namely, that one can achieve impressive performance even in 2D tasks by explicitly modeling the 3D scene through a *volumetric* field. We also take inspiration from Computer Graphics, where global, volumetric *deformation fields* have been proposed as tools for manipulating implicitly represented objects [41, 44]. Finally, we draw inspiration from photogrammetry, where reconstruction uncertainty is often modeled by placing Gaussian distributions on the spatial positions of identified feature points (see Fig. 2).

4.1. Intuition



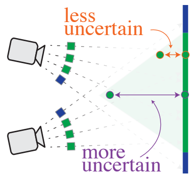
The intuition behind our reparametrization is better seen with the simple scene shown in the inset. Consider a single solid blue segment with a green center embedded in the 2D plane. Imagine that this object is observed by two simplified cameras that capture rays in a 60-degree cone, and let us now consider the NeRF reconstruction problem as stated on this small dataset.



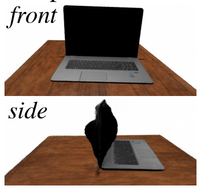
Trivially, the problem is an underdetermined one: as shown in the inset, the green segment could be substituted by many possible curves while still resulting in a "perfect" photometric reconstruction according to the four pixels in our dataset. Indeed, there is a whole *null-space* of solutions (green shaded region) to which the green segment could be perturbed without affecting the reconstruction loss[1], and a NeRF trained on this dataset may

[1]Note this is analogous to the condition number of near/far-field triangulation from photogrammetry that was discussed in Section 1.

converge to any one of these configurations depending on the training seed. Hence, one may quantify the uncertainty of a trained NeRF by asking "how much can one perturb it without hurting the reconstruction accuracy?"



Crucially, this quantity will vary spatially: some regions of space will be more constrained by the training set and will allow only a limited perturbation before having an adverse effect on the loss (e.g., the edges of the segment, orange in the inset) while others will be much more uncertain (e.g., the middle of the segment, purple in the inset). Hence, we will be able to quantify the *spatial* uncertainty of any trained NeRF by asking "which *regions* can one perturb without hurting the reconstruction accuracy?"



This quantity will be helpful beyond simple didactic examples: indeed, even general 3D scenes like the one in the inset can seem like pixel-perfect reconstructions from all training camera views (in this case, we trained a Nerfacto [49] model for 30,000 epochs with 40 front and back views of the laptop) but reveal large geometric artifacts when seen from a different angle.

4.2. Modeling Perturbations

Inspired by all the considerations above, we introduce a deformation field $\mathcal{D}: \mathbb{R}^D \to \mathbb{R}^D$, which one can interpret as a block that is ran on the input coordinates before the NeRF network. We choose a spatially meaningful parametrization in the form of vector displacements stored on the vertices of a grid of length M, allowing $\boldsymbol{\theta}$ to be represented as a matrix $\boldsymbol{\theta} \in \mathbb{R}^{M^D \times D}$, and defining a deformation for every spatial coordinate via trilinear interpolation:

$$\mathcal{D}_{\boldsymbol{\theta}(\mathbf{x})} = \text{Trilinear}(\mathbf{x}, \boldsymbol{\theta}) \,. \tag{7}$$

We can now reparametrize the radiance field $\mathcal{R}$ with θ by perturbing each coordinate $\mathbf{x}$ before applying the *already-optimized* NeRF neural network:

$$\tilde{\tau}_{\boldsymbol{\theta}}(\mathbf{x}) = \tau_{\boldsymbol{\phi}^*}(\mathbf{x} + \mathcal{D}_{\boldsymbol{\theta}}(\mathbf{x}))\,, \tag{8}$$

$$\tilde{\mathbf{c}}_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{c}_{\boldsymbol{\phi}^*}(\mathbf{x} + \mathcal{D}_{\boldsymbol{\theta}}(\mathbf{x}), \mathbf{d})\,, \tag{9}$$

resulting in the predicted pixel colors:

$$\tilde{\mathbf{C}}_{\boldsymbol{\theta}}(\mathbf{r}) = \sum_i \exp\left(-\sum_{j<i} \tilde{\tau}_j \delta_j\right) (1 - \exp(-\tilde{\tau}_i \delta_i))\, \tilde{\mathbf{c}}_i \,.$$

We proceed by assuming a likelihood of the same form as with the NeRF parametrization, $\mathbf{C}_n^{\text{gt}} \sim \mathcal{N}(\tilde{\mathbf{C}}_{\boldsymbol{\theta}}|\boldsymbol{\theta}, \frac{1}{2})$. Under our assumption that $\boldsymbol{\phi}^*$ are optimal parameters of the trained NeRF, it would be unreasonable to expect any non-trivial deformation to decrease the reconstruction loss; therefore we place a regularizing independent Gaussian prior $\boldsymbol{\theta} \sim \mathcal{N}(\mathbf{0}, \lambda^{-1})$ on our new parameters and formulating posterior $p(\boldsymbol{\theta}|I)$ whose negative log-likelihood $h(\boldsymbol{\theta})$ is given by

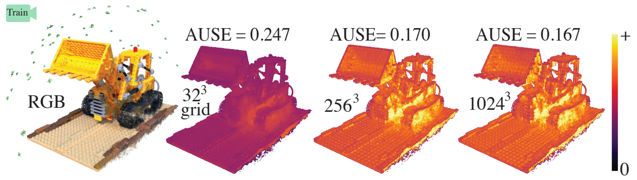


Figure 5. Very low resolutions may cause uncertainties to be underestimated, with diminishing returns for $M > 256$.

$$h(\boldsymbol{\theta}) = \mathbb{E}_n \mathbb{E}_{\mathbf{r}\sim\mathbf{I}_n} \|\tilde{\mathbf{C}}_{\boldsymbol{\theta}}(\mathbf{r}) - \mathbf{C}_n^{\text{gt}}(\mathbf{r})\|_2^2 + \lambda\|\boldsymbol{\theta}\|^2 \,. \tag{10}$$

The minimum of (10) must be obtained when $\boldsymbol{\theta} = 0$, as in that case $\tilde{\tau}_0(\mathbf{x}) = \tau_{\boldsymbol{\phi}^*}(\mathbf{x})$, $\tilde{\mathbf{c}}_0(\mathbf{x}) = \mathbf{c}_{\boldsymbol{\phi}^*}(\mathbf{x}, \mathbf{d})$ and thus $\tilde{\mathbf{C}}_0(\mathbf{r}) = \mathbf{C}_{\boldsymbol{\phi}}(\mathbf{r})$. Thus, zero is the mode of the distribution $p(\boldsymbol{\theta}|I)$ and we are finally in the ideal conditions for a Laplace approximation around $\boldsymbol{\theta}^* = 0$. Following Section 3.2, this result in a distribution $\boldsymbol{\theta} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ where

$$\boldsymbol{\Sigma} = -\mathbf{H}(\mathbf{0})^{-1} \tag{11}$$

where $\mathbf{H}$ is the Hessian matrix of second derivatives of $h(\boldsymbol{\theta})$ evaluated at zero. Computing these second derivatives is a computationally intensive task; however, as we show below, a combination of statistical and NeRF-specific tools allows us to approximate it in terms of first derivatives only.

4.3. Approximating H

For any parametric family of statistical distributions $p_{\boldsymbol{\theta}}$, the Hessian of the log-likelihood with respect to the parameters is also known as the *Fisher information*

$$\mathcal{I}(\boldsymbol{\theta}) = -\mathbb{E}_{\mathbf{X}\sim p_\theta}\left[\frac{\partial^2 h(\mathbf{X}\,;\,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}^2}\middle|\boldsymbol{\theta}\right] = -\mathbf{H}(\boldsymbol{\theta})\,, \tag{12}$$

which (under reasonable regularity assumptions) can also be defined as the variance of the parametric score [22, 5.3]

$$\mathcal{I}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{X}\sim p_\theta}\left[\frac{\partial h(\mathbf{X}\,;\,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}^\top \frac{\partial h(\mathbf{X}\,;\,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\middle|\boldsymbol{\theta}\right] \,. \tag{13}$$

Let us now denote the pair of random variables corresponding to a ray and its predicted color as $(\mathbf{r}, \mathbf{y})$, where $\mathbf{r} \sim \{\mathbf{I}_n\}$ and $\mathbf{y} = \mathbf{C}_n^{\text{gt}}(\mathbf{r})$. In our case, (13) takes the form

$$\mathcal{I}(\boldsymbol{\theta}) = \mathbb{E}_{(\mathbf{r},\mathbf{y})}\left[4\, \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{r})\, \mathbf{J}_{\boldsymbol{\theta}}(\mathbf{r})^\top \mathbf{J}_{\boldsymbol{\theta}}(\mathbf{r})\right] + 2\lambda\mathbf{I} \tag{14}$$

where $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{r})$ is the ray residual error

$$\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{r}) = \|\tilde{\mathbf{C}}_{\boldsymbol{\theta}}(\mathbf{r}) - \mathbf{C}_n^{\text{gt}}(\mathbf{r})\|^2$$

and $\mathbf{J}_{\boldsymbol{\theta}}(\mathbf{r})$ is the Jacobian of first derivatives

$$\mathbf{J}_{\boldsymbol{\theta}}(\mathbf{r}) = \frac{\partial \tilde{\mathbf{C}}_{\boldsymbol{\theta}}(\mathbf{r})}{\partial \boldsymbol{\theta}} \tag{15}$$

which can easily be computed via backpropagation.

Further, as we typically do not have multiple observations of ray color for a single ray $\mathbf{r}$, we can further simplify the above using the definition of conditional expectation

$$\mathcal{I}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{r}}\left[4\,\mathbb{E}_{\mathbf{y}|\mathbf{r}}\left[\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{r})\right]\mathbf{J}_{\boldsymbol{\theta}}(\mathbf{r})^\top \mathbf{J}_{\boldsymbol{\theta}}(\mathbf{r})\right] + 2\lambda\mathbf{I}\,, \tag{16}$$

noting that $\mathbb{E}_{\mathbf{y}|\mathbf{r}}\left[\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{r})\right]$ is nothing more than $\frac{1}{2}$, the variance of our stated likelihood $\mathbf{C}_n^{\text{gt}} \sim \mathcal{N}(\tilde{\mathbf{C}}_{\boldsymbol{\theta}}, \frac{1}{2})$,

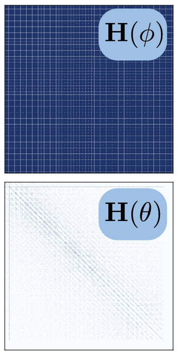
$$\mathcal{I}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{r}}\left[2\,\mathbf{J}_{\boldsymbol{\theta}}(\mathbf{r})^\top \mathbf{J}_{\boldsymbol{\theta}}(\mathbf{r})\right] + 2\lambda\mathbf{I}\,. \tag{17}$$

Combining (17) with (12) and approximating the expectation via sampling of R rays, we have our final expression for $\mathbf{H}$:

$$\mathbf{H}(\boldsymbol{\theta}) \approx -\frac{2}{R}\sum_{\mathbf{r}} \mathbf{J}_{\boldsymbol{\theta}}(\mathbf{r})^\top \mathbf{J}_{\boldsymbol{\theta}}(\mathbf{r}) - 2\lambda\mathbf{I}\,, \tag{18}$$

It is worth remarking that, while $\mathbf{H}$ contains in it all the information that we will need to quantify the epistemic uncertainty of the given radiance field, its computation in (18) *does not* explicitly rely on the data from the training images but *only* on information contained in the pre-trained model and the training camera parameters.

4.4. Spatial Uncertainty



We can now fully take advantage of our proposed reparametrization. First, since each vector entry in $\boldsymbol{\theta}$ corresponds to a vertex on our grid, its effect will be spatially limited to the cells containing it, making $\mathbf{H}(\boldsymbol{\theta})$ necessarily sparse and minimizing the number of correlated parameters (see inset, which compares the sparsity of $\mathbf{H}(\boldsymbol{\theta})$ to the that of an NeRF's MLP parameters $\boldsymbol{\phi}$). In fact, thanks to this low number of correlations, we will proceed like Ritter et al. [38] and approximate $\boldsymbol{\Sigma}$ only through the diagonal entries of $\mathbf{H}$:

$$\boldsymbol{\Sigma} \approx \operatorname{diag}\left(\frac{2}{R}\sum_{\mathbf{r}} \mathbf{J}_{\boldsymbol{\theta}}(\mathbf{r})^\top \mathbf{J}_{\boldsymbol{\theta}}(\mathbf{r}) + 2\lambda\mathbf{I}\right)^{-1}\,. \tag{19}$$

Secondly, by measuring the variance of our deformation field (intuitively, how much one could change the NeRF geometry without harming reconstruction quality), $\boldsymbol{\Sigma}$ critically encodes the *spatial uncertainty* of the radiance field. We can formalize this by considering the (root) diagonal entries of $\boldsymbol{\Sigma}$, which define a marginal variance vector $\boldsymbol{\sigma} = (\sigma_x, \sigma_y, \sigma_z)$. Much like in the photogrammetry works discussed in Section 2 and Figure 2, at each grid vertex, $\boldsymbol{\sigma}$ defines a spatial ellipsoid within which it can be deformed to minimal reconstruction cost. The norm of this vector $\sigma = \|\boldsymbol{\sigma}\|_2$ is then a positive scalar that measures the local spatial uncertainty of the radiance field at each grid vertex. Through it, we can define our *spatial uncertainty field* $\mathcal{U} : \mathbb{R}^3 \to \mathbb{R}^+$ given by

$$\mathcal{U}(\mathbf{x}) = \text{Trilinear}(\mathbf{x}, \sigma)\,, \tag{20}$$

which intuitively encodes how much the positioning of geometric region in our reconstruction can be trusted. Strictly speaking, as defined above, $\mathcal{U}$ measures the uncertainty at $(1 + \mathcal{D}_{\boldsymbol{\theta}})^{-1}(\mathbf{x})$, not $\mathbf{x}$; however, we note that these are equivalent for the trained NeRF for which $\mathcal{D}_{\boldsymbol{\theta}^*} = 0$.

The uncertainty field $\mathcal{U}$ is the main output of our algorithm and serves to illustrate the success of our approach. It is a first-of-its-kind theoretically derivated spatial measure of uncertainty that can be computed on *any* NeRF architecture, without the need for additional training, expensive sampling or even access to the training images.

5. Experiments & Applications

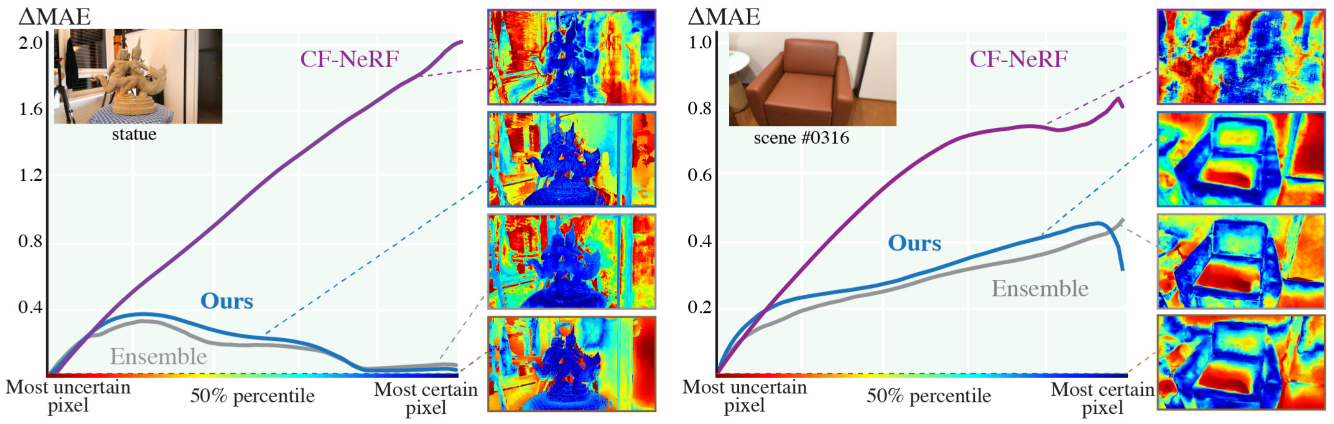
We validate our theoretically-derived algorithm through our uncertainty field's correlation with the depth prediction error (Section 5.1), show a prototypical application to a NeRF clean-up task (Section 5.2) and justify our parametric choices through ablation studies (Section 5.3).

Implementation. Unless specified otherwise, all NeRFs used throughout this paper use Nerfstudio's Nerfacto [49] as the base architecture and are pre-trained for 30,000 steps. We extract our uncertainty field $\mathcal{U}$ using 1,000 random batches of 4,096 rays each sampled from a scene's training cameras, with $M = 256$ and $\lambda = 10^{-4}/M^3$, in a process that takes around 90 seconds on an NVIDIA RTX 6000 and takes 64MB of additional storage (25% original model size). Once computed for a given scene, our uncertainty field conveniently functions as an additional color channel that can be volumetrically rendered similar to the usual RGB. No overhead is incurred on rendering speed, with uncertainty computation per point being a simple look-up in the single-resolution uncertainty grid. For visualization clarity, all our results use a logarithmic scale, rendering $\log \mathcal{U}$ instead of $\mathcal{U}$.

5.1. Uncertainty Evaluation – Figures 6

We evaluate the estimated uncertainty of Bayes' Rays by showing its correlation with the NeRF depth error. We choose predicted depth error as the best signal conveying NeRF's unreliability in *geometric* prediction, as RGB error has been shown to not be representative of this uncertainty due to radiance accumulation and model biases [47].

Metric. We measure correlation through the Area Under Sparsification Error (AUSE) [6, 17]. The pixels in each test image are removed gradually ("sparsified") twice: first, according to their respective depth error; second, by their uncertainty measure. The difference between the Mean Absolute depth Error (ΔMAE) of the remaining pixels in both processes provides the sparsification curves.



Method \ Scene	africa	basket	statue	torch	#0000	#0079	#0158	#0316
Ensemble	0.18	0.24	0.15	0.19	0.28	0.36	0.19	0.26
CF-NeRF	0.35	0.31	0.46	0.97	0.59	0.43	0.55	0.54
Ours	**0.27**	**0.28**	**0.17**	**0.22**	**0.28**	**0.35**	**0.20**	**0.29**

Figure 6. The uncertainties computed with our algorithm on the ScanNet and Light Field dataset are significantly more calibrated to the real NeRF depth error than the previous state-of-the-art CF-NeRF [1], even matching the performance of extremely costly ensembles. Images are colored by uncertainty / depth error *percentile* instead of value to be comparable. Please refer to Appendix B for more results.

Data. In Figure 6, we use 4 ScanNet scenes (#0000-001, #0079-000, #0158-000, #0316-000) with ground-truth depths provided. Each scene employs 40 images split into 5 test and 35 train images, following NerfingMVS [53]. Additionally, we use 4 scenes from the Light Field dataset [56, 59] (torch, statue, basket, africa), with the same train-test split and pseudo-ground-truth depth map approach as CF-NeRF [1]. To solve the scale ambiguity between the trained NeRF scenes and ground-truth scenes, we use the same rescaling as in [14, 53] for ground-truth depth.

Baselines. For Figure 6, we display sparsification curves derived from our uncertainty field, with the previous state-of-the-art CF-NeRF [1] and with the standard deviations obtained by a costly process of training an ensemble of 10 identical yet differently seeded Nerfacto models. We forgo density-aware ensemble [47] since Nerfacto does not have the black zero-density space failure, evident in Figures 1, 4, where colored and high-density floaters appear outside the training hull. Next to each graph, we visualize the depth error together with the (ascending) per-pixels rank produced by each method (i.e., the ordering that produces the curves). Unlike CF-NeRF [1], we do not measure disparity error due to its heightened sensitivity to low-range depth errors and significant underestimation of errors in distant points.

Results. The results are consistent across Figure 6. Bayes' Rays's uncertainty shows significant improvement in correlation with depth error compared to CF-NeRF [1], both quantitatively and qualitatively. Further, our uncertainty is extremely close to the standard deviation of a costly ensemble in both AUSE and sparsification plots, while requiring no additional trained NeRFs, hence 10x less time and memory.

5.2. NeRF Clean Up – Figures 1 and 4

A common reconstruction artifact in NeRFs are "floaters", often caused by a lack of information in training data. These inherently correspond to regions of high uncertainty; therefore, we propose removing them by thresholding the scene according to our uncertainty field $\mathcal{U}$ during rendering.

In Figure 4, we compare our algorithm's performance to the current state of the art for post-hoc floater removal, Nerfbusters [52], which employs a 3D diffusion model and a "visibility mask" to guide additional training steps during which some floaters are removed. For our comparison, we use the same dataset proposed by Nerfbusters along with their proposed metric of *Coverage*, together with more common measures of image quality. An ideal clean-up would boost image quality while keeping pixel coverage high.

When using fixed threshold values like 0.9 or 0.4, Bayes' Rays obtains similar PSNR values to Nerfbusters while allowing for a higher coverage. If one selects the best possible threshold value for each scene out of ten equally spaced ones, Bayes' Rays outperforms Nerfbusters in both PSNR and coverage. It is worth noting that Bayes' Rays achieves with a significantly lower computational footprint: unlike Nerfbusters, we do not require storing and evaluating a 3D diffusion model, we are faster (96 seconds vs 20 minutes) by eliminating the need for additional training and we circumvent the use of a "visibility mask" altogether by storing all necessary information in our computed Hessian $\mathbf{H}$.

Our qualitative results in Figure 4 show that our method can filter floaters that are missed by Nerfbusters (*Century*),

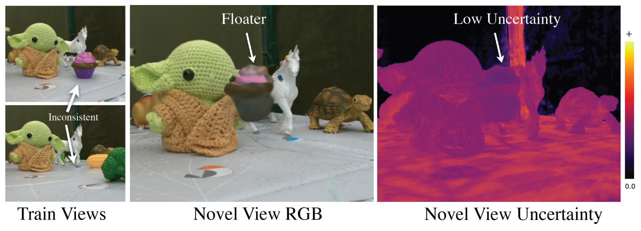


Figure 7. Bayes' Rays quantifies only epistemic uncertainty in NeRF and is thus unable to capture aleatoric effects like those stemming from training inconsistencies.

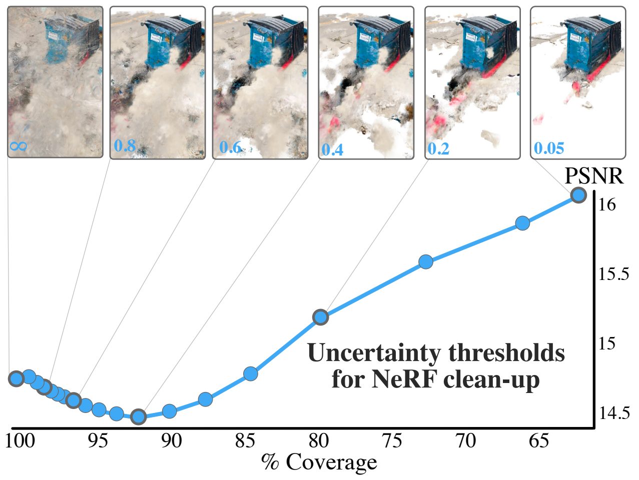


Figure 8. Applying different thresholds to uncertainty for the NeRF clean up task on the "garbage" scene from Nerfbusters dataset [52]. Left-most image (threshold=∞) shows the original NeRF result.

is not prone to sampling artifacts caused by floater removal (*Flowers*) and provides the parametric flexibility necessary to avoid over-filtering (*Pikachu*).

5.3. Algorithmic Ablations – Figures 3, 5 and 8

In Section 4, we justified our introduction of the perturbation field $\mathcal{D}$ partly through the desire to make our algorithm independent to the specific NeRF architecture used. In Figure 3, we show that this is indeed the case, as we obtain qualitatively similar results for three representatively different architectures (Mip-NeRF [7], Instant-NGP [28] and Nerfacto [49]) on the "Lego" scene from the NeRF Synthetic dataset [25] with 60 training views from its left hemisphere.

This success introduces an algorithmic choice; namely, the discretization of the deformation field. In Section 4, we propose storing it in a uniform spatial grid of size M^3 from which deformation values can be trilinearly interpolated. The value of M thus becomes an algorithmic parameter, which we explore for the same example in Figure 5. We find that surface uncertainty can be missed for small values of M, resulting in a generally more certain map that is only activated on points of very high uncertainty, with diminishing returns being obtained for larger, more costly $M > 256$.

Finally, our algorithm's flagship application to NeRF artifact removal also contains a parameter choice, in the form of the uncertainty threshold. As we show in Figure 8, decreasing this parameter can gradually clean a floater-heavy scene leaving a floater-free clean capture of the target object. Since our uncertainty field only needs to be computed once, we suggest that this threshold can serve as *real-time* user control in interactive NeRF setups like Nerfstudio [49].

6. Conclusions

We have introduced Bayes' Rays, an algorithm to quantify the uncertainty of any trained Neural Radiance Field independently of its architecture and without additional training nor access to the original training images. Our algorithm outputs a spatial uncertainty field, which we have shown is meaningfully correlated to the NeRF depth error and can be thresholded for use in applications like NeRF cleanup.

We discretize our spatial deformation field using a uniform grid, which can lead to a high memory cost being incurred in regions of little geometric interest. Future work may significantly improve our performance by considering more complex hierarchical data structures like octrees. Separately, in our algorithm's derivation, we focus only on the diagonal of $\mathbf{H}$, disregarding (minimal) inter-parametric correlations. Future applications may require their inclusion, possibly through low-rank matrix decompositions [3, 4].

At a higher level, our algorithm's stated goal is to capture only the epistemic uncertainty of the NeRF, often present through missing or occluded data. As such, aleatoric uncertainty caused by noise or inconsistencies between views is not captured by our method (see Figure 7). We are optimistic that combining our work with current frameworks for aleatoric quantification like [24, 39] will result in a complete study of all sources of uncertainty in NeRF.

More broadly, our algorithm is limited to quantifying the uncertainty of NeRFs and may not directly apply to other frameworks. However, we anticipate similar Laplace approximations for newer representations like 3D Gaussian splatting [20]. As Deep Learning continues to advance Computer Vision algorithms, we hope that our work can contribute to the understanding of models' knowledge and limitations.

Acknowledgements. We thank Alireza Mousavi-Hosseini, Agustinus Kristiadi, Towaki Takikawa, Kevin Swersky, Richard Szeliski, Aaron Hertzmann, Georgios Kopanas, Vladimir Kim, and Nathan Carr for discussions. AT acknowledges support from NSERC Discovery [2016-05786] and SFU visual computing research chair. AJ acknowledges support from NSERC Discovery [2022–04680], Ontario Early Research Award, Canada Research Chairs Program, Sloan Research Fellowship, DSI Catalyst Grant program, and gifts by Adobe Inc. SS acknowledges support from NSERC Vanier Scholarship and Adobe Research Fellowship. We also thank the Vector Institute for compute infrastructure.

References

[1] Conditional-flow nerf: Accurate 3d modelling with reliable uncertainty quantification. *ECCV*, 2022. 1, 3, 7

[2] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Abbas Khosravi, U Rajendra Acharya, Vladimir Makarenkov, and Saeid Nahavandi. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *arXive preprint at arXiv:2011.06225*, 2021. 2

[3] Sivaram Ambikasaran and Eric Darve. An o(n log n) fast direct solver for partial hierarchically semi-separable matrices: With application to radial basis function interpolation. *Journal of Scientific Computing*, 2013. 8

[4] Sivaram Ambikasaran, Michael O'Neil, and Karan Raj Singh. Fast symmetric factorization of hierarchical matrices with applications. *arXiv preprint arXiv:1405.0223*, 2014. 8

[5] Murat Seckin Ayhan and Philipp Berens. Test-time data augmentation for estimation of heteroscedastic aleatoric uncertainty in deep neural networks. *Medical Imaging with Deep Learning*, 2022. 2

[6] Gwangbin Bae, Ignas Budvytis, and Roberto Cipolla. Estimating and exploiting the aleatoric uncertainty in surface normal estimation. *ICCV*, 2021. 6

[7] 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. 8

[8] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CVPR*, 2021. 3

[9] Adrien Bartoli. Towards gauge invariant bundle adjustment: A solution based on gauge dependent damping. *ICCV*, 2003. 2

[10] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *ICML*, 2015. 2

[11] Piotr Bojanowski, Armand Joulin, David Lopez-Paz, and Arthur Szlam. Optimizing the latent space of generative networks. *ICML*, 2019. 2

[12] Kashyap Chitta, Aditya Prakash, and Andreas Geiger. Neat: Neural attention fields for end-to-end autonomous driving. *ICCV*, 2021. 1

[13] Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace redux – effortless bayesian deep learning. *NeuRIPS*, 2021. 2, 4

[14] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. *arXiv preprint arXiv:2107.02791*, 2021. 7

[15] John Ellis, Mary K Gaillard, Dimitri V Nanopoulos, and Serge Rudaz. Uncertainties in the proton lifetime. *Nuclear Physics B*, 1980. 2

[16] Matthew D. Hoffman, Tuan Anh Le, Pavel Sountsov, Christopher Suter, Ben Lee, Vikash K. Mansinghka, and Rif A. Saurous. Probnerf: Uncertainty-aware inference of 3d shapes from 2d images. *AISTATS*, 2022. 1, 3

[17] Eddy Ilg, Özgün Çiçek, Silvio Galesso, Aaron Klein, Osama Makansi, Frank Hutter, and Thomas Brox. Uncertainty estimates and multi-hypotheses networks for optical flow. *ECCV*, 2018. 6

[18] Liren Jin, Xieyuanli Chen, Julius Rückin, and Marija Popovic. Neu-nbv: Next best view planning using uncertainty estimation in image-based neural rendering. *IROS*, 2023. 2

[19] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *NeuRIPS*, 2017. 2

[20] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 2023. 8

[21] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *NeuRIPS*, 2016. 2

[22] Erich L. Lehmann and George Casella. *Theory of Point Estimation*. Springer-Verlag, second edition, 1998. 5

[23] Sergio Naval Marimont and Giacomo Tarroni. Implicit field learning for unsupervised anomaly detection in medical images. *MICCAI*, 2021. 1

[24] Ricardo Martin-Brualla, Noha Radwan, Mehdi Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the Wild: Neural radiance fields for unconstrained photo collections. *CVPR*, 2020. 2, 8

[25] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *ECCV*, 2020. 2, 3, 8

[26] Miguel Monteiro, Loïc Le Folgoc, Daniel Coelho de Castro, Nick Pawlowski, Bernardo Marques, Konstantinos Kamnitsas, Mark van der Wilk, and Ben Glocker. Stochastic segmentation networks: Modelling spatially correlated aleatoric uncertainty. *Advances in neural information processing systems*, 2020. 2

[27] Daniel Morris, Kenichi Kanatani, and Takeo Kanade. Uncertainty modeling for optimal structure from motion. *Workshop on Vision Algorithms*, 1999. 2

[28] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *SIGGRAPH*, 2022. 8

[29] Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag, 1995. 2

[30] 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. *CVPR*, 2021. 3

[31] Tim N Palmer. Predicting uncertainty in forecasts of weather and climate. *Reports on progress in Physics*, 2000. 2

[32] Xuran Pan, Zihang Lai, Shiji Song, and Gao Huang. Activenerf: Learning where to see with uncertainty estimation. *ECCV*, 2022. 1, 2

[33] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B. Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Deformable neural radiance fields. *ICCV*, 2020. 3

[34] 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 Trans. Graph.*, 2021. 3

[35] Keunhong Park, Philipp Henzler, Ben Mildenhall, Jonathan T. Barron, and Ricardo Martin-Brualla. Camp: Camera preconditioning for neural radiance fields, 2023. 3

[36] Yunlong Ran, Jing Zeng, Shibo He, Jiming Chen, Lincheng Li, Yingfeng Chen, Gimhee Lee, and Qi Ye. NeurAR: Neural uncertainty for autonomous 3d reconstruction with implicit neural representations. *IEEE Robotics and Automation Letters*, 2023. 2

[37] Daniel Rebain, Mark Matthews, Kwang Moo Yi, Dmitry Lagun, and Andrea Tagliasacchi. Lolnerf: Learn from one look. *CVPR*, 2022. 3

[38] Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable laplace approximation for neural networks. *ICLR*, 2018. 2, 4, 6

[39] Sara Sabour, Suhani Vora, Daniel Duckworth, Ivan Krasin, David J. Fleet, and Andrea Tagliasacchi. Robustnerf: Ignoring distractors with robust losses. *CVPR*, 2023. 8

[40] William D Schecher and Charles T Driscoll. An evaluation of the equilibrium calculations within acidification models: the effect of uncertainty in measured chemical components. *Water Resources Research*, 1988. 2

[41] Dario Seyb, Alec Jacobson, Derek Nowrouzezahrai, and Wojciech Jarosz. Non-linear sphere tracing for rendering deformed signed distance fields. *ACM Trans. Graph.*, 2019. 4

[42] Jianxiong Shen, Adria Ruiz, Antonio Agudo, and Francesc Moreno-Noguer. Stochastic neural radiance fields: Quantifying uncertainty in implicit 3d representations. *3DV*, 2021. 1, 2

[43] Ralph C Smith. *Uncertainty quantification: theory, implementation, and applications*. Siam, 2013. 2

[44] Masamichi Sugihara, Brian Wyvill, and Ryan Schmidt. Warpcurves: A tool for explicit manipulation of implicit surfaces. *Computers & Graphics*, 2010. 4

[45] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Nature, 2022. 1, 2

[46] Richard Szeliski and Sing Bing Kang. Shape ambiguities in structure from motion. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 1997. 2

[47] Niko Sünderhauf, Jad Abou-Chakra, and Dimity Miller. Density-aware nerf ensembles: Quantifying predictive uncertainty in neural radiance fields. *ICRA*, 2023. 1, 2, 6, 7

[48] Andrea Tagliasacchi and Ben Mildenhall. Volume rendering digest (for nerf). *arXiv preprint arXiv:2209.02417*, 2022. 3

[49] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David Mcallister, Justin Kerr, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Proceedings*, 2023. 5, 6, 8

[50] Anastasia Tkach, Andrea Tagliasacchi, Edoardo Remelli, Mark Pauly, and Andrew Fitzgibbon. Online generative model personalization for hand tracking. *ACM Trans. Graph.*, 2017. 2

[51] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew William Fitzgibbon. Bundle adjustment - a modern synthesis. *Workshop on Vision Algorithms*, 1999. 2

[52] Frederik Warburg, Ethan Weber, Matthew Tancik, Aleksander Holynski, and Angjoo Kanazawa. Nerfbusters: Removing ghostly artifacts from casually captured nerfs. *ICCV*, 2023. 3, 7, 8

[53] Yi Wei, Shaohui Liu, Yongming Rao, Wang Zhao, Jiwen Lu, and Jie Zhou. Nerfingmvs: Guided optimization of neural radiance fields for indoor multi-view stereo. *ICCV*, 2021. 7

[54] Kyle Wilson and Scott Wehrwein. Visualizing spectral bundle adjustment uncertainty. *2020 International Conference on 3D Vision (3DV)*, 2020. 2

[55] Zike Yan, Haoxiang Yang, and Hongbin Zha. Active neural mapping. *ICCV*, 2023. 3

[56] Kaan Yücer, Alexander Sorkine-Hornung, and Oliver Disney. Efficient 3 d object segmentation from densely sampled light fields with applications to 3 d reconstruction. *ACM TOG*, 2016. 7

[57] Sheheryar Zaidi, Arber Zela, Thomas Elsken, Christopher C. Holmes, Frank Hutter, and Yee Whye Teh. Neural ensemble search for uncertainty estimation and dataset shift. *NeuRIPS*, 2021. 2

[58] Huangying Zhan, Jiyang Zheng, Yi Xu, Ian Reid, and Hamid Rezatofighi. Activermap: Radiance field for active mapping and planning. *arXiv preprint arXiv:2211.12656*, 2022. 1, 3

[59] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *preprint arXiv:2010.07492*, 2020. 7