ProvNeRF: Modeling per Point Provenance in NeRFs as a Stochastic Process

George Kiyohiro Nakayama[1] Mikaela Angelina Uy[1,3] Yang You[1]
Ke Li[2] Leonidas Guibas[1]
[1]Stanford University [2]Simon Fraser University [3]Google

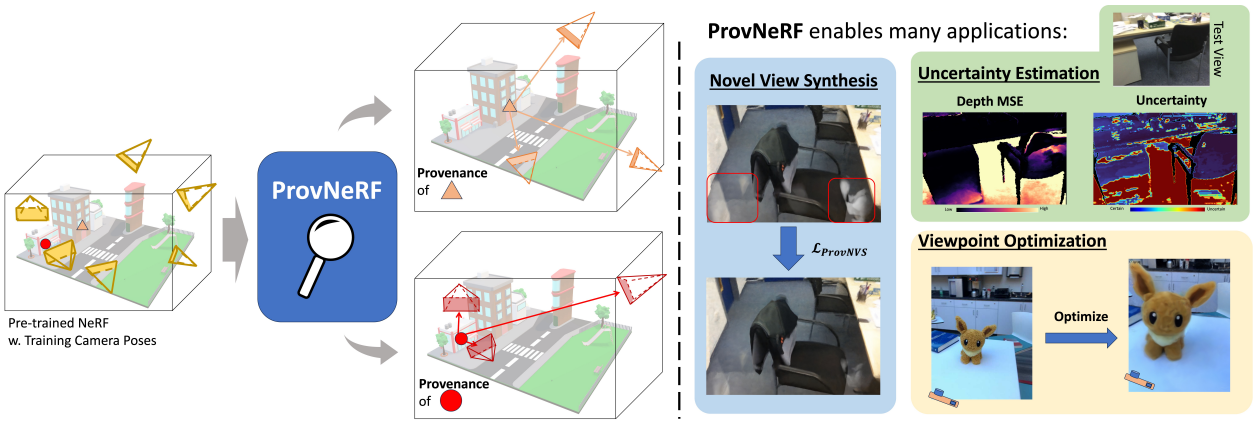

Figure 1. (**Left**) Illustration of per-point provenance. We model the origin or *provenance* of each point or *"from where it was seen."*. Our **ProvNeRF** takes as input the sparse training cameras (yellow), and outputs the *provenances* for each 3D point modeled as a stochastic process. For 3D points (orange triangle and red circle), the corresponding output provenances are illustrated by the orange the red locations, which depict from where these points were observed. (**Right**) Multiple downstream applications enabled by **ProvNeRF**, namely uncertainty estimation, criteria-based viewpoint optimization and sparse view novel view synthesis.

Abstract

Neural radiance fields (NeRFs) have gained popularity across various applications. However, they face challenges in the sparse view setting, lacking sufficient constraints from volume rendering. Reconstructing and understanding a 3D scene from sparse and unconstrained cameras is a long-standing problem in classical computer vision with diverse applications. While recent works have explored NeRFs in sparse, unconstrained view scenarios, their focus has been primarily on enhancing reconstruction and novel view synthesis. Our approach takes a broader perspective by posing the question: "from where has each point been seen?" – which gates how well we can understand and reconstruct it. In other words, we aim to determine the origin or provenance of each 3D point and its associated information under sparse, unconstrained views. We introduce ProvNeRF, a model that enriches a traditional NeRF representation by incorporating per-point provenance, modeling likely source locations for each point. We achieve this by extending implicit maximum likelihood estimation (IMLE) for stochastic processes. Notably, our method is compatible with any pre-trained NeRF model and the associated training camera poses. We demonstrate that modeling per-point provenance offers several advantages, including uncertainty estimation, criteria-based view selection, and improved novel view synthesis, compared to state-of-the-art methods. Please visit our project page at https://provnerf.github.io.

1. Introduction

Neural radiance fields (NeRFs) [36] have increasingly grown in popularity in the recent years, touching different applications in novel view synthesis [3, 5], depth estimation [11], robotics [1, 18], localization [30, 32], etc. However, NeRFs struggle in the sparse-view regime due to inadequate constraints from volume rendering alone [16, 39, 55]. Despite the challenges, such views are easier in practice to obtain by user-friendly methods, such as when capturing images in-the-wild with a smartphone. Therefore, understanding a 3D scene given *sparse* [14], *unconstrained* [49] camera views is an important problem dating back to classical computer vision with multiple applications such as in structure-from-motion [2, 31], SLAM [12, 34] and uncertainty estimation [22, 37].

Recent works explore NeRFs under the sparse, unconstrained view regime. To tackle the under-constrained problem, they incorporate priors to the NeRF optimization such as depth [11, 44, 56], local geometry [39] or global shape information [20, 43, 67]. These works, however, only focus on enabling better novel view synthesis and do not address the question of how well we understand the scene from a more holistic perspective, such as uncertainty estimation,

optimal view selection, and the like.

In this work, we show that a broader set of problems can be addressed by asking *"from where has each point been seen?"* If we know from where a point has been seen from a set of training images, these locations can be used not only for better reconstruction but also for different applications such as uncertainty estimation and view selection. This motivates us to extract the **provenance** or the *origin* of each point of NeRFs under sparse, unconstrained views.

Recent works use NeRFs to predict additional information such as semantics [70] and features [24] on top of their original per-point color and opacity values. Inspired by these, we enrich the NeRF representation by predicting the provenance of each continuous coordinate, i.e., from where each continuous coordinate has been seen. Since a point can be observed at multiple source locations, the provenance for each point cannot be modeled by a scalar field that maps each 3D location to a single deterministic value. To solve this issue, we propose to model the per-point provenance of NeRFs probabilistically as a *stochastic process*.

To model per-point provenance as a stochastic process, a key challenge is how to model the *distribution* of viewing source locations for each 3D coordinate. One option is to use explicit probabilistic models with closed-form densities, e.g. Gaussians [46]. Unfortunately, these distributions have limited expressivity as they have a fixed shape, and this is not ideal for modeling the distribution of per-point provenance as the distribution can be multimodal when a point is observed from multiple locations. Another option is to use implicit probabilistic models where the distribution is represented as a set of samples generated from a neural network. We extend implicit probabilistic models, specifically IMLE [27] (implicit maximum likelihood estimation) to handle stochastic processes by adapting the original objective to functional space and show that it is equivalent to a pointwise matching of empirical and model samples.

We dub our method **ProvNeRF**, a model that enriches the traditional NeRF representation by modeling per-point provenance as a stochastic process using an extended IMLE. Our approach can be applied to any pre-trained NeRF only with associated training camera poses. We demonstrate that modeling per-point provenance offers several advantages in multiple applications such as modeling uncertainty, criteria-based view selection as well as improvement in novel view synthesis compared to state-of-the-art works.

2. Related Works

NeRFs and their Extensions. Neural radiance fields (NeRFs) [36] have revolutionized the field of 3D reconstruction [17] and novel view synthesis [26, 45] with its powerful representation of a scene using weights of an MLP that is rendered by volume rendering [35, 57]. Follow-ups on NeRF further tackle novel view synthesis under more difficult scenarios such as unconstrained photo collections [33], unbounded [4], dynamic [29] and deformable [41] scenes, and reflective objects [6, 58]. Going beyond novel view synthesis, the NeRF representation have also shown great promise in different applications such as autonomous driving [52, 62], robotics [1, 18] and editing [63, 66]. Recent works have also extended NeRFs to model other fields in addition to color and opacity such as semantics [69, 70], normals [65], CLIP embeddings [23], image features [24] and scene flow [28]. Most of these works learn an additional function that predicts an auxillary *deterministic* output that is either a scalar or a vector, trained with extra supervision using volume rendering. In our setting, as each point can be observed from multiple different locations, the output for each point is no longer deterministic and hence is not a field. Our work also enriches NeRFs with an additional output, but models this as a stochastic process instead of a field.

Sparse View Novel View Synthesis. A well-known drawback of NeRFs is it struggles under a small number of views due to insufficient constraints in volume rendering alone. Several approaches have been proposed to train NeRFs under the sparse-view regime with regularization losses [39], semantic consistency [19], and image [59] or cost volume [8, 9, 61] feature constraints. Other works also constrain the optimization using priors from data [20, 64] or depth [44, 50, 56]. Despite addressing the setting with limited number of input views, a number of works are on object-level [19, 20, 39, 64] and limited camera baselines [9, 61] or forward-facing scenes [59], and they are not designed to tackle our desired sparse, unconstrained views setting. Recent works [44, 50, 56] have looked into improving the NeRF quality on a more difficult setting of sparse, unconstrained (outward-facing) input views by incorporating depth priors, however, their main focus is on the task of novel view synthesis. Our work is also focused on this more difficult setting, but is not limited to the task of novel view synthesis as we showcase its usability in multiple other downstream applications.

Uncertainty Modeling in NeRFs. Under the sparse view regime, 3D reconstruction is plague with uncertainty. A line of works explore uncertainty estimation in NeRFs by directly estimating a value for uncertainty [33] or computing the variance by learning a distribution of NeRFs [47, 48] or by training an ensemble [51] of NeRFs. These works however are specifically trained or designed for the task of uncertainty estimation, and they only experiment on a forward-facing camera views in the sparse setting. Our work tackles a wider range of tasks under the more difficult sparse, *unconstrained* view setting, where uncertainty estimation is one byproduct that can be derived from modeling per point provenance. A concurrent unpublished work [15] is a post-hoc framework that use Laplace approximation to quantify uncertainty by intuitively measuring how much a point can deform. Similar to [15], our work is also a post-hoc approach that can be applied to any pretrained NeRF,

but our approach is not limited to uncertainty estimation. Additionally, a number of works measure uncertainty in NeRFs for next best view augmentation [21, 25, 40, 42]. We distinguish this from our criteria-based view selection application as we are not adding more training views, but instead optimize for viewpoints from a user-specified objective using our modelled per point provenance.

3. Preliminaries

1. Neural Radiance Fields (NeRF)

A neural radiance field (NeRF) is a coordinate-based neural network that learns a field in 3D space, where each point $\boldsymbol{x} \in \mathbb{R}^3$ is of certain *opacity* and *color*. Mathematically, a NeRF is parameterized by two functions representing the two fields $\boldsymbol{F}_{\phi,\psi} = (\boldsymbol{\sigma}_\psi(\boldsymbol{x}), \boldsymbol{c}_\phi(\boldsymbol{x}, \boldsymbol{d}))$, one for opacity $\boldsymbol{\sigma}_\psi : \mathbb{R}^3 \to \mathbb{R}_+$ and one for color $\boldsymbol{c}_\phi : \mathbb{R}^3 \times \mathbb{S}^2 \to [0,1]^3$, where $\boldsymbol{d} \in \mathbb{S}^2$ is the direction from where $\boldsymbol{x}$ is viewed from. One of the key underpinnings of NeRFs is volume rendering allowing for end-to-end differentiable learning with only training images. Concretely, given a set of M images $I_1, I_2, ..., I_M$ and their corresponding camera poses $P_1, P_2, ..., P_M$, the rendered color of a pixel x is the expected color along a camera ray $\boldsymbol{r}_{i,x}(t) = \boldsymbol{o}_i + t\boldsymbol{d}_{i,x}$, where $\boldsymbol{o}_i$ is the camera origin and $\boldsymbol{d}_{i,x}$ is the ray direction for pixel x that can be computed from the corresponding camera pose P_i. The pixel value for 2D coordinate x is then given by the line integral:

$$\boldsymbol{C}_{\phi,\psi}\left(\boldsymbol{r}_{i,x}\right) = \int_{t_n}^{t_f} \boldsymbol{\sigma}_\psi\left(\boldsymbol{r}_{i,x}(t)\right) T\left(\boldsymbol{r}_{i,x}(t)\right) \boldsymbol{c}_\phi\left(\boldsymbol{r}_{i,x}(t)\right)\, dt, \tag{1}$$

where t_n, t_f defines the near and far plane, and

$$T\left(\boldsymbol{r}_{i,x}(t)\right) = \exp\left(-\int_{t_n}^{t} \sigma\left(\boldsymbol{r}_{i,x}(s)\right)\, ds\right) \tag{2}$$

is the transmittance, which represents the visibility of the point $\boldsymbol{r}_{i,x}(t)$ along the direction $\boldsymbol{d}_{i,x}$.

2. Implicit Maximum Likelihood Estimation

Leveraging on recent advantages in probabilistic models, one choice is implicit maximum likelihood estimation (IMLE) [27] that allows to handle multimodal distributions, as per point provenance is inherently multimodal, i.e. a point can be observed at multiple different locations. As an implicit probabilistic models, IMLE learns a parameterized transformation $\boldsymbol{H}_\theta(\cdot)$ of a latent random variable, e.g. a Gaussian $\boldsymbol{z} \sim \mathcal{N}(0, \mathbf{I})$, where $\boldsymbol{H}_\theta(\cdot)$ often takes the form of a neural network that output samples $\boldsymbol{w}_j = \boldsymbol{H}_\theta(z_j)$ with $\boldsymbol{w}_j \sim \mathbb{P}_\theta(W)$. Here $\mathbb{P}_\theta$ is the distribution measure of the random variable W is the pushforward of the standard Gaussian distribution measure via the transformation $\boldsymbol{H}_\theta$. Given a set of data samples $\{\hat{\boldsymbol{w}}_1, ..., \hat{\boldsymbol{w}}_N\}$, the IMLE objective is designed such that high density is assigned to each data sample $\hat{\boldsymbol{w}}_i$. This is enabled by optimizing the model parameters $\boldsymbol{H}_\theta$ such that each data sample $\hat{\boldsymbol{w}}_i$ is close to *some* model sample $\boldsymbol{w}_j$, which is shown to be equivalent to maximizing the likelihood. Concretely, if $\hat{\boldsymbol{w}}_1, \ldots, \hat{\boldsymbol{w}}_N$ are samples from the data distribution, and $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K$ are i.i.d. samples from $\mathbb{P}_\theta(W)$, the IMLE objective is given as:

$$\hat{\theta} = \arg\min_\theta \mathbb{E}_{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_K}\left[\sum_{i=1}^{N} \min_j \left\|\boldsymbol{H}_\theta(z_j) - \hat{\boldsymbol{w}}_i\right\|_2^2\right]. \tag{3}$$

4. Method

In this paper, we enrich the NeRF representation by modeling per-point *provenance*, i.e. the origin of each point, or in layman's words *from where it was seen*. As each point can be observed from multiple different positions and directions, the provenance at each point is a set rather than a single quantity. Hence it cannot be modeled as a simple function that maps each coordinate to a vector (i.e., a neural field). Instead, we propose modelling per-point provenance with a *stochastic process* indexed by coordinates $\boldsymbol{x} \in \mathbb{R}^3$, whose marginal distribution at $\boldsymbol{x}$ encodes the *provenance* at $\boldsymbol{x}$. Intuitively, the marginal distribution represents the distribution over possible locations that can see $\boldsymbol{x}$. We detail this below and show that modeling per-point provenance is beneficial for multiple downstream applications (Figure 1).

1. Provenance as a Stochastic Process

Provenance is defined as *the place of origin*. We desire to model the origin of each point or in other words, *"from where it was seen"* based on the given sparse training camera views $P_1, \ldots, P_M$ (Figure 1). We concretely define our notion of per-point provenance. If $\boldsymbol{x}$ is inside the camera frustum Π_i for view P_i, then **a provenance** of $\boldsymbol{x}$ a distance-direction tuple $(t_{i,\boldsymbol{x}}, \boldsymbol{d}_{i,\boldsymbol{x}}) \in \mathbb{R}_+ \times \mathbb{D}^3$ [1] such that

$$(t_{i,\boldsymbol{x}}, \boldsymbol{d}_{i,\boldsymbol{x}}) = \left(v_{i,\boldsymbol{x}} \left\|\boldsymbol{x} - \boldsymbol{o}_i\right\|, v_{i,\boldsymbol{x}} \frac{\boldsymbol{x} - \boldsymbol{o}_i}{\left\|\boldsymbol{x} - \boldsymbol{o}_i\right\|}\right), \tag{4}$$

where $v_{i,\boldsymbol{x}} \in [0,1]$ is the visibility of $\boldsymbol{x}$ at view P_i in order to handle occlusions. Now considering all M training views, per-point provenance becomes a set of distance-direction tuples given as follows:

$$\text{Prov}\left(\boldsymbol{x}\right) = \left\{\left(t_{i,\boldsymbol{x}}, \boldsymbol{d}_{i,\boldsymbol{x}}\right) | i = 1, \ldots, M\right\}. \tag{5}$$

To obtain the per-point provenance, we need to model $\text{Prov}\left(\boldsymbol{x}\right)$ for all $\boldsymbol{x} \in \mathbb{R}^3$. A naive solution is to model it as a deterministic field on $\mathbb{R}^3$. However, since each $\boldsymbol{x}$ can be seen by multiple training camera views, the cardinality and elements of the set $\text{Prov}\left(\boldsymbol{x}\right)$ will vary depending on $\boldsymbol{x}$. Hence, modeling it with any deterministic function is suboptimal as this limits to assigning $\text{Prov}\left(\boldsymbol{x}\right)$ to some summary statistics for the set, and this point estimate by a deterministic field can be far from any of its elements.

To address this issue, we propose to model per-point provenance as a distribution of *stochastic process* $\mathcal{D}_\theta$ indexed by points in $\mathbb{R}^3$, such that a sample $\boldsymbol{D}_\theta \sim \mathcal{D}_\theta$ is a deterministic function on $\mathbb{R}^3$ mapping each point $\boldsymbol{x}$ to one of its provenances.

[1] $\mathbb{D}^3$ denotes a solid ball in $\mathbb{R}^3$

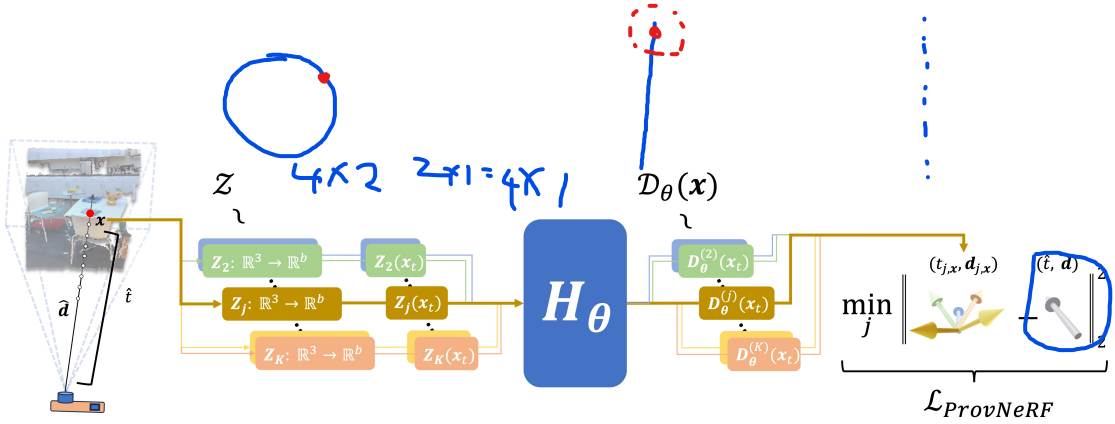

Figure 2. **Training pipeline for ProvNeRF.** For each point $\boldsymbol{x}$ seen from provenance tuple $(\hat{t}, \hat{\boldsymbol{d}})$, with direction $\boldsymbol{d}$ at distance t, we first sample K latent random functions from distribution $\mathcal{Z}$. Evaluating the latent random functions at $\boldsymbol{x}$ provides us with spatially varying latent random vectors $\{\boldsymbol{Z}_j(\boldsymbol{x})\}$. The learned transformation $\boldsymbol{H}_\theta$ then takes $\boldsymbol{Z}_j(\boldsymbol{x})$ and output a provenance sample $\boldsymbol{D}_\theta^{(j)}(\boldsymbol{x})$ from the implicit distribution $\mathcal{D}_\theta(\boldsymbol{x})$. To train $\boldsymbol{H}_\theta$ that transforms the distribution $\mathcal{Z}$ to the implicit distribution $\mathcal{D}_\theta$, we use $\mathcal{L}_{\text{ProvNeRF}}$ to match the provenance samples $\boldsymbol{D}_\theta^{(j)}(\boldsymbol{x})$ with the empirical provenance $(\hat{t}, \hat{\boldsymbol{d}})$ at point $\boldsymbol{x}$.

A stochastic process $\boldsymbol{D}_\theta$ with the index set being $\mathbb{R}^3$ is defined as a collection of uncountably many random variables (R.V.) $\{\boldsymbol{D}_\theta(\boldsymbol{x})\}_{x\in\mathbb{R}^3}$ indexed by points $\boldsymbol{x} \in \mathbb{R}^3$. Then, if each of the random variables $\boldsymbol{D}_\theta(\boldsymbol{x})$ follows distribution $\mathcal{D}_\theta(\boldsymbol{x})$, we can assign a distribution $\mathcal{D}_\theta$ to the stochastic process $\boldsymbol{D}_\theta$ simply as the joint distribution over all the R.V.'s in $\{\boldsymbol{D}_\theta(\boldsymbol{x})\}_{x\in\mathbb{R}^3}$. To model the per-point provenance using a distribution of stochastic process, we define the per-point provenance in Eq. 5 as a discrete distribution $\hat{\mathcal{D}}(\boldsymbol{x})$ over the different observations of $\boldsymbol{x}$. If $\boldsymbol{x}$ is *not* visible by training camera j, the provenance $(t_{j,\boldsymbol{x}}, \boldsymbol{d}_{j,\boldsymbol{x}})$ is assigned to the sample $(0, \boldsymbol{0})$ with probability $\frac{1}{M}$. Mathematically, we can define the per-point empirical provenance distribution as a discrete distribution over Prov $(\boldsymbol{x})$ by specifying its probability mass function, $\mathbb{P}\left(\hat{\boldsymbol{D}}(\boldsymbol{x})\right)$. Specifically, for $(\hat{t}, \hat{\boldsymbol{d}}) \in \text{Prov}(\boldsymbol{x})$, we set

$$\mathbb{P}\left(\hat{\boldsymbol{D}}(\boldsymbol{x})\right)((\hat{t}, \hat{\boldsymbol{d}})) = \begin{cases} \frac{1}{M} & \text{if } (\hat{t}, \hat{\boldsymbol{d}}) \neq (0, \boldsymbol{0}) \\ 1 - \frac{M_{\boldsymbol{x}}}{M} & \text{otherwise.} \end{cases} \tag{6}$$

Here, $M_{\boldsymbol{x}} = M - |\text{Prov}(\boldsymbol{x})| + 1$ is the number of nonzero provenances across all M training views. When viewing the above empirical distribution at each $\boldsymbol{x} \in \mathbb{R}^3$ collectively, we obtain a distribution $\hat{\mathcal{D}}$ of the empirical provenance stochastic process $\hat{\boldsymbol{D}} = \{\boldsymbol{D}(\boldsymbol{x})\}_{\boldsymbol{x}\in\mathbb{R}^3}$ that we seek to model.

For notation simplicity, we let hat $\hat{(\cdot)}$ denote the empirical distribution (i.e. $\hat{\mathcal{D}}$) and let subscript θ refer to our model distribution (i.e. $\mathcal{D}_\theta$) with parameters θ. We denote the distribution of stochastic process with a calligraphic font (e.g. $\mathcal{D}$), and the same letter bolded and in uppercase be the stochastic process itself (e.g. $\boldsymbol{D}$). We note that the marginalization of the distribution of stochastic process over all points in $\mathbb{R}^3$ except for $\boldsymbol{x}$, denoted as $\mathcal{D}(\boldsymbol{x})$, is a distribution over the provenances of $\boldsymbol{x}$. And we also have $\boldsymbol{D}(\boldsymbol{x})$, which is the random variable of provenance at $\boldsymbol{x}$, following the marginalized distribution $\mathcal{D}(\boldsymbol{x})$.

2. ProvNeRF

We propose **ProvNeRF** that models per-point provenance as a stochastic process by extending implicit probabilistic models, specifically IMLE [27], to handle stochastic processes. As detailed in Sec. 2, IMLE learns a transformation of a latent random variable to the data distribution, where each data sample is either a scalar or a vector. However in our context, since samples from the distribution of stochastic process $\hat{\mathcal{D}}$ are *functions* mapping each point to one of its provenances, we need to extend IMLE to learn a transformation $\boldsymbol{H}_\theta$ that maps a pre-defined distribution of *latent random functions*, or a distribution of latent stochastic process, $\mathcal{Z}$ to the model distribution $\mathcal{D}_\theta$.

Let $\mathcal{Z}$ be the distribution of latent stochastic process such that each sample $\boldsymbol{Z} \sim \mathcal{Z}$ is a function, mapping points $\boldsymbol{x}$ to latent vectors $\boldsymbol{z} \in \mathbb{R}^b$. Then, **ProvNeRF** learns a deterministic transformation $\mathbf{H}_\theta : \mathbb{R}^b \to \mathbb{R}_+ \times \mathbb{D}^3$ that maps each latent random function $\boldsymbol{Z} \sim \mathcal{Z}$ to a function $\boldsymbol{D}_\theta \sim \mathcal{D}_\theta$.

In this way, $\mathbf{H}_\theta$ maps each random function sample from $\mathcal{Z}$ to a function $\boldsymbol{D}_\theta$ via composition, and evaluating $\boldsymbol{D}_\theta$ at any point $\boldsymbol{x}$ would output a provenance tuple $(t, \boldsymbol{d})$ that observes $\boldsymbol{x}$. To handle complex transformations from $\mathcal{Z}$ to $\mathcal{D}_\theta$, we represent $\boldsymbol{H}_\theta$ as a neural network. Moreover, we define our latent distribution $\mathcal{Z}$ to be the concatenation of a *random linear transformation* of input $\boldsymbol{x}$ and $\boldsymbol{x}$ itself. Mathematically, each latent function sample $\boldsymbol{Z} \sim \mathcal{Z}$ is a block matrix in $\mathbb{R}^{(b+4)\times 3}$ such that

$$\boldsymbol{Z}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{z} \\ \hline \boldsymbol{I} \end{bmatrix} \boldsymbol{x}, \text{ where } \boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \lambda^2 \boldsymbol{I}) \tag{7}$$

for all $\boldsymbol{x} \in \mathbb{R}^3$. Although the distribution $\mathcal{Z}$ can be designed to have non-linear dependence on the input location $\boldsymbol{x}$, we experimentally show that this simple design choice works well across different downstream applications.

To optimize $\mathcal{D}_\theta$ using the empirical distribution $\hat{\mathcal{D}}$ in Eq. 6, we extend IMLE to model distributions of stochastic processes. We adapt Eq. 3 to functional space and show that it is equivalent to a pointwise matching of the empirical samples $\hat{\boldsymbol{D}}(\boldsymbol{x}) \sim \hat{\mathcal{D}}(\boldsymbol{x})$ and model samples $\boldsymbol{D}_\theta(\boldsymbol{x}) \sim \mathcal{D}_\theta(\boldsymbol{x})$ at each point $\boldsymbol{x}$.

3. IMLE for Stochastic Processes

To optimize the parameters θ to match $\mathcal{D}_\theta$ with the empirical distribution $\hat{\mathcal{D}}$, we construct an IMLE objective to implicitly maximize the likelihood of samples $\hat{\boldsymbol{D}} \sim \hat{\mathcal{D}}$ under $\mathcal{D}_\theta$. Similar to Eq. 3 for scalars and vectors, if we have i.i.d. functional empirical samples $\hat{\boldsymbol{D}}_1, \ldots, \hat{\boldsymbol{D}}_M$ from $\hat{\mathcal{D}}$, and model samples $\boldsymbol{D}_\theta^{(1)}, \ldots, \boldsymbol{D}_\theta^{(K)}$ from $\mathcal{D}_\theta$, we can define the functional Implicit Maximum Likelihood Estimator (fIMLE) objective as:

$$\hat{\theta} = \arg\min_\theta \mathbb{E}_{\boldsymbol{D}_\theta^{(1)}, \ldots, \boldsymbol{D}_\theta^{(K)}} \left[\sum_{i=1}^{n} \min_j \left\| \hat{\boldsymbol{D}}_i - \boldsymbol{D}_\theta^{(j)} \right\|_{L^2}^2 \right] \tag{8}$$

where

$$\left\| \hat{\boldsymbol{D}}_i - \boldsymbol{D}_\theta^{(j)} \right\|_{L^2}^2 = \int_{\mathbb{R}^3} \left\| \hat{\boldsymbol{D}}_i(\boldsymbol{x}) - \boldsymbol{D}_\theta^{(j)}(\boldsymbol{x}) \right\|_2^2 d\boldsymbol{x} \tag{9}$$

is the squared L^2 normed difference between functions $\hat{\boldsymbol{D}}_i$ and $\boldsymbol{D}_\theta^{(j)}$. Unlike the original IMLE objective (Eq. 3) that can be directly optimized using gradient descent, the fIMLE objective in Eq. 8 cannot, as the integral in Eq. 9 cannot be computed analytically in general. In our context, the empirical samples $\hat{\boldsymbol{D}}_i$ can take arbitrary forms and our model is implicit, making both not analytically computable. [2]

To get around this, we show that fIMLE is equivalent to a pointwise matching loss using the theory of calculus of variations, and unlike the functional objective (Eq. 8), the equivalent pointwise objective can be efficiently optimized with gradient descent. Specifically, as a consequence of the Euler-Lagrange equation, because the functional $\|\hat{\boldsymbol{D}}_i - \boldsymbol{D}_\theta^{(j)}\|_{L^2}^2$ does not depend on the gradient of $\boldsymbol{D}_\theta^{(j)}$ w.r.t. $\boldsymbol{x}$, minimizing the functional $\|\hat{\boldsymbol{D}}_i - \boldsymbol{D}_\theta^{(j)}\|_{L^2}^2$ is equivalent to minimizing $\|\hat{\boldsymbol{D}}_i(\boldsymbol{x}) - \boldsymbol{D}_\theta^{(j)}(\boldsymbol{x})\|_2^2$ for all $\boldsymbol{x}$.

Thus, using the above observation, we can reformulate the functional objective in Eq. 8 to minimize the pointwise difference between the empirical samples and our model samples. Specifically, the fIMLE objective is equivalent to

$$\mathcal{L}_{\text{fIMLE}} = \mathbb{E}_{\boldsymbol{D}_\theta^{(1)}, \ldots, \boldsymbol{D}_\theta^{(K)} \sim \mathcal{D}_\theta} \mathbb{E}_{\boldsymbol{x} \sim \mathcal{U}(\Omega)} \left[\sum_{i=1}^{n} \min_j \left\| \hat{\boldsymbol{D}}_i(\boldsymbol{x}) - \boldsymbol{D}_\theta^{(j)}(\boldsymbol{x}) \right\|_2^2 \right]. \tag{10}$$

Here Ω is the scene bound from the associated training cameras. We note that Eq. 10 has no integral over the function domain, but only pointwise difference between samples from $\hat{\mathcal{D}}(\boldsymbol{x})$ and $\mathcal{D}_\theta(\boldsymbol{x})$, making it efficiently optimizable with gradient descent. Hence **ProvNeRF** learns the distribution of $\mathcal{D}_\theta$ using empirical provenance samples $(\hat{t}, \hat{\boldsymbol{d}}) \sim \hat{\mathcal{D}}(\boldsymbol{x})$ at each point $\boldsymbol{x}$ using our loss function:

$$\mathcal{L}_{\textbf{ProvNeRF}} = \mathbb{E}_{\boldsymbol{D}_\theta^{(1)}, \ldots, \boldsymbol{D}_\theta^{(K)} \sim \mathcal{D}_\theta} \mathbb{E}_{\boldsymbol{x} \sim \mathcal{U}(\Omega)} \mathbb{E}_{(\hat{t}, \hat{\boldsymbol{d}}) \sim \hat{\mathcal{D}}(\boldsymbol{x})} \left[\min_j \left\| (\hat{t}, \hat{\boldsymbol{d}}) - (t_{j,\boldsymbol{x}}, \boldsymbol{d}_{j,\boldsymbol{x}}) \right\|_2^2 \right], \tag{11}$$

where $(t_{j,\boldsymbol{x}}, \boldsymbol{d}_{j,\boldsymbol{x}}) = \boldsymbol{D}_\theta^{(j)}(\boldsymbol{x})$. Recall that the empirical samples $(\hat{t}, \hat{\boldsymbol{d}})$ from $\hat{\mathcal{D}}(\boldsymbol{x})$ requires ground truth visibility $v_{i,\boldsymbol{x}}$. In practice, this is difficult to obtain especially under the sparse, unconstrained view regime. To get around this, we set the visibility of $\boldsymbol{x}$ from P_i to $v_{i,\boldsymbol{x}} = T(\boldsymbol{r}_{i,x}(t))$, i.e. the transmittance at $\boldsymbol{x}$ when rendered from camera ray $\boldsymbol{r}_{i,x}$ from a given pre-trained NeRF model.

5. Experiments

In this section, we showcase that modeling per-point provenance allows for multiple downstream applications. Our **ProvNeRF** can learn per-point provenance process $\mathcal{D}_\theta$ by optimizing $\mathcal{L}_{\textbf{ProvNeRF}}$ on an arbitrary pre-trained NeRF model with its associated training camera poses. We present experiments on different applications namely uncertainty modeling (Sec 1), criteria-based viewpoint optimization (Sec 2) and novel view synthesis (Sec 3), and then further provide an ablation study (Sec 4) on our approach [3].

1. Uncertainty Modeling

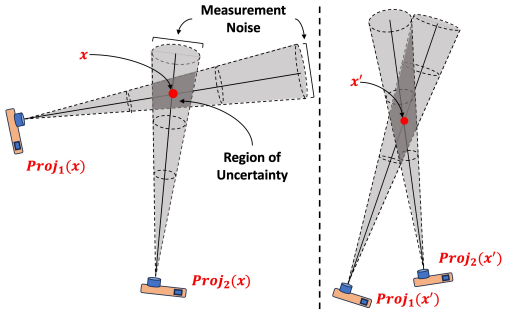

Figure 3. **Uncertainty of reconstruction [17].** The area in dark grey indicates the uncertain region of points $\boldsymbol{x}$ and $\boldsymbol{x}'$, assuming measurement noise (light grey) in the pixel observations. Notice that $\boldsymbol{x}'$ is more uncertain compared to $\boldsymbol{x}$ since the former's camera has narrower baselines. We formalize this intuition to model uncertainty using our per-point provenance process $\mathcal{D}_\theta$.

First, we show that modeling per-point provenance allows for modeling uncertainty in 3D reconstruction. In classical multiview geometry literature [17], it is stated that "a good rule of thumb is that the angle between the rays determines the accuracy of reconstruction". As illustrated in Fig. 3, 3D points $\boldsymbol{x}$ are less precisely localized along rays that are close to parallel. Using this intuition, we can derive the uncertainty, i.e. the probability of reconstructing a 3D point $\boldsymbol{x}$, of a pretrained NeRF model by asking from where is each point seen, and this is precisely what we are modeling with our **ProvNeRF**. We formalize below how we compute for the probability reconstructing a 3D point $\boldsymbol{x}$ from our learned provenance process, i.e. $\mathcal{D}_\theta(\boldsymbol{x})$.

[2] Even if we approximate the integral for our (implicit) model samples $\boldsymbol{D}_\theta$, it can be computationally very expensive since each point query to $\boldsymbol{D}_\theta$ needs one forward pass through the deep neural net $\boldsymbol{H}_\theta$.

[3] **ProvNeRF** implementation details can be found in the supplementary.

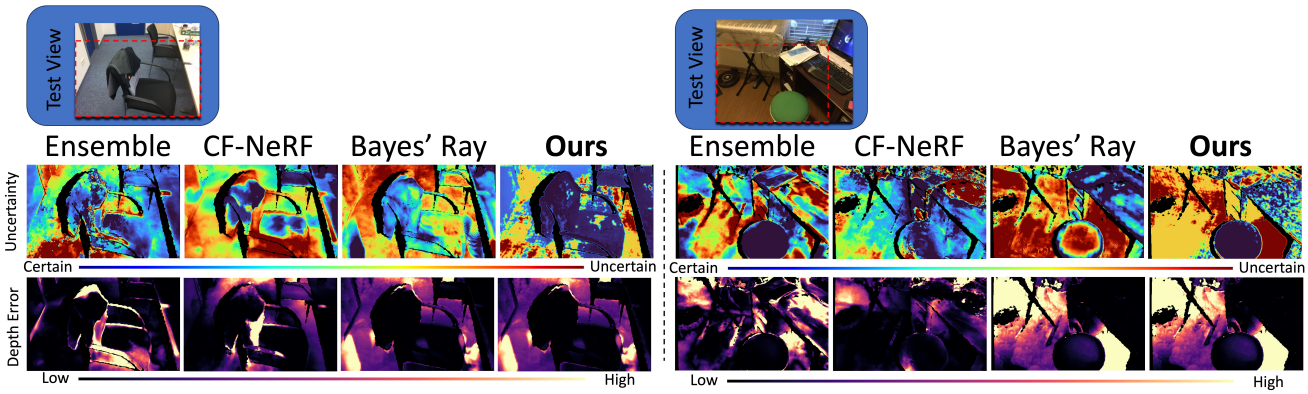

Figure 4. **Qualitative Results for Uncertainty Modeling.** The uncertainty and depth error maps are shown with color bars specified. Uncertainty values and depth errors are normalized per test image for the result to be comparable.

For a 3D point $\boldsymbol{x}$, using our **ProvNeRF**, we sample provenances $(t_1, \boldsymbol{d}_1), \ldots, (t_K, \boldsymbol{d}_K)$ from the distribution $\mathcal{D}_\theta(\boldsymbol{x})$. For each provenance sample $(t_j, \boldsymbol{d}_j)$, we have a corresponding pixel observation $x_j = \text{Proj}_j(\boldsymbol{x})$ of $\boldsymbol{x}$ in pseudo image $\tilde{I}_j$ defined by a camera with origin $(\boldsymbol{x} - t_j \boldsymbol{d}_j)$ and principal axis $\boldsymbol{d}_j$ [4]. Following chapter 12.6 of [17], the notion of uncertainty is defined to be the probability of a 3D point given its noisy 2D observations or projections in corresponding images. Specifically, for $\boldsymbol{x}$ and its pixel observations $x_1, \ldots, x_K$ in $\tilde{I}_1, \ldots, \tilde{I}_K$, the probability of any pixel $x \in \tilde{I}_j$ given $\boldsymbol{x}$ has a Gaussian PDF given as

$$\mathbb{P}(x|\boldsymbol{x}) = \frac{1}{2\pi\sigma^2} \exp\left[-\frac{\|x - \text{Proj}_j(\boldsymbol{x})\|^2}{2\sigma^2}\right]. \tag{12}$$

The posterior probability of $\boldsymbol{x}$ its pixel observations is derived as follows if we assume the 2D measurement noises are independent given $\boldsymbol{x}$ [5]:

$$\begin{aligned}\mathbb{P}(\boldsymbol{x}|x_1, \ldots, x_K) &= \frac{\mathbb{P}(x_1, \ldots, x_K|\boldsymbol{x})\,\mathbb{P}(\boldsymbol{x})}{\mathbb{P}(x_1, \ldots, x_K)} \\ &= \frac{\mathbb{P}(\boldsymbol{x}) \prod_{j=1}^K \mathbb{P}(x_j|\boldsymbol{x})}{\mathbb{E}_{\boldsymbol{x}' \sim \mathbb{P}}\left[\prod_{j=1}^K \mathbb{P}(x_j|\boldsymbol{x}')\right]}\end{aligned} \tag{13}$$

If we assume a uniform prior distribution of $\mathbb{P}(\boldsymbol{x})$ over the given scene bound, Eq. 13 can be efficiently computed using importance sampling [6]. Intuitively, Eq. 13 corresponds to how well a point is triangulated based on the training viewpoints allowing for modeling uncertainty on NeRFs. We apply our **ProvNeRF** to pre-trained NeRF models [44, 56] and compute for the likelihood of each 3D point $\boldsymbol{x}$ within the scene bound Ω using Eq. 13. See supplementary for details on dataset, metrics and baselines.

Results. Tab. 1 shows the quantitative results on Scannet [10] and Matterport3D [7]. We measure the negative log-likelihood (NLL) of the ground truth surface under the distributions given each model's uncertainty prediction. Since our **ProvNeRF** can be applied to any pre-trained NeRF module, we plug it in to SCADE [56] for Scannet and to DDP [44] for Matterport3D, which are the state-of-the-art approaches for reconstruction in each dataset. We see that our approach achieves the best NLL across all scenes in both datasets by a margin [7]. This is due to our ability to compute for a more fundamentally grounded posterior likelihood from classical multiview geometry [17], whereas both CF-NeRF and Bayes' Rays require an approximation of the true posterior likelihood. As shown in Figure 1, our uncertainty map is able to correctly assign regions with low depth error as certain (tabletop and chair), and regions with high depth error as with high uncertainty (cabinet and floor). Figure 4 shows qualitative comparisons between baselines and our method's uncertainty estimation. Notice that in both examples, our method's certain regions (blue regions) mostly have a low depth error (the tabletop in the left example and the chair in both). This is because our formulation only assigns a region to be certain if it is well triangulated and has a smaller uncertain region (See Figure 3). On the other hand, baselines struggle to do so (the office chair in the left example for Ensemble, the underneath the piano in the right example of CF-NeRF, and the lower left corner of the floor in the right example for Bayes' Ray) because they either use the empirical approximation from data, which can be error-prone, or use a rough Gaussian approximation of the ground truth posterior likelihood.

	Scannet	**Matterport**
Ensemble	7.71	63.0
CF-NeRF [48]	660	507
Bayes' Rays [15]	5.47	5.49
Ours	**-4.40**	**-10.4**

Table 1. **Quantitaive Results for Uncertainty Modeling.** The average negative log-likelihood is reported.

[4] Given a camera pose and its intrinsics, we can project a 3D point to get its 2D pixel observation.

[5] See supplementary for details.

[6] Details can be found in the supplementary.

[7] See supplementary for per-scene breakdown.

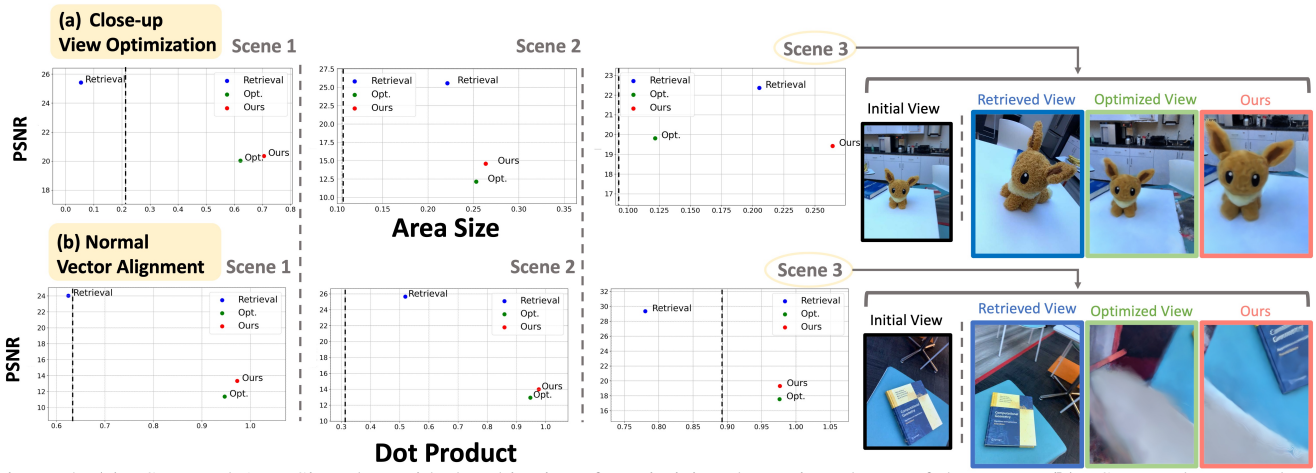

Figure 5. **(a)** PSNR and Area Size plots with the objective of maximizing the projected area of the target. **(b)** PSNR and Dot Product plots the objective of maximizing the dot product between the viewing angle and the target object's normal. The dotted lines denote the objective for the initial views. *(Right)* We show the final view comparison of our provenance-aided viewpoint selection compared to the baselines under the two objectives in scene 3. Notice that our method (in red) is able to arrive at an objective maximizing view while retaining reconstruction quality. On the other hand, both the retrieval and optimization baselines fail to balance between the two.

2. Criteria-based Viewpoint Optimization

We further show that our per-point provenance can be applied to criteria-based viewpoint selection, which leverages a neural rendering framework to determine the most favorable camera perspective based on a predefined criterion. For instance, this can encompass orienting the camera to align with the normal vector of a specified target or achieving a detailed close-up view of the target. The first criterion can be attained by minimizing the negative dot product between the camera's principal axis and the object's normal, whereas the second can be achieved through the minimization of the negative area of the 2D projection from a 3D plane encasing the object. A naive solution is to directly run gradient descent to optimize each corresponding objective loss $\mathcal{L}_{\text{obj}}$.

We propose that the integration of two novel viewing loss functions within our model enhances the gradient's informativeness, thereby facilitating an approximation to the objective that is superior to conventional methods. More precisely, for each 3D point $\boldsymbol{x}_i$ lying on the object of interest, using our **ProvNeRF**, we sample provenances $(t_1^{(i)}, \boldsymbol{d}_1^{(i)}), \ldots, (t_K^{(i)}, \boldsymbol{d}_K^{(i)})$ from the distribution $\mathcal{D}_\theta(\boldsymbol{x}_i)$. Each distance-direction tuple sample $(t_j^{(i)}, \boldsymbol{d}_j^{(i)})$ gives a location $\boldsymbol{y}_j^{(i)} = \boldsymbol{x}_i - t_j^{(i)} \boldsymbol{d}_j^{(i)}$ from which $\boldsymbol{x}_i$ is seen, based on this, we define an additional loss $\mathcal{L}_{\text{select}} = -(\mathcal{L}_c + \mathcal{L}_d)$ as follows:

$$\mathcal{L}_c = \sum_i \max_j \|\boldsymbol{y}_j^{(i)} - \boldsymbol{c}\|_2^2; \; \mathcal{L}_d = \sum_i \max_j \left(R_{:,3}^\top \boldsymbol{d}_j^{(i)} \right), \quad (14)$$

where $[R|\boldsymbol{t}]$ denotes the camera pose, and $R_{:,3}$ corresponds to the third column of R, reflecting the principal axis of the camera in the global coordinate system. Intuitively, $\mathcal{L}_c$ seeks to align the camera center with the network's prediction, whereas $\mathcal{L}_d$ aims to orient the principal axis of the camera toward the points. By running gradient descent on $\mathcal{L}_{\text{select}} + \mathcal{L}_{\text{obj}}$ from an initial viewpoint of the target object, our method can achieve better PSNR while moving closer to the desired criteria. We benchmark against two baselines: 1) a naive approach that exhaustively searches all training views to locate the one that minimizes L_{obj}, and 2) a direct optimization approach that applies gradient descent solely on L_{obj} without considering L_{select}. [8]

	PSNR (↑)	SSIM (↑)	LPIPS (↓)
NerfingMVS [61]	16.29	0.626	0.502
IBRNet [59]	13.25	0.529	0.673
MVSNeRF [9]	15.67	0.533	0.635
DS-NeRF [11]	20.85	0.713	0.344
DDP [44]	19.29	0.695	0.368
SCADE [56]	21.54	0.732	0.292
DäRF [50]	21.58	**0.765**	0.325
Ours	**21.68**	0.733	**0.291**

Table 2. **Novel View Synthesis Results.**

Results. Both the quantitative (Left) and qualitative (Right) outcomes, presented in Figure 5, demonstrate that our approach achieves a superior balance between PSNR and the intended objective [9]. Although the naive retrieval baseline attains a higher PSNR, it fails to satisfactorily meet the desired objective, especially when training views are sparse and limited.

3. Novel View Synthesis

Finally, we show how modeling per-point provenance improves the task of sparse, unconstrained novel view synthesis. The idea is the samples from the modelled provenance

[8] See supplementary for implementation details.

[9] To compute for PSNR at arbitrary poses, we train an oracle NeRF using the full captured sequence. We use Nerfacto [53] as the oracle NeRF's representation.

Figure 6. **Visualization of Effect of $\mathcal{L}_{\text{ProvNVS}}$.**

process for each point $\boldsymbol{x}$ in the scene gives locations from where $\boldsymbol{x}$ should be equally visible. We use this intuition to improve sparse view novel view synthesis in NeRFs by employing an additional loss on each training ray.

Concretely, starting from a pretrained NeRF model, we sample visible points $\boldsymbol{x}_1, ..., \boldsymbol{x}_N$ for a training camera ray parameterized as $\hat{\boldsymbol{r}}_x(t)$, here we denote visible point $\boldsymbol{x}_i = \hat{\boldsymbol{r}}_x(\hat{t}_i)$ [10]. For each visible point $\boldsymbol{x}_i$, using our **ProvNeRF**, we sample provenances $(t_1^{(i)}, \boldsymbol{d}_1^{(i)}), \ldots, (t_K^{(i)}, \boldsymbol{d}_K^{(i)})$ from $\mathcal{D}_\theta(\boldsymbol{x}_i)$. Then each distance-direction tuple sample $(t_j^{(i)}, \boldsymbol{d}_j^{(i)})$ gives a location $\boldsymbol{y}_j^{(i)} = \boldsymbol{x}_i - t_j^{(i)}\boldsymbol{d}_j^{(i)}$ from which $\boldsymbol{x}_i$ is seen, and that means $\boldsymbol{x}$ should be equally visible when rendered from ray parameterized as $\boldsymbol{r}_x^{(i)}(t) = \boldsymbol{y}_j^{(i)} + t\boldsymbol{d}_j^{(i)}, \forall j$. With this, we can then define our additional provenance loss for novel view synthesis as follows:

$$\mathcal{L}_{\text{ProvNVS}} = \sum_{i=1}^{N}\sum_{j=1}^{K}\left[\alpha + T(\boldsymbol{r}_x^{(i)}(t_j^{(i)})) - T(\hat{\boldsymbol{r}}_x(\hat{t}_i))\right]_+, \tag{15}$$

where $[\ldots]_+$ denotes the hinge loss and α is a constant margin parameter. Intuitively, $\mathcal{L}_{\text{ProvNVS}}$ encourages the transmittance at $\boldsymbol{x}_i$ along rays given by the modeled provenances from our **ProvNeRF** to be at least the transmittance at $\boldsymbol{x}_i$ along the training camera ray. This additional loss can be used on pretrained NeRF models to further improve performance. We apply **ProvNeRF** to a recent sparse view NeRF model [56] show state-of-the-art results on sparse, unconstrained novel view synthesis [11]. See supplement for details on dataset, metrics and baselines.

Results. As shown in Table 2, our approach outperforms the state-of-the-art baselines in the task of sparse, unconstrained novel view synthesis on both the standard Scannet [10] and Matterport [7] scenes, highlighting the utility of modeling per-point provenance. Qualitative examples are also shown in Fig. 6. We see that compared to the baseline, whose geometry is already relatively crisp, our $\mathcal{L}_{\text{ProvNVS}}$ can further improve its NVS quality by removing additional cloud artifacts, as shown in the encircled regions. Note that this improvement comes essential for free as the training of **ProvNeRF** only requires the pre-trained NeRF model and the training camera poses.

[10] Note that visible points $\boldsymbol{x}_i$ are points with transmittance greater than a selected threshold λ, i.e. $T(\boldsymbol{r}_x(\hat{t}_i)) \geq \lambda$, based on the NeRF.

[11] More implementation details can be found in supplementary.

	AP (↑)	AUC (↑)
Deterministic Field	0.163	0.168
Gaussian-based w/ $C = 2$	0.537	0.539
Gaussian-based w/ $C = 5$	0.629	0.631
VAE-based	0.323	0.325
ProvNeRF w/ Spatial Inv. $\mathcal{Z}$	<u>0.742</u>	<u>0.744</u>
Ours	**0.745**	**0.747**

Table 3. **Ablation Results on Scannet.**

4. Ablation Study

We validate the design choices of our **ProvNeRF** by measuring the average precision (AP) [13] and area under the curve (AUC) [13] of the predicted samples $(t, \boldsymbol{d})$ against ground truth observations $(\hat{t}, \hat{\boldsymbol{d}})$ [12].

Deterministic Field v.s. Stochastic Process. We first validate the importance of modeling per-point provenance as a stochastic process rather than a deterministic field. We replace our distribution of stochastic process $\mathcal{D}_\theta$ with a deterministic field parameterized by a neural network. Table 3 shows the importance of modeling per-point provenance as a stochastic process. Since a point can be observed at multiple locations, modeling a stochastic process allows each provenance distribution $\mathcal{D}_\theta(\boldsymbol{x})$ to assign high likelihood to multiple observations in contrast to a deterministic field that can only map each $\boldsymbol{x}$ to a single provenance output.

Choice of Probabilistic Model. We also validate our choice of using IMLE [27] as our probabilistic model. We first ablate using an explicit probabilistic model for the provenance process, where the distribution assumes a particular form, e.g. a mixture of C Gaussian. Table 3 shows results for $C = 2, 5$, where although the performance improves as we increase C, the simple Gaussian assumption still limits the expressivity of the outputs. We also ablate on using a VAE-based model for provenance. On the other hand, our choice of using IMLE, which is an implicit probabilistic model, enables capturing a more complex distribution with a learned transformation $\boldsymbol{H}_\theta$ as shown in Table 3.

Choice of Random Function $\mathcal{Z}$. Lastly, we validate the choice of our latent stochastic process $\mathcal{Z}$. We ablate our choice of $\mathcal{Z}$ with instead using a spatially invariant latent stochastic process $\mathcal{Z}^\star$, where we set $\mathcal{Z}^\star(\boldsymbol{x}) = [\boldsymbol{\varepsilon}, \boldsymbol{x}] \forall x$. Here, $\boldsymbol{\varepsilon}$ is a Gaussian noise vector in $\mathbb{R}^d$. Table 3 shows the comparison between $\mathcal{D}_\theta$ obtained by transforming $\mathcal{Z}$ (**Ours**) and transforming $\mathcal{Z}^\star$ (**Spatial Inv.** $\mathcal{Z}$). We see that using a spatially varying latent stochastic process further increases the expressivity of our model.

[12] See supplementary for metric and ablation implementation details.

6. Conclusion

We present ProvNeRF, a model that enhances the traditional NeRF representation by incorporating per-point provenance through an extension of IMLE for stochastic processes. ProvNeRF can be easily applied to any pre-trained NeRF model with associated training camera poses. We showcase the advantages of modeling per-point provenance in various downstream applications, including uncertainty modeling, criteria-based view selection, and improved novel view synthesis compared to existing state-of-the-art methods.

7. Acknowledgement

This work is supported by ARL grant W911NF-21-2-0104, a Vannevar Bush Faculty Fellowship, an Apple Scholars in AI/ML PhD Fellowship, a Snap Research Fellowship, the Outstanding Doctoral Graduates Development Scholarship of Shanghai Jiao Tong University, and the Natural Sciences and Engineering Research Council of Canada.

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*, 7(2): 4606–4613, 2022. 1, 2

[2] Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz, and Richard Szeliski. Building rome in a day. In *ICCV*, 2009. 1

[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. In *ICCV*, 2021. 1

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

[5] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. *ICCV*, 2023. 1

[6] Sai Bi, Zexiang Xu, Pratul Srinivasan, Ben Mildenhall, Kalyan Sunkavalli, Miloš Hašan, Yannick Hold-Geoffroy, David Kriegman, and Ravi Ramamoorthi. Neural reflectance fields for appearance acquisition. *arXiv preprint arXiv:2008.03824*, 2020. 2

[7] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *3DV*, 2017. 6, 8, 1

[8] Di Chang, Aljaž Božič, Tong Zhang, Qingsong Yan, Yingcong Chen, Sabine Süsstrunk, and Matthias Nießner. Rc-mvsnet: Unsupervised multi-view stereo with neural rendering. In *ECCV*, 2022. 2

[9] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *CVPR*, 2021. 2, 7

[10] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017. 6, 8, 1, 7

[11] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. In *CVPR*, 2022. 1, 7

[12] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3):611–625, 2018. 1

[13] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010. 8

[14] Yasutaka Furukawa and Carlos Hernández. 2015. 1

[15] Lily Goli, Cody Reading, Silvia Sellán, Alec Jacobson, and Andrea Tagliasacchi. Bayes' rays: Uncertainty quantification in neural radiance fields. 2023. 2, 6, 1

[16] Guangcong, Zhaoxi Chen, Chen Change Loy, and Ziwei Liu. Sparsenerf: Distilling depth ranking for few-shot novel view synthesis. *ICCV*, 2023. 1

[17] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 edition, 2004. 2, 5, 6

[18] Jeffrey Ichnowski*, Yahav Avigal*, Justin Kerr, and Ken Goldberg. Dex-NeRF: Using a neural radiance field to grasp transparent objects. In *Conference on Robot Learning (CoRL)*, 2020. 1, 2

[19] Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting nerf on a diet: Semantically consistent few-shot view synthesis. In *ICCV*, 2021. 2

[20] Wonbong Jang and Lourdes Agapito. Codenerf: Disentangled neural radiance fields for object categories. In *ICCV*, 2021. 1, 2

[21] Liren Jin, Xieyuanli Chen, Julius Rückin, and Marija Popović. Neu-nbv: Next best view planning using uncertainty estimation in image-based neural rendering. In *IROS*, 2023. 3

[22] Berk Kaya, Suryansh Kumar, Carlos Oliveira, Vittorio Ferrari, and Luc Van Gool. Uncertainty-aware deep multi-view photometric stereo. In *CVPR*, 2022. 1

[23] Justin Kerr, Chung Min Kim, Ken Goldberg, Angjoo Kanazawa, and Matthew Tancik. Lerf: Language embedded radiance fields. In *ICCV*, 2023. 2

[24] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing nerf for editing via feature field distillation. In *NeurIPS*, 2022. 2

[25] Georgios Kopanas and George Drettakis. Improving nerf quality by progressive camera placement for unrestricted navigation in complex environments. In *Vision, Modeling, and Visualization*, 2023. 3

[26] K.N. Kutulakos and S.M. Seitz. A theory of shape by space carving. In *ICCV*, 1999. 2

[27] Ke Li and Jitendra Malik. Implicit maximum likelihood estimation, 2018. 2, 3, 4, 8

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

[29] Zhengqi Li, Qianqian Wang, Forrester Cole, Richard Tucker, and Noah Snavely. Dynibar: Neural dynamic image-based rendering. In *CVPR*, 2023. 2

[30] Jianlin Liu, Qiang Nie, Yong Liu, and Chengjie Wang. Nerf-loc: Visual localization with conditional neural radiance field. In *ICRA*, 2023. 1

[31] Sheng Liu, Xiaohan Nie, and Raffay Hamid. Depth-guided sparse structure-from-motion for movies and tv shows. In *CVPR*, 2022. 1

[32] Dominic Maggio, Marcus Abate, Jingnan Shi, Courtney Mario, and Luca Carlone. Loc-nerf: Monte carlo localization using neural radiance fields, 2022. 1

[33] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *CVPR*, 2021. 2

[34] Hidenobu Matsuki, Raluca Scona, Jan Czarnowski, and Andrew J. Davison. Codemapping: Real-time dense mapping for sparse slam using compact scene representations. *IEEE Robotics and Automation Letters*, 2021. 1

[35] Nelson Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995. 2

[36] 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. In *ECCV*, 2020. 1, 2

[37] Daniel Morris, Kenichi Kanatani, and Takeo Kanade. Uncertainty modeling for optimal structure from motion. In *Internationa Workshop of Vision Algorithms,*, 2000. 1

[38] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, 2022. 12

[39] 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 *CVPR*, 2022. 1, 2

[40] Xuran Pan, Zihang Lai, Shiji Song, and Gao Huang. Activenerf: Learning where to see with uncertainty estimation. In *ECCV*, 2022. 3

[41] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. 2021. 2

[42] Yunlong Ran, Jing Zeng, Shibo He, Jiming Chen, Lincheng Li, Yingke Chen, Gim Hee Lee, and Qi Ye. Neurar: Neural uncertainty for autonomous 3d reconstruction with implicit neural representations. *IEEE Robotics and Automation Letters*, 2022. 3

[43] Konstantinos Rematas, Ricardo Martin-Brualla, and Vittorio Ferrari. Sharf: Shape-conditioned radiance fields from a single view. In *ICML*, 2021. 1

[44] Barbara Roessle, Jonathan T. Barron, Ben Mildenhall, Pratul P. Srinivasan, and Matthias Nießner. Dense depth priors for neural radiance fields from sparse input views. In *CVPR*, 2022. 1, 2, 6, 7

[45] S.M. Seitz and C.R. Dyer. Photorealistic scene reconstruction by voxel coloring. In *CVPR*, 1997. 2

[46] Silvia Sellán and Alec Jacobson. Stochastic poisson surface reconstruction. *ACM Transactions on Graphics*, 2022. 2

[47] J. Shen, A. Ruiz, A. Agudo, and F. Moreno-Noguer. Stochastic neural radiance fields: Quantifying uncertainty in implicit 3d representations. In *3DV*, 2021. 2, 1, 6

[48] Jianxiong Shen, Antonio Agudo, Francesc Moreno-Noguer, and Adria Ruiz. Conditional-flow nerf: Accurate 3d modelling with reliable uncertainty quantification. In *ECCV*, 2022. 2, 6, 1

[49] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: Exploring photo collections in 3d. *ACM Trans. Graph.*, 2006. 1

[50] Jiuhn Song, Seonghoon Park, Honggyu An, Seokju Cho, Min-Seop Kwak, Sungjin Cho, and Seungryong Kim. Därf: Boosting radiance fields from sparse inputs with monocular depth adaptation. In *NeurIPS*, 2023. 2, 7

[51] Niko Sünderhauf, Jad Abou-Chakra, and Dimity Miller. Density-aware nerf ensembles: Quantifying predictive uncertainty in neural radiance fields. In *ICRA*, 2022. 2, 1, 6

[52] 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 *CVPR*, 2022. 2

[53] 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. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Proceedings*. ACM, 2023. 7

[54] Zachary Teed and Jia Deng. Tangent space backpropagation for 3d transformation groups. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10338–10347, 2021. 6

[55] Prune Truong, Marie-Julie Rakotosaona, Fabian Manhardt, and Federico Tombari. Sparf: Neural radiance fields from sparse and noisy poses. CVPR, 2023. 1

[56] Mikaela Angelina Uy, Ricardo Martin-Brualla, Leonidas Guibas, and Ke Li. Scade: Nerfs from space carving with ambiguity-aware depth estimates. In *CVPR*, 2023. 1, 2, 6, 7, 8

[57] Mikaela Angelina Uy, George Kiyohiro Nakayama, Guandao Yang, Rahul Krishna Thomas, Leonidas Guibas, and Ke Li. Nerf revisited: Fixing quadrature instability in volume rendering. In *NeurIPS*, 2023. 2

[58] 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. In *CVPR*, 2022. 2

[59] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *CVPR*, 2021. 2, 7

[60] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 7

[61] 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. In *ICCV*, 2021. 2, 7

[62] Yuanbo Xiangli, Linning Xu, Xingang Pan, Nanxuan Zhao, Anyi Rao, Christian Theobalt, Bo Dai, and Dahua Lin. Bungeenerf: Progressive neural radiance field for extreme multi-scale scene rendering. In *ECCV*, 2022. 2

[63] Tianhan Xu and Tatsuya Harada. Deforming radiance fields with cages. In *ECCV*, 2022. 2

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

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

[66] Yu-Jie Yuan, Yang-Tian Sun, Yu-Kun Lai, Yuewen Ma, Rongfei Jia, and Lin Gao. Nerf-editing: Geometry editing of neural radiance fields. In *CVPR*, 2022. 2

[67] Yu-Jie Yuan, Yang-Tian Sun, Yu-Kun Lai, Yuewen Ma, Rongfei Jia, Leif P. Kobbelt, and Lin Gao. Interactive nerf geometry editing with shape priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. 1

[68] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, pages 586–595, 2018. 7

[69] Xiaoshuai Zhang, Abhijit Kundu, Thomas Funkhouser, Leonidas Guibas, Hao Su, and Kyle Genova. Nerflets: Local radiance fields for efficient structure-aware 3d scene representation from 2d supervision. *CVPR*, 2023. 2

[70] Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew Davison. In-place scene labelling and understanding with implicit scene representation. In *ICCV*, 2021. 2

S. Supplementary Materials

We provide additional results and visualizations and elaborate on implementation, metrics, baselines, and dataset details for **ProvNeRF** in this supplementary material. We organize the supplement as the following: in Sec. S.1, we detail the implementation detail of **ProvNeRF**. In Sec. S.1, we elaborate on the metrics, ablation, and implementation details and further provide additional visualizations on both the Scannet and Matterport datasets for our uncertainty modeling application. In Sec. S.1, we explain the implementation details and provide additional visualizations of our criteria-based viewpoint optimization compared to the baseline. In Sec. S.1, we provide further results on Tanks and Temple Dataset and show further qualitative results for the sparse-view novel view synthesis application. In Sec. S.1, we conduct further ablation study and explain the implementation details of the ablations introduced in the main paper. We also further provide more details on the derivation of our fIMLE objective in Sec S.1. Lastly, Sec. S.1, we talk about the limitation and future works for **ProvNeRF**.

S.1. ProvNeRF Implementation Details

Since our method can be plugged into any pre-trained NeRF model, we apply our method to SCADE [56] and DDP [44], two recent state-of-the-art sparse, unconstrained NeRF models. On each of the backbones, we append an additional provenance prediction branch that is a 3-layer MLP. The provenance prediction branch takes the output of the latent random functions $\boldsymbol{Z}(\boldsymbol{x})$ and outputs a provenance tuple $(t_{\boldsymbol{x}}, \boldsymbol{d}_{\boldsymbol{x}})$ as defined in the main. In practice, the provenance branch outputs $t_{\boldsymbol{x}}$ as a normalized distance by the near and far plane of the training cameras, and the direction $\boldsymbol{d}_{\boldsymbol{x}}$ is a 3D vector with its norm indicating the visibility/transmittance of $\boldsymbol{x}$ when seen from $\boldsymbol{y} = \boldsymbol{x} - t\boldsymbol{d}_{\boldsymbol{x}}$.

To train the distribution $\mathcal{D}_\theta$, we freeze the weights of the pretrained NeRF model and run stochastic gradient descent on the provenance prediction branch using our $\mathcal{L}_{\text{ProvNeRF}}$ loss (defined in Eq. 11 of the main paper) for 100K iterations. In each training iteration, we pick a training camera pose at random and shoot 256 rays within the image uniformly. The loss in Eq. 10 in main is then computed and backpropagated for all coarse and fine samples along the ray against the direction and camera distance from the rays. The set of random functions is resampled every 1000 iterations. The size of the provenance function pool is set to $K = 16$ and the latent space for latent random functions $\mathbb{Z}$ is set to $b = 32$. Additionally, following the usual NeRF training, the input and output of the latent random function both use the positionally encoded $\boldsymbol{x}$ instead of the actual 3D coordinate. The entire training process takes around 8 hours on an NVIDIA A5000. Once $\mathcal{D}_\theta$ is trained, provenances of each point $\boldsymbol{x}$ can be sampled by sampling random functions $\boldsymbol{Z}$ from the distribution $\mathcal{Z}$ and evaluate $\boldsymbol{H}_\theta \circ \boldsymbol{Z}(\boldsymbol{x})$. To obtain the visible provenances, we only use samples with the norm of the predicted directions $\boldsymbol{d}_{\boldsymbol{x}} > 0.7$.

S.1. Uncertainty Estimation

Datasets & Baselines We evaluate uncertainty quantification on two sparse, unconstrained views (outward-facing) datasets – Scannet [10] and Matterport3D [7] using the training and test splits released by [44]. Each dataset contains three scenes with $17 - 36$ training and 8 test images at a room scale. We compare our approach with the previous state-of-the-art CF-NeRF [48], concurrent work Bayes' Rays [15] and a standard derivation of uncertainty by training an ensemble of NeRFs [51]. We use the same backbone NeRF model as ours when comparing with Bayes' Rays and Ensemble. Following previous works [47, 48, 51], we report the negative log-likelihood (NLL) of $\mathbb{P}(\boldsymbol{x}|\{I\})$ to measure uncertainty. We measure the NLL of the ground truth surface under the distributions given each model's uncertainty prediction. Specifically, for Ensemble, we run 10 identical models using different seeds and compute NLL of the ground truth depth under the empirical Gaussian distribution given by the 10 different depth renderings of each

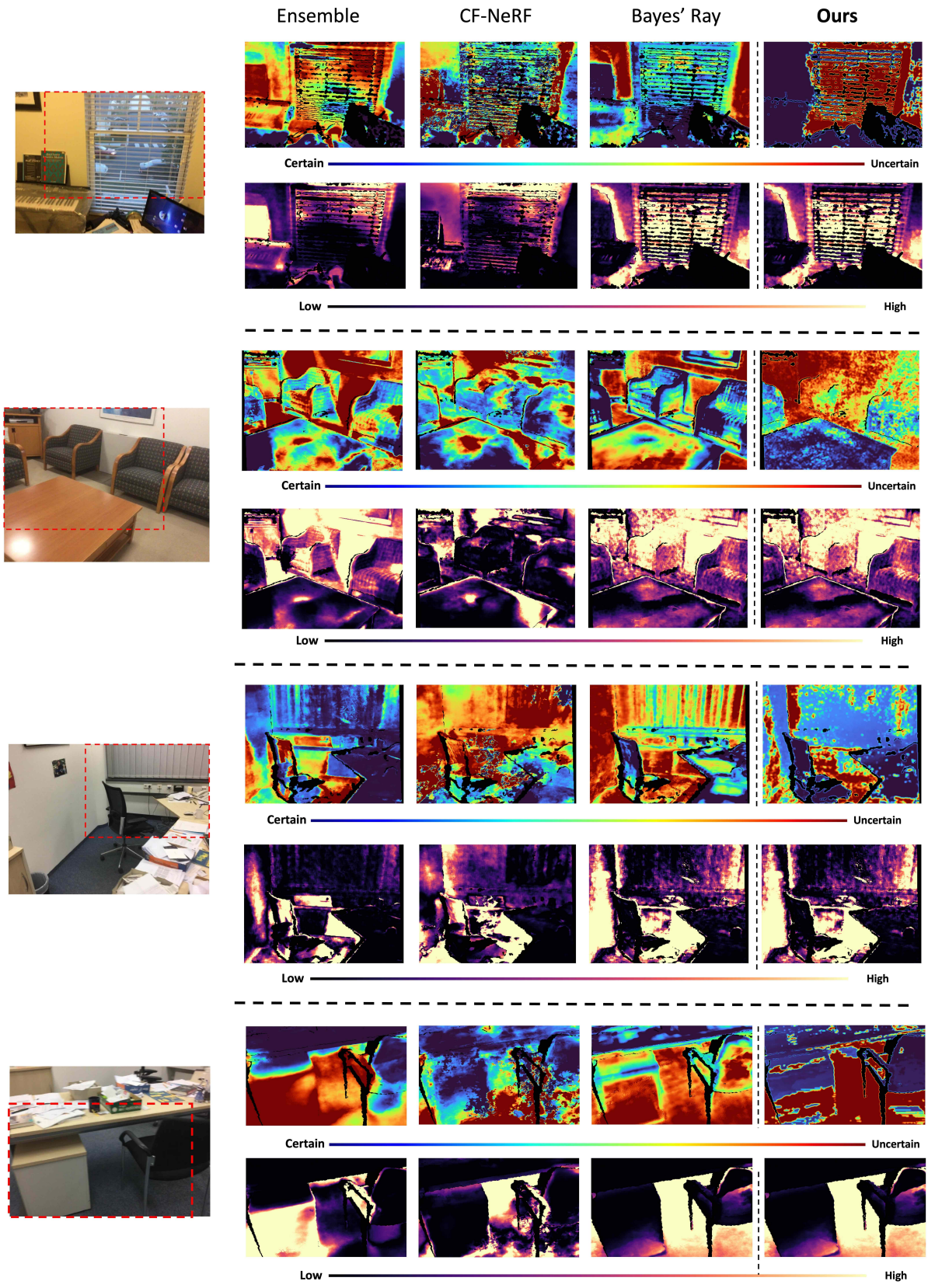

Figure S.1. **Visualization of Uncertainty Estimation on the Scannet Dataset**.

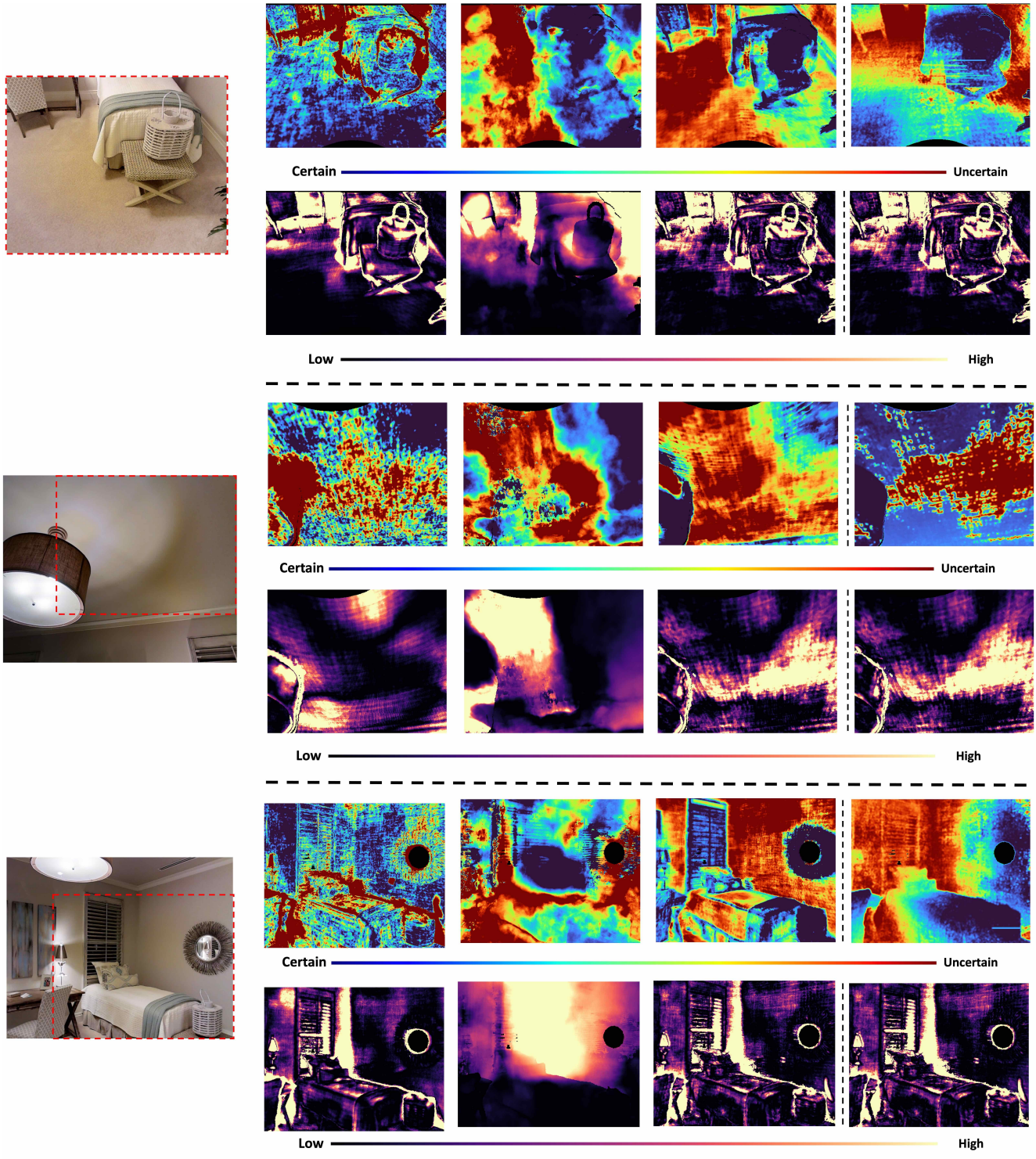

Figure S.2. **Visualization of Uncertainty Estimation on the Matterport Dataset**.

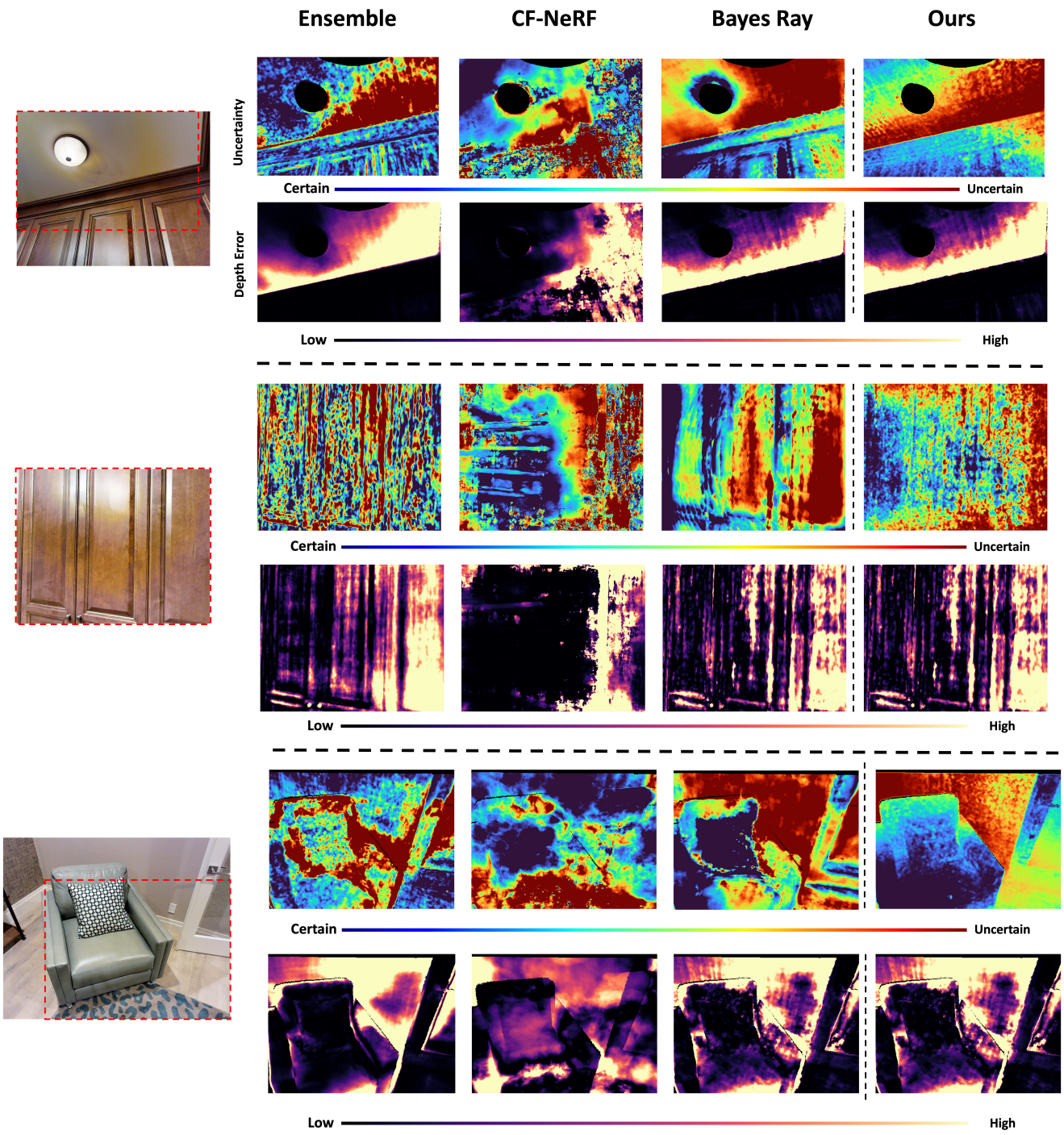

Figure S.3. **Visualization of Uncertainty Estimation on the Matterport Dataset**.

test view. For Bayes' Ray, we obtain the posterior likelihood by evaluating their Laplace approximated distribution $p(\boldsymbol{\theta}|I)$ at $\theta = 0$. This is equivalent to evaluating the likelihood of 0 under $\mathcal{N}(\mathbf{0}, \text{Diag}(\sigma_x, \sigma_y, \sigma_z))$ for each 3d location $\boldsymbol{x} = (x, y, z)$. For a fair comparison, we set Bayes' Ray's prior distribution, which is a zero-centered Gaussian, to have a standard deviation that is half of the diagonal length of the scene bound given by the training camera poses. Lastly for CF-NeRF, we follow their default setting and evaluate the ground truth depth under the empirical Gaussian given by 32 sample rays.

Additional Visualizations Figures S.2 and S.3 demonstrate qualitative comparisons of baselines' and our uncertainty estimation on test images of the Matterport Dataset. Notice that in the last example of Figure S.2, CF-NeRF's uncertainty does not respect the geometry of the scene, while Bayes' Ray and Ensemble mark the bed as uncertain despite it having low depth error. On the other hand, our method is able to correctly mark the bed as certain thanks to our derived uncertainty estimation based on classical photogrammetry. Moreover, even in challenging settings such as the ceiling of the first example in Figure S.3, and the closeup view of the closet in the second example of Figure S.3, our method is able to mark the regions with bad triangulation correctly (i.e., edge of the ceiling) using our formulation. On the other hand, approximated methods such as Bayes' Ray struggle to capture these small uncertainty variations, as Bayes' Ray simply marked the entire ceiling as uncertain.

Lastly, Figure S.1 shows additional visualizations of uncertainty estimations on the Scannet Dataset. Notice that compared to baselines, our method's uncertainty estimation can achieve better correlations with the depth error map (e.g., The carpet under the desk in the last example, and the tabletop in the second example).

Implementation Details To compute for the posterior likelihood shown in Eq. 13 in the main. Assuming that $\boldsymbol{x}$ is uniformly distributed within the scene bound Ω, and that the noise in 2D measurements are independent of each other given $\boldsymbol{x}$, we rederive and simplify Eq. 13 as

$$\begin{aligned}
&\mathbb{P}(\boldsymbol{x}|x_1, \dots, x_K) \\
&= \frac{\mathbb{P}(x_1, \dots, x_K|\boldsymbol{x})\,\mathbb{P}(\boldsymbol{x})}{\mathbb{P}(x_1, \dots, x_K)} \\
&= \frac{\mathbb{P}(x_1, \dots, x_K|\boldsymbol{x})\,\mathbb{P}(\boldsymbol{x})}{\mathbb{E}_{\boldsymbol{x}'\sim\mathbb{P}}\left[\mathbb{P}(x_1, \dots, x_K|\boldsymbol{x}')\right]}
\end{aligned}$$

Since $x'_j s$ are pairwise independent given $\boldsymbol{x}$,

$$\begin{aligned}
&= \frac{\mathbb{P}(\boldsymbol{x})\prod_{j=1}^{K}\mathbb{P}(x_j|\boldsymbol{x})}{\mathbb{E}_{\boldsymbol{x}'\sim\mathbb{P}}\left[\prod_{j=1}^{K}\mathbb{P}(x_j|\boldsymbol{x}')\right]} \\
&= \frac{\mathbb{P}(\boldsymbol{x})\prod_{j=1}^{K}\mathbb{P}(x_j|\boldsymbol{x})}{\int_\Omega \prod_{j=1}^{K}\mathbb{P}(x_j|\boldsymbol{x}')}\mathbb{P}(\boldsymbol{x}')\,d\boldsymbol{x}' \\
&= \frac{\prod_{j=1}^{K}\mathbb{P}(x_j|\boldsymbol{x})}{\int_\Omega \prod_{j=1}^{K}\mathbb{P}(x_j|\boldsymbol{x}')}\,d\boldsymbol{x}'.
\end{aligned} \tag{S.16}$$

The last step is because $\mathbb{P}(\boldsymbol{x}) = \mathbb{P}(\boldsymbol{x}') = \frac{1}{\text{Vol}(\Omega)}$. Since the numerator of the final expression in the above derivation is directly computable, we only need to compute the integral in the denominator. Because the computation of $\mathbb{P}(x_j|\boldsymbol{x}')$ involves a non-linear projection operation of $\boldsymbol{x}'$ to the image space, the integral is not expressible in analytical form. Thus, we instead use importance sampling over the intersection of the camera frustums to efficiently approximate the integral.

Specifically, let $\Pi_{j,\boldsymbol{x}}$ be the camera frustum given by the camera projection matrix Proj_j as defined in Sec. 5.1 in the main, and let $\Pi_{j,\boldsymbol{x},\delta}$ be the δ neighbor camera frustum given by only pixels within δ range of the center pixel. Then, if we assume that the 2D measurements are corrupted by a zero mean, σ^2 variance Gaussian noise, it is safe to assume that the probability $\mathbb{P}(x_j|\boldsymbol{x}') \approx 0$ for $\boldsymbol{x}' \notin \Pi_{j,\boldsymbol{x},5\sigma}$ because point outside would fall outside of 5 standard deviation of the Gaussian, and thereby has negligible probability. Thus, for the product $\prod_{j=1}^{K}\mathbb{P}(x_j|\boldsymbol{x}')$ to be nonzero, $\boldsymbol{x}'$ needs to be in the intersection of all 5δ neighbor frustums $\Pi_{j,\boldsymbol{x},5\sigma}$. Thus, let $\mathbb{Q}(\boldsymbol{x})$ be a proposal distribution that is uniform within the intersection of all $\Pi_{j,\boldsymbol{x},5\sigma} \forall j = 1, \dots, K$. Then we can rewrite the last expression in Eq. S.16 as

$$\begin{aligned}
&\mathbb{P}(\boldsymbol{x}|x_1, \dots, x_K) \\
&= \frac{\prod_{j=1}^{K}\mathbb{P}(x_j|\boldsymbol{x})}{\int_\Omega \prod_{j=1}^{K}\mathbb{P}(x_j|\boldsymbol{x}')}\,d\boldsymbol{x}' \\
&= \frac{\prod_{j=1}^{K}\mathbb{P}(x_j|\boldsymbol{x})}{\int_{\Omega_{5\delta}} \prod_{j=1}^{K}\mathbb{P}(x_j|\boldsymbol{x}')}\,d\boldsymbol{x}' \\
&= \frac{\prod_{j=1}^{K}\mathbb{P}(x_j|\boldsymbol{x})}{\text{Vol}(\Pi_{1:K,\boldsymbol{x},5\delta})\,\mathbb{E}_{\boldsymbol{x}'\sim\mathbb{Q}}\left[\prod_{j=1}^{K}\mathbb{P}(x_j|\boldsymbol{x}')\right]}
\end{aligned} \tag{S.17}$$

where

$$\Pi_{1:K,\boldsymbol{x},5\delta} = \bigcap_{j=1}^{K} \Pi_{j,\boldsymbol{x},5\sigma}.$$

Notice that $\Pi_{1:K,\boldsymbol{x},5\delta}$ is nonempty since $\boldsymbol{x} \in \Pi_{1:K,\boldsymbol{x},5\delta}$. The last expression in the final derivation can be efficiently and robustly approximated using Monte Carlo sampling because $\Pi_{1:K,\boldsymbol{x},5\delta}$ is the high-density region of $\mathbb{P}(x_j|\boldsymbol{x}')$. In practice, we set the noise's standard deviation to 2 and sample 10^6 points to compute the denominator in the final expression of Eq. S.17 for each $\boldsymbol{x}$. We set K to be 16 but only take the visible provenance samples defined by a predicted visibility of above 0.7. This means that each location $\boldsymbol{x}$ will have a variable number of associated projection cameras. Additionally, to speed up computation, we compute the posterior likelihood in the closed form if there is only one visible provenance sample from $\mathcal{D}_\theta(\boldsymbol{x})$.

Discussion on AUSE. We include a discussion on another metric reported in previous uncertainty estimation literature [15] – Area Under Sparsification Error (AUSE). In our work, we report the negative log likelihood (NLL) following other previous works [47, 48, 51] as we found AUSE to be not reflective of the uncertainty of the model in certain scenarios, which will be discussed in this section. AUSE was designed to measure the correlation between the uncertainty estimation and depth error. Concretely to compute for AUSE (see Figure S.4), the pixels in each test image are removed gradually ("sparsified") first i) according to their depth error (blue dotted curve) and then ii) according to their uncertainty estimation (red solid curve). Then the area between the two curves is the the resulting value for the AUSE metric (green region), where lower is considered better. Here, we bring up two discussions on how the AUSE metric may not reflect the methods' uncertainty estimation.

First, the score for AUSE is good if regions with high uncertainty imply that they have high depth error. However, an uncertain region implies that its *variance* is high, which *does not* necessarily imply that the region has large depth error. Such a region may have low depth error but with high variance, which will affect its AUSE score.

Second, a low AUSE number does not necessarily correspond to a good correlation between a method's uncertainty estimation and depth error. Following [15], Figure S.4 shows sparsification curves for Bayes' Ray (a) and our method (b) uncertainty estimation w.r.t. to depth mean absolute error (MAE). Notice that, despite (a) obtaining a lower AUSE score (0.16) compared to ours (0.3), the curve sparsified by Bayes' Rays uncertainty has a positive slope throughout the course of sparsification. This behavior implies that for this test view, Bayes' Ray (a) actually assigned higher uncertainty values to regions with a lower depth MAE error. On the other hand, our method's sparsification curve in (b) mostly show negative slopes throughout the course of pixel removal. This indicates that high uncertainty was roughly assigned to regions with higher depth MAE error and vice versa. This observation is consistent with the visualized uncertainty maps in the last row of Figure S.1. Notice that Bayes' Rays ((a) in Figure S.4 and the third column from the left in Figure S.1) assigned high uncertainty to the drawer and the back of the chair despite these regions having a low depth error as shown in the depth error mean below. On the other hand, our method ((b) in Figure S.4 and the fourth column from the left in Figure S.1) was able to correctly correlate the uncertainty estimation and the depth error despite it having a higher AUSE score than Bayes' Rays.

S.1. Criteria-based Viewpoint Optimization

Datasets & Baselines To evaluate this task, we captured 3 video sequences of various indoor objects using an iPhone X. For each video, the model is trained on 16, 16, and 22 sparse, wide-baseline views, respectively. We use the SCADE [56] backbone for this task. During testing, we refine the initial viewpoint of the target object by applying gradient descent to the sum of $\mathcal{L}_{\text{obj}}$ and $\mathcal{L}_{\text{select}}$. We evaluate the model on two different criteria: 1) the alignment of the camera's principal axis with the target object's normal and 2) the maximization of the target object's 2D projected area for an enhanced close-up perspective. The first criterion is measured via the dot product given by the camera's principal axis and the negative of the object's norm, and the second criterion is measured on the 2D projected area of the object. Because the object's ground truth area of projection is difficult to compute due to shape irregularities, we approximate the area using the area of the projected convex hull of the object.

Visualization of Optimization Trajectory Figure S.6 visualizes our rendering results along the optimization trajectory in area maximization, and Figure S.7 illustrates the optimization trajectory in aligning normals. Baseline methods achieve much worse intermediate rendering results than ours.

Visualization of Training Views In Figure S.5, we visualize the training views of the three scenes used for our viewpoint optimization.

Implementation Details To optimize the camera pose given the target objective, we parameterize the camera rotation in $SO(3)$ manifold and run gradient descent on the tangent space using LieTorch [54]. We use stochastic gradient descent (SGD) with momentum 0.9. The learning rate for both rotation and translation is set to 0.01. We take 10 steps of optimization starting from the initial view. In the area maximization experiment, we normalize the area to lie within $[0, 1]$ by dividing the image width/height, to make the optimization more stable.

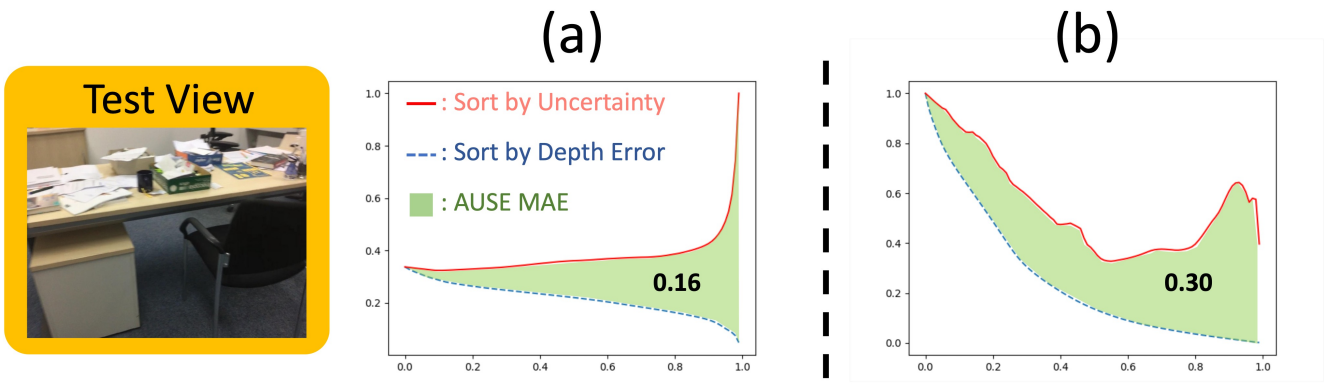

Figure S.4. **Visualization of Sparsification Curves of depth MAE for Bayes' Ray (a) and Ours (b) for the test view on the left**. Notice that despite (a) assigning higher uncertainty to regions with a lower depth error as indicated by its red curve with a positive slope, it incurred a lower AUSE score than method (b)'s curve, which shows a rough correlation w.r.t. depth MAE as shown by its red curve with mostly a negative slope. For visualizations of their uncertainty estimation vis-á-vis depth error, see the last row of Figure S.1.

	PSNR (↑)	SSIM (↑)	LPIPS (↓)
DDP [44]	19.18	0.651	0.361
SCADE [56]	19.83	0.664	0.347
Ours	**20.09**	**0.665**	**0.345**

Table S.1. **Novel View Synthesis Results on Tanks and Temples Dataset.** SCADE numbers are obtained by running their released model weights. Scores for Vanilla NeRF and DDP are copied from SCADE.

S.1. Novel View Synthesis

Datasets & Baselines Following previous works [44, 50, 56], we evaluate on a subset of ScanNet [10] released by DDP [44] comprising of three scenes each with $17-20$ training and 8 test views. For quantitative evaluation, we report the standard metrics PSNR, SSIM [60], LPIPS [68] on the novel test views. We compare with recent NeRF-based works including state-of-the-art methods DDP [44], SCADE [56] and DäRF [50] on the sparse, unconstrained setting. The scores reported are directly copied from SCADE [56] and DäRF [50].

Additional Results on Tanks and Temples We further conduct experiments on the Tanks and Temples dataset (three large indoor scenes – Courtroom, Church, and Auditorium) released by SCADE [56] to test the robustness of our novel view synthesis application, as detailed in Sec. 5.3 in the main. We use the same test split as SCADE which consists of 21. 26, and 21 training views and 8 test views for the Church, Courtroom, and Auditorium respectively. Similar to experiments on Scannet, we start from the released SCADE model and finetune it for an additional 200K iterations with our extra objective $\mathcal{L}_{\text{ProvNVS}}$ (see Eq. 15 main paper). Table S.1 shows that our method surpasses the baselines across all metrics. Moreover, Figure S.9 shows qualitative comparisons of our method and SCADE. Notice that with $\mathcal{L}_{\text{ProvNVS}}$ enabled by our provenance stochastic process, we are able to clear out large chunks of clouds as shown in the first and third columns of the visualizations.

Additional Visualization on Scannet We show additional visualizations on the effect of $\mathcal{L}_{\text{ProvNVS}}$ on the Scannet Dataset. Notice that with our objective, we can clear out artificial geometries as shown in both the RGB and depth visualizations (second and fourth columns).

Implementation Details We take the released pre-trained SCADE [56] model on Scannet and finetune the model with our proposed objective $\mathcal{L}_{\text{ProvNVS}}$ (Eq. 15 main paper) for another 200K iterations. The provenance loss is used together with the photometric loss and the space carving loss. Thus, in total each training iteration the following objective is used for NeRF's optimization:

$$\mathcal{L}_{\text{photometric}} + \beta_1 \mathcal{L}_{\text{space carving}} + \beta_2 \mathcal{L}_{\text{ProvNVS}}. \quad \text{(S.18)}$$

We set $\beta_1 = 0.005$ for all scenes and $\beta_2 = 0.001$ for scene 710 and 781, and $\beta_2 = 0.01$ for scene 758. We set the learning rate to be $5e{-}5$ for all of the Scannet scenes and we only enforce the provenance loss for points with rendered transmittance greater than 0.7 and for provenance samples with visibility greater than 0.9. Lastly, we set the margin for the provenance loss as $\alpha = 0.05$.

S.1. Ablation Study

Metrics Detail In the Ablation section refsec˙ablation of main, we computed the Average Precision (AP) and Area Under the Curve (AUC) scores with the confidence score given by the Euclidean distance between the predicted sample and ground truth observation. Here we detail our metric computation. We follow the usual definition of AP and

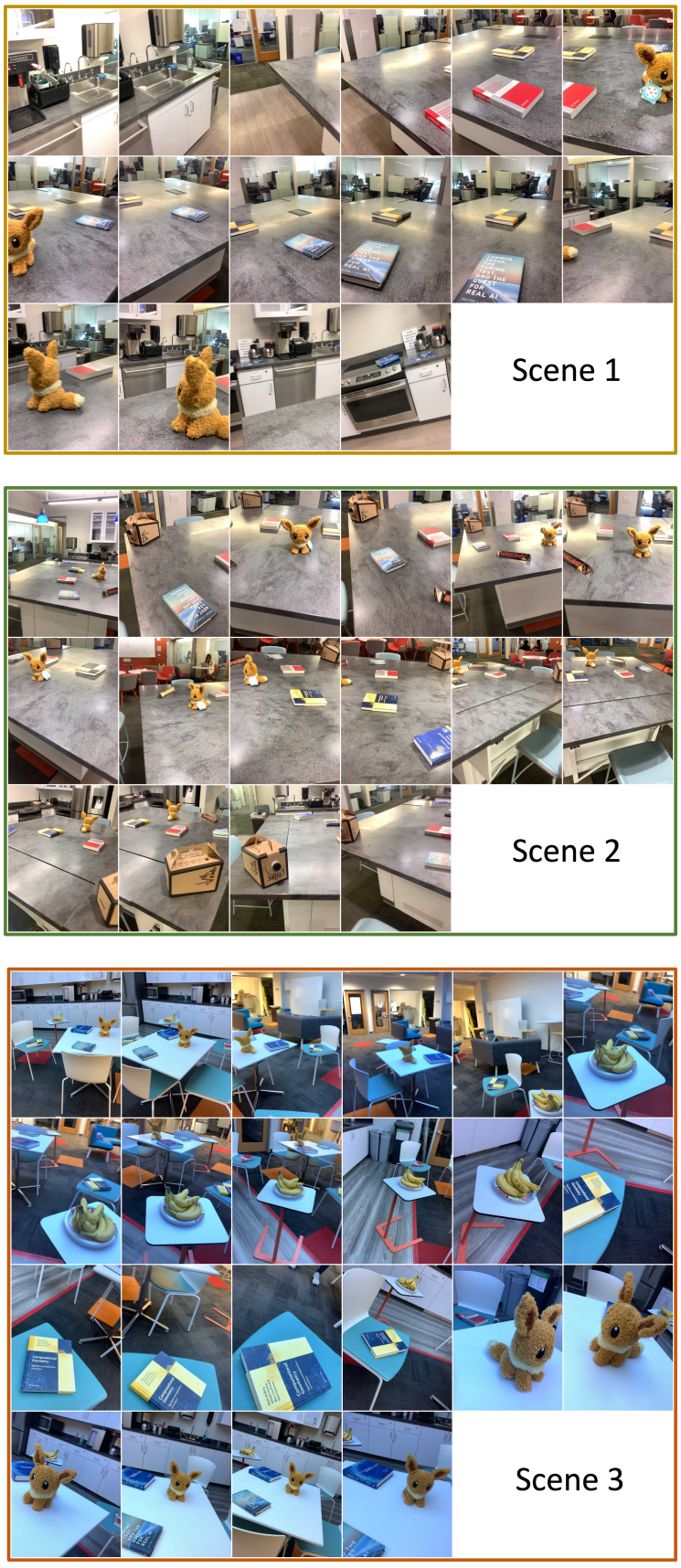

Figure S.5. **Visualization Training Views used for viewpoint optimization**.

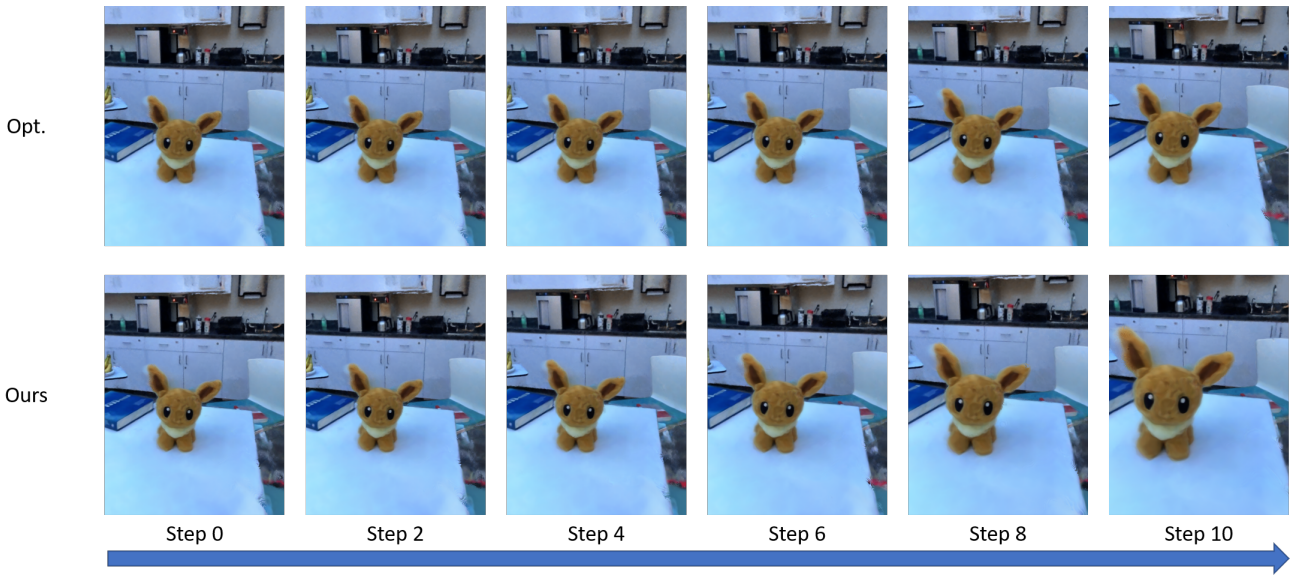

Figure S.6. Rendering results along the trajectory in area maximization between ours and naive optimization.

Figure S.7. Rendering results along the trajectory in normal alignment between ours and naive optimization.

AUC. Given a set of confidence thresholds $\delta_0, \ldots, \delta_{N-1}$, we define

$$\text{AP} = \sum_{k=0}^{N-1} \left[\text{Recall}\left(k\right) - \text{Recall}\left(k+1\right)\right] \cdot \text{Precision}\left(k\right) \tag{S.19}$$

where $\text{Recall}\left(N\right) = 0$ and $\text{Precision}\left(N\right) = 1$. On the other hand, AUC is defined as the area under the Precision and Recall curve of the set of thresholds. Thus, to compute for AP and AUC, we need to define how do we measure recall and precision in our setting. To do so, for 3D point $\boldsymbol{x}$, we use the set of provenances $\text{Prov}\left(\boldsymbol{x}\right) \setminus \{(0, \mathbf{0})\}$ as defined in Eq. 4 [13] of the main as the ground truth labels, and we sample $K = 128$ provenances from the learned provenance distribution $\mathcal{D}_\theta\left(\boldsymbol{x}\right)$ and only take the samples with visibility above 0.9. If we define a discrete uniform distribution over $\widehat{\text{Prov}}(\boldsymbol{x})$ as $\mathcal{U}\left(\boldsymbol{x}\right)$, for a particular confidence threshold δ, the precision and recall at $\boldsymbol{x}$ is defined as

- **Precision** $\left(\boldsymbol{x}\right)$ =

$$\mathbb{E}_{(\hat{\boldsymbol{d}}_i, \hat{t}_i) \sim \mathcal{U}(\boldsymbol{x})} \left[\mathbb{1}_{\left\{ \min_j \left\| \left((\boldsymbol{d}_j, t_j) - (\hat{\boldsymbol{d}}_i, \hat{t}_i) \right) \right\|_2^2 < \delta \right\}} \right] \tag{S.20}$$

- **Recall** $\left(\boldsymbol{x}\right)$ =

[13] The visibility term is computed based on the ground truth geometry.

Figure S.8. **Additional Novel View Synthesis Comparisons on Tanks and Temples**.

$$\frac{1}{K}\sum_{i=1}^{K}\mathbb{1}_{\left\{\min_{\widehat{\text{Prov}}(\boldsymbol{x})}\left\|\left((\boldsymbol{d}_j,t_j)-(\hat{\boldsymbol{d}}_i,\hat{t}_i)\right)\right\|_2^2<\delta\right\}}, \tag{S.21}$$

We note that **Precision**$(\boldsymbol{x})$ measures whether all the ground truth observations are covered by the predicted observations, while **Recall**$(\boldsymbol{x})$ measures if the predicted observations are close to one of the ground truth observations. We use a set of 500 thresholds log-linearly interpolated from e^{-20} to 1 for accurate and robust AP and AUC computation.

K	AP (↑)	AUC (↑)
1	0.197	0.199
2	0.534	0.536
4	0.644	0.646
8	0.738	0.740
16	**0.745**	**0.747**

Table S.2. **Ablation Results on Scannet.**

Further Ablation Study We further ablate on the number of random function samples K to take during each time of resampling. Tab. S.2 shows quantitative AP and AUC results on the Scannet dataset. As shown in the table, the quality of the provenance samples improves as we increase the number of latent random function samples. This is because a larger number of sample pools can help the deep transformation better transform the distribution from latent space to the space of provenances. We note that the improvement gradually becomes marginal as K increases beyond 8. Thus, we use $K = 16$ for all of our experiments.

Implementation of Deterministic Field Deterministic Field (see entry "Deterministic Field" in Tab. 3 main paper) is implemented using the same architecture as the provenance branch of **ProvNeRF** (i.e., 3 layers of MLP). Input the deterministic field is set to the positionally encoded 3D location and the output is a direction and distance tuple, similarly parameterized as **ProvNeRF** (For detail of the parameterization, please refer to Sec. S.1). The deterministic field is trained with the mean square distance of the predicted provenance samples at $\boldsymbol{x}$ with an empirical sample from $\hat{\mathcal{D}}(\boldsymbol{x})$. We train the deterministic field for 200K iterations using the same hyperparameters as our method.

Implementation of Gaussian-Based Models The Gaussian-based models (see Gaussian-based w/ C=2, C=5 entries in Tab. 3 main paper) are parameterized using a 3-layer MLP that takes in a 3D location and outputs the means, variances, and weights of each Gaussian. The Gaussian models are trained with the negative log-likelihood of the empirical provenance samples from $\hat{\mathcal{D}}(\boldsymbol{x})$ under the parameterized Gaussian mixture. The models are trained for 200K iterations with the same set of hyperparameters as **ProvNeRF**.

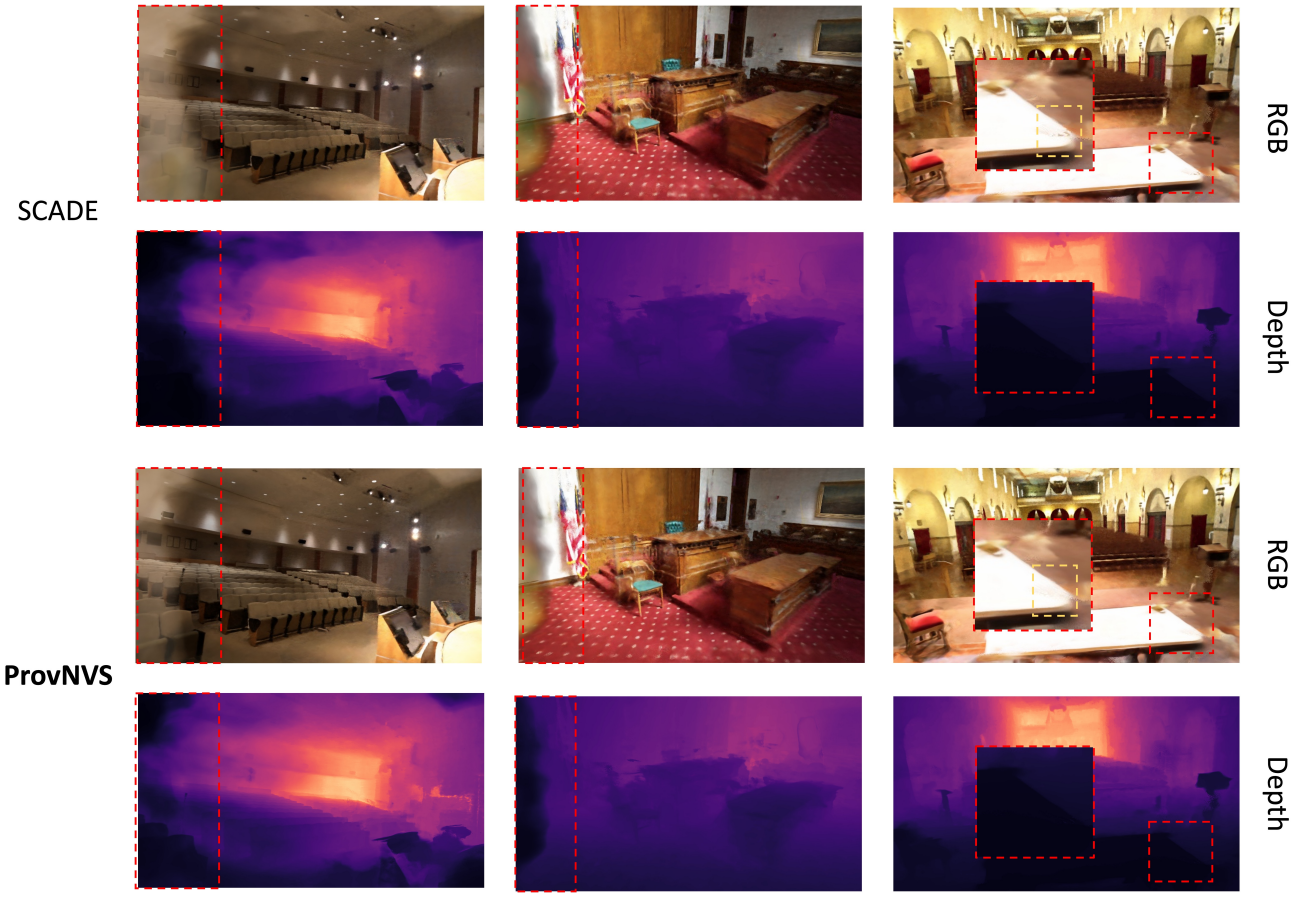

Figure S.9. **Additional Novel View Synthesis Comparisons on Tanks and Temple dataset**

S.1. Derivation of Objective

In Sec. 4.3 of the main paper, we showed that the fIMLE objective in Eq. 8 of the main paper is equivalent to a pointwise matching loss using the theory of calculus of variation and Euler-Lagrange Equation. We outline the details here.

Specifically, assuming $\boldsymbol{D}_\theta^{(j)}$ is differentiable for all $j = 1, \ldots, K$, we show that any minimum for the L2 norm difference in Eq. 9 of the main paper is only achieved when $\boldsymbol{D}_i$ and $\boldsymbol{D}_\theta^{(j)}$ are pointwise equivalent. Fix M empirical function samples $\hat{\boldsymbol{D}}_1, \ldots, \hat{\boldsymbol{D}}_M$ from $\hat{\mathcal{D}}$, and K i.i.d. sample functions $\boldsymbol{D}_\theta^{(1)}, \ldots, \boldsymbol{D}_\theta^{K}$ from the distribution $\mathcal{D}_\theta$. Then, we define a functional J of K variables in $\boldsymbol{D}_\theta^{(1)}, \ldots, \boldsymbol{D}_\theta^{K}$ in the form of

$$J\left(\boldsymbol{D}_\theta^{(1)}, \ldots, \boldsymbol{D}_\theta^{(K)}\right) = \frac{1}{2}\sum_{i=1}^{M}\left\|\hat{\boldsymbol{D}}_i - \boldsymbol{D}_\theta^{(j_i)}\right\|_{L^2}^2 \quad \text{(S.22)}$$

where

$$j_i = \arg\min_{j=1,\ldots,K}\left\|\boldsymbol{D}_i - \boldsymbol{D}_\theta^{(j)}\right\|_{L^2}^2$$

for each $i = 1, \ldots, M$ denote the argmin of $\left\|\boldsymbol{D}_i - \boldsymbol{D}_\theta^{(j)}\right\|_{L^2}^2$ over $j \in \{1, \ldots, K\}$. Then, by the Euler-Lagrange Equation, we know that if the tuple $\left(\boldsymbol{D}_{\theta^*}^{(1)}, \ldots, \boldsymbol{D}_{\theta^*}^{(K)}\right)$ is a minimizer to J, they must satisfy

$$\frac{\partial}{\partial \boldsymbol{D}_\theta^{(j)}} J\left(\boldsymbol{D}_{\theta^*}^{(1)}(\boldsymbol{x}), \ldots, \boldsymbol{D}_{\theta^*}^{(K)}(\boldsymbol{x})\right) = 0 \quad \text{(S.23)}$$

for all $\boldsymbol{x} \in \mathbb{R}^3$ and $j = 1, \ldots, K$. On the other hand, taking the functional derivative with respect to each of the K variables of J, we derive that at point $\boldsymbol{x} \in \mathbb{R}^3$,

$$\frac{\partial}{\partial \boldsymbol{D}_\theta^{(k)}} J\left(\boldsymbol{D}_\theta^{(1)}(\boldsymbol{x}), \ldots, \boldsymbol{D}_\theta^{(K)}(\boldsymbol{x})\right) = \begin{cases} 0 & \text{if } k \notin \{j_1, \ldots, j_M\} \\ \left\|\hat{\boldsymbol{D}}_i(\boldsymbol{x}) - \boldsymbol{D}_\theta^{(k)}(\boldsymbol{x})\right\|_2 & \text{if } k = j_i \end{cases}. \quad \text{(S.24)}$$

Then, combining Eqs S.22 and S.24, we see that objective 8 in the main only can achieve its minimum when

$$\left\|\left(\hat{\boldsymbol{D}}_i - \boldsymbol{D}_\theta^{(j_i)}\right)(\boldsymbol{x})\right\|_2 = 0, \ \forall \boldsymbol{x} \in \mathbb{R}^3. \quad \text{(S.25)}$$

The above derivation shows that minimizing the L^2 differences of functions is equivalent to minimizing the norm differences of the functions at all points. This shows the equivalence of the fIMLE objective in Eq. 8 and **ProvNeRF** ob-

jective Eq. 10 in the main.

S.1. Limitation & Future Works

We note that our work is not without limitations. Our **ProvNeRF** requires post-hoc optimization which takes around 8 hours limiting its current usability for real-time or on-demand applications. The idea presented in our work is however not specific to the model design and it can be adapted to faster coordinate-based representations such as hash grids [38] that allow it for real-time capabilities. We leave this exploration as future work. Moreover, we also note that the hyperparameters to incorporate **ProvNeRF** are chosen for better performance, e.g. for the uncertainty and novel view synthesis applications, and in the future, it will be beneficial to explore a more adaptive approach in integrating provenance to different downstream applications.