CLONeR: **C**amera-**L**idar Fusion for **O**ccupancy Grid-aided **Ne**ural **R**epresentations

Alexandra Carlson*,[1], Manikandasriram Srinivasan Ramanagopal*,[2],
Nathan Tseng[1], Matthew Johnson-Roberson[3], Ram Vasudevan[2] and Katherine A. Skinner[2]

***Abstract*— Recent advances in neural radiance fields (NeRFs) achieve state-of-the-art novel view synthesis and facilitate dense estimation of scene properties. However, NeRFs often fail for large, unbounded scenes that are captured under very sparse views with the scene content concentrated far away from the camera, as is typical for field robotics applications. In particular, NeRF-style algorithms perform poorly: (1) when there are insufficient views with little pose diversity, (2) when scenes contain saturation and shadows, and (3) when finely sampling large unbounded scenes with fine structures becomes computationally intensive. This paper proposes *CLONeR*, which significantly improves upon NeRF by allowing it to model large outdoor driving scenes that are observed from sparse input sensor views. This is achieved by decoupling occupancy and color learning within the NeRF framework into separate Multi-Layer Perceptrons (MLPs) trained using LiDAR and camera data, respectively. In addition, this paper proposes a novel method to build differentiable 3D Occupancy Grid Maps (OGM) alongside the NeRF model, and leverage this occupancy grid for improved sampling of points along a ray for volumetric rendering in metric space. Through extensive quantitative and qualitative experiments on scenes from the KITTI dataset, this paper demonstrates that the proposed method outperforms state-of-the-art NeRF models on both novel view synthesis and dense depth prediction tasks when trained on sparse input data.**

I. Introduction

The estimation of dense scene properties from sparse and low diversity sensor data is a critical task in robot perception. In particular, for autonomous vehicle perception, estimating dense and accurate depth is essential for scene understanding used for downstream planning and navigation tasks. Neural radiance fields (NeRFs) are an attractive solution for producing joint dense depth and color maps of a scene [1] . NeRFs use implicit functions to encode volumetric density and color observations from multiple images with known poses. Recent advances in NeRFs have demonstrated state-of-the-art performance for novel view synthesis and 3D modeling.

Still, outdoor urban scenes captured in autonomous vehicle datasets remain challenging for NeRFs. These scenes are typically captured from sparse viewpoints with limited view diversity, i.e., a camera mounted on a vehicle moving forward. Outdoor driving scenes tend to be large and unbounded, with much of the scene content concentrated far away from the camera. Lastly, images captured outdoors feature varying exposure and brightness levels depending upon both the camera settings and weather conditions. These challenges can lead to learning inaccurate scene geometry, blurry scene renderings and poorly modeled fine/thin structures, especially at long distances.

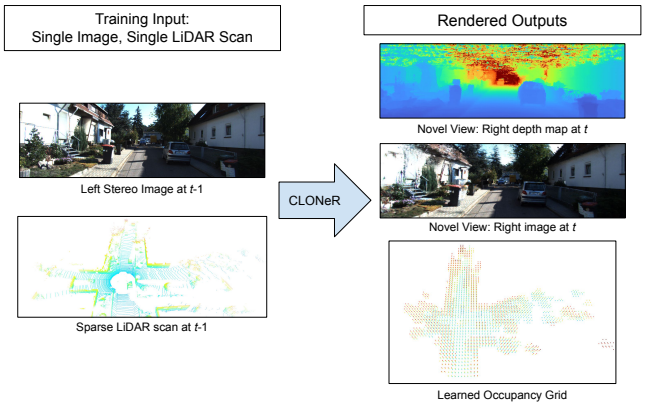

Fig. 1: An illustration of CLONeR method proposed in this paper.

This paper presents ***CLONeR***, **C**amera-**L**idar fusion for **O**ccupancy grid-aided **Ne**ural **R**epresentation, a novel NeRF framework that addresses key challenges to applying NeRFs to outdoor driving scenes. The proposed method can accurately *clone* a 3D scene from a sparse set of image views and LiDAR scans. Impressively, CLONeR can render novel views and dense depth maps with fine detail from a single image and LiDAR scan, see Fig. 1 for an example. CLONeR accurately models the operation of both LiDAR and RGB camera sensors within a NeRF framework to learn a dense 3D representation of a scene. In addition, CLONeR simultaneously builds a differentiable 3D Occupancy Grid Map (OGM) using LiDAR data, which it leverages to efficiently sample large outdoor scenes in metric space. This learned occupancy grid map is capable of maintaining known versus unknown regions in a probabilistic manner, and replaces the standard coarse MLP sampling scheme used in NeRFs. Critically, CLONeR decouples occupancy and color learning within the NeRF by utilizing two separate MLPs. One of these MLPs learns occupancy from LiDAR ray data, and the other MLP learns the RGB model of the scene from camera rays. By doing this, sensor artifacts and illumination effects in the RGB images do not negatively influence the learned scene depth.

In summary, the key contributions of the proposed work, CLONeR, are:

*denotes equal contribution

This work was supported by a grant from Ford Motor Company via the Ford-UM Alliance under award N028603.

[1]A. Carlson and N. Tseng are with Ford Motor Company, but completed this work during graduate studies at the Department of Robotics at University of Michigan askc,tsnathan@umich.edu

[2]M. Srinivasan Ramanagopal, R. Vasudevan and K. Skinner are with the Department of Robotics at University of Michigan srmani,ramv,kskin@umich.edu

[3]M. Johnson-Roberson is with the Robotics Institute at Carnegie Mellon University mkj@andrew.cmu.edu

- A decoupled NeRF model where LiDAR is used to supervise geometry learning while camera images are used to supervise color learning. This allows one to perform novel view synthesis using as few as a single image when lidar scans are available.
- A differentiable 3D occupancy grid learned directly on the GPU alongside the decoupled NeRF model. This 3D occupancy grid explicitly maintains information about known and unknown regions, and outperforms standard coarse NeRF based importance sampling methods.

We demonstrate that our method outperforms state-of-the-art NeRF models on both novel view synthesis and depth prediction tasks through extensive quantitative and qualitative experiments on several scenes from the KITTI dataset.

The remainder of this paper is organized as follows: We review related work in Section II, and give a detailed description of the proposed method, CLONeR, in Section III. We present extensive quantitative and qualitative results in Section IV. Section V provides the conclusion and future work.

II. Related Works

The impressive performance achieved by the original NeRF [1] has inspired many subsequent works to extend it to various applications including generative modeling [2], dynamic scene rendering [3] and scene relighting [4]. In particular, NeRFs have demonstrated success for indoor scenes [5] and unbounded scenes [6], [7], [8], [9]. In this work, we develop a novel NeRF framework that leverages camera and LiDAR to enable depth-guided learning for unbounded, outdoor scenes observed from limited viewpoints.

A. Depth-guided Learning in Neural Radiance Fields

In prior work, NeRF-based models have been used along with depth supervision from structure-from-motion techniques [10], RGB-D cameras [5], [11], [12] and continuous-wave Time-of-Flight sensors [13]. However, RGB-D cameras do not operate well outdoors in the presence of sunlight, and SfM pipelines operate poorly in low-texture environments, making this depth supervision very difficult to obtain for outdoor scenes that would be encountered in autonomous driving applications.

B. Neural radiance fields for outdoor scenes

Outdoor scenes still present a unique challenge for NeRFs. Many outdoor scenes have a relatively large scale and are unbounded. Much of the scene content is concentrated far away from the camera(s) with a significant amount of empty space. The scenes are large enough that sampling the 3D scene volume to learn accurate scene structures becomes computationally challenging [12]. As a result, NeRFs struggle to learn even coarse scene occupancy. This results in blurry scene renderings and poorly modeled fine/thin structures, especially at long distances [7], [6].

Very few NeRF models have tried to use LiDAR information to learn scene occupancy in outdoor scenes. Of particular relevance to our work is Urban Radiance Fields [9] and Neural Point Light Fields [14]. The method proposed in Urban Radiance fields utilizes LiDAR data along with images from panoramic cameras captured in outdoor scenes.Their data collection trajectory specifically ensures a large number of views are available with diversity in the camera poses in addition to their LiDAR data. In comparison, the proposed method is designed to operate on very sparse camera views and LiDAR data [9].

In Neural Point Light Fields, they represent outdoor scenes with a light field learned on top of a sparse point cloud. They aggregate point features along each ray to get a final RGB prediction. However, this method learns a feature vector (via CNN encoding) for each point in the point cloud, and thus requires long driving trajectories/videos to learn a sufficient scene representation [14]. In contrast, our proposed method can produce accurate scene models from as few as a single image paired with a single point cloud. In summary, neither [9], [14] have demonstrated the ability to learn dense depth using just LiDAR returns and sparse views.

C. Sampling in Neural Radiance Fields

A key challenge in the original NeRF formulation is the need to finely sample along a ray to approximate the line integral. Mip-NeRF [15] addressed the sampling and aliasing challenges in the original NeRF formulation and Mip-NeRF 360 [6] extended it to address unbounded scenes with focus on a central object. Block-NeRF [16] addresses this challenge with training of multiple NeRF-like models to large city scale scenes. TermiNeRF [17] and DO-NeRF [12] discretize a ray into bins and employ a separate sampling MLP that estimates the probability of each bin being occupied. However, in order to train the aforementioned networks, the input data still needs to cover a diverse set of views.

Recent developments have popularized the use of occupancy grids for sampling during test-time in NeRFs [18], [19]. Most similar to ours, in [20], the authors propose to maintain a cascaded set of occupancy grids in order to skip ray marching on empty space. They present heuristic methods to update the occupancy grids as training of the NeRF model progresses. In contrast, we propose to jointly perform occupancy grid mapping while training the NeRF model. Furthermore, we distinguish our approach by leveraging LiDAR measurements to learn and construct these occupancy grids in a probabilistic manner.

III. Methods

Figure 2 provides an overview of the proposed method. Its two key components are (i) the differentiable occupancy grid map (OGM), which is used to learn the coarse occupancy of the 3D metric space of the scene, and (ii) the decoupled NeRF MLPs, which independently learn a fine color and depth model of the scene. We describe each model component and the training process of the framework in the following subsections.

A. Overview of Neural Radiance Fields

A Neural Radiance Field (NeRF) [1] models the scene using a 5D vector-valued function using an MLP network. This MLP takes a 3D location $\vec{x} = (x, y, z)$ and a viewing direction $\vec{d} = (x_d, y_d, z_d)$ as inputs and outputs the color $\vec{c} = (r, g, b)$ emitted by that scene point and a scalar volume density σ.

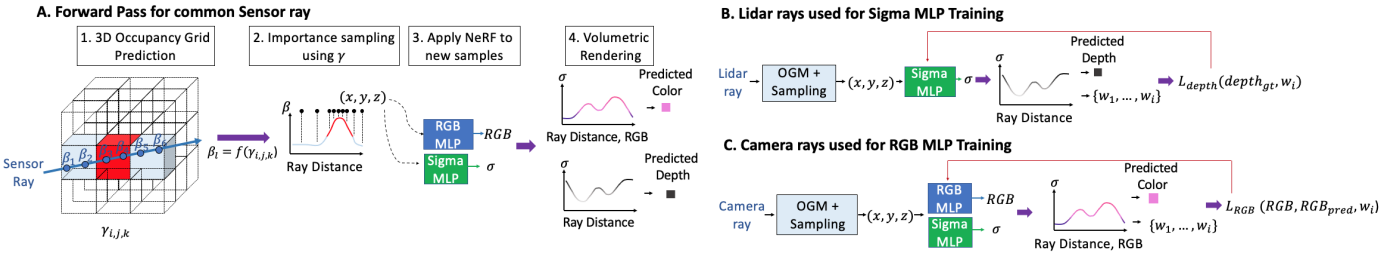

Fig. 2: Overview of proposed CLONeR framework. Panel A shows the forward pass of the framework. In Panel B, LiDAR data is used as the only supervision for the Sigma MLP during training. In Panel C, camera data is the only supervision used to train the Color MLP.

An image pixel (u, v) is mapped to a camera ray $\vec{r}(t) = \vec{o} + t\vec{d}$ using a simple pinhole camera model. The expected color of the camera ray is computed using the quadrature approximation of the volumetric rendering equation between the near and far bounds t_n and t_f [1]:

$$\hat{C}(\vec{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i\delta_i))\vec{c}_i, \quad \text{where} \quad T_i = \exp\big(-\sum_{j=1}^{i-1}\sigma_j\delta_j\big), \tag{1}$$

and $\delta_i = (t_{i+1} - t_i)$ is the distance between samples along the ray where the MLP is applied. The weights of the MLP are then optimized by minimizing the mean squared error between the rendered color $\hat{C}(\vec{r})$ of a pixel and the ground truth color of that pixel.

The termination depth of a ray $\hat{D}(\vec{r})$ is computed as:

$$\hat{D}(\vec{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i\delta_i))t_i. \tag{2}$$

B. CLONeR World Coordinate System

Given the set of camera and LiDAR poses from an external source such as a GNSS system, we follow [1] and subtract the average translation and normalize the rotation such that the local $z-$axis for each camera points into the camera. The sensor poses and the point clouds from the LiDAR are further scaled by a common factor such that the region of interest falls completely within $[-1, 1]^3$, as required by most types of positional encodings used in NeRFs [1]. We refer to this scaled, bounded domain as the *world cube*.

For generic outdoor driving scenes, we determine the region of interest in the following way: for each input camera pose, we use its intrinsic parameters and user-defined near and far values to compute the 8 corners of its view frustum in world coordinates. We choose a set of desired camera poses along with its intrinsics and compute corresponding view frustum points. We then compute a common scale factor such that all the sensor poses in addition to all the above points fall within the world cube. We explicitly avoid using the LiDAR point clouds to define the scale factor as LiDAR data could contain regions never seen in any of the images, such as points behind the camera.

C. Integrating LiDAR Measurements into NeRF

A 3D scanning LiDAR continuously sweeps the scene using a set of laser beams. Let H_L denote the pose of the LiDAR in world coordinates (as a 3×4 matrix) and let $\vec{x}$ denote a 3D point measured in the local sensor frame at some time t. This corresponds to a ray in the world coordinates with ray origin $\vec{o} = H_L[0, 0, 0, 1]^T$ and ray direction $\vec{d} = H_L[\vec{x}^T 1]^T - \vec{o}$. Recall from Sec. III-A that one can render a camera ray using the NeRF MLP and obtain a termination depth, expected color, as well as opacity at samples along the ray. Similarly, based upon how we model the LiDAR ray, we can also render these quantities for each LiDAR ray cast into the scene.

D. Decoupled NeRF Model

We propose a decoupled NeRF model that contains two separate MLP networks, one for learning the scene geometry and one for scene color: $F^g_\Theta : \mathrm{R}^3 \rightarrow \mathrm{R}$, which we call the sigma MLP, and $F^c_\Phi : \mathrm{R}^5 \rightarrow \mathrm{R}^{n_c}$,which we call the color MLP. Note that n_c is the number of color channels. These decoupled MLP networks are trained simultaneously such that $\sigma_i = F^g_\Theta(\vec{r}(t_i))$ and $\vec{c}_i = F^c_\Phi(\vec{r}(t_i), \vec{d})$.

These two networks are shown in panels B and C of Fig. 2. Our sigma MLP takes the positionally encoded 3D positions only as input. It has 1 hidden layer of 64 neurons with the estimated differential opacity σ as output. Our color MLP takes the positionally encoded 3D positions and encoded view directions as input. It has 2 hidden layers of 64 neurons each and outputs colors with n_c channels. We use multi-resolution hash tables [20] for positional encoding and use Spherical Harmonics [18] for encoding view direction. We use separate hash tables for positional encoding of the sigma and color MLPs. Unlike the original NeRF, there is no interaction between the parameters of the sigma MLP and the color MLP.

When rendering a LiDAR ray, we only evaluate the sigma MLP while computing gradients. When rendering a camera ray, we evaluate the sigma MLP, without computing gradients, to estimate the differential opacity needed in Eq. 1 to compute the expected color. In contrast to the original NeRF model, which uses a simple pinhole camera model, we instead use the full pinhole camera model to compute ray directions. This is needed to maintain consistency with camera-LiDAR calibration parameters. Additionally, we use multiple samples per pixel by generating rays corresponding to fractional pixel coordinates. In order to obtain ground truth color information for the fractional pixels, we use bilinear interpolation of the input images.

E. Loss Functions for NeRF MLPs

We use separate loss functions for the LiDAR rays and camera rays.

a) LiDAR Line-of-sight (LOS) loss: We use a variant of the line-of-sight loss used in [9]. For a given LiDAR ray, let z^* be the ground truth termination depth along the ray. Let t_i be the samples along the ray and w_i be the associated accumulated opacity at those samples. We define $w_i^* = \mathcal{K}_\epsilon(t_i - z^*)$ where $\mathcal{K}_\epsilon$ is a truncated Gaussian distribution with variance equal to $(\epsilon/3)^2$ and ϵ is a non-zero number that is decayed as training progresses. We can now define the line-of-sight loss as

$$\mathcal{L}_{\texttt{sight}}(\Theta) = \|w_i - w_i^*\|_1 \tag{3}$$

Ideally, we would like w_i to be non-zero only at a single point and zero everywhere else. To encourage this sparse behavior, we use an L1 penalty.

b) LiDAR Opacity loss: The volumetric rendering equation as implemented in the original NeRF model only requires that the sum of accumulated opacity does not exceed 1. However, Eq. 2 implicitly assumes that the accumulated opacities sum to 1. Therefore, we use the penalty method to force the accumulated opacity for a ray to sum to 1 by using the following opacity loss:

$$\mathcal{L}_{\texttt{opacity}_1}(\Theta) = \|1 - \sum_i w_i\| \tag{4}$$

where Θ is the trainable parameters of the sigma MLP.

c) Camera Color loss: Following [5], we use L1 penalty for supervising camera rays.

$$\mathcal{L}_{\texttt{color}}(\Phi) = \|\hat{C}(\vec{r}) - C(\vec{r})\|_1 \tag{5}$$

d) Camera Opacity loss: The camera opacity loss is defined similarly to the opacity loss for LiDAR, except that it operates on Φ, the parameters of the color MLP:

$$\mathcal{L}_{\texttt{opacity}_c}(\Phi) = \|1 - \sum_i w_i\| \tag{6}$$

e) Total Loss: The complete loss function is given by

$$\mathcal{L}(\Theta, \Phi) = \lambda_1 \mathcal{L}_{\texttt{sight}}(\Theta) + \mathcal{L}_{\texttt{opacity}_1}(\Theta) + \lambda_2 \mathcal{L}_{\texttt{color}}(\Phi) + \mathcal{L}_{\texttt{opacity}_c}(\Phi)$$

where $\lambda_1 = 1e_3$ and $\lambda_2 = 0.1$, and only λ_1 is decayed over training iterations.

F. Occupancy Grid Mapping

Occupancy Grid Mapping (OGM) has long been used in robotics to model 3D space in a probabilistic manner [21]. In this paper, we cast OGM as an optimization problem that can be solved using stochastic gradient descent. We then use the learned OGM to efficiently sample along sensor rays, taking the place of a coarse MLP in standard NeRF algorithms.

Let $\mathcal{O} \in R^{N \times N \times N}$ denote a regular 3D grid partitioning a bounded region of interest into voxels. If p_k refers to the probability that a single grid element is occupied, then the corresponding log-odds is defined as $l_k = \log \frac{p_k}{1-p_k}$. In a typical OGM algorithm, at each iteration n, given a range measurement z_n, each grid element is updated as follows:

$$l_{n,k} = l_{n-1,k} + \mathcal{S}(k, z_n) - l_0 \tag{7}$$

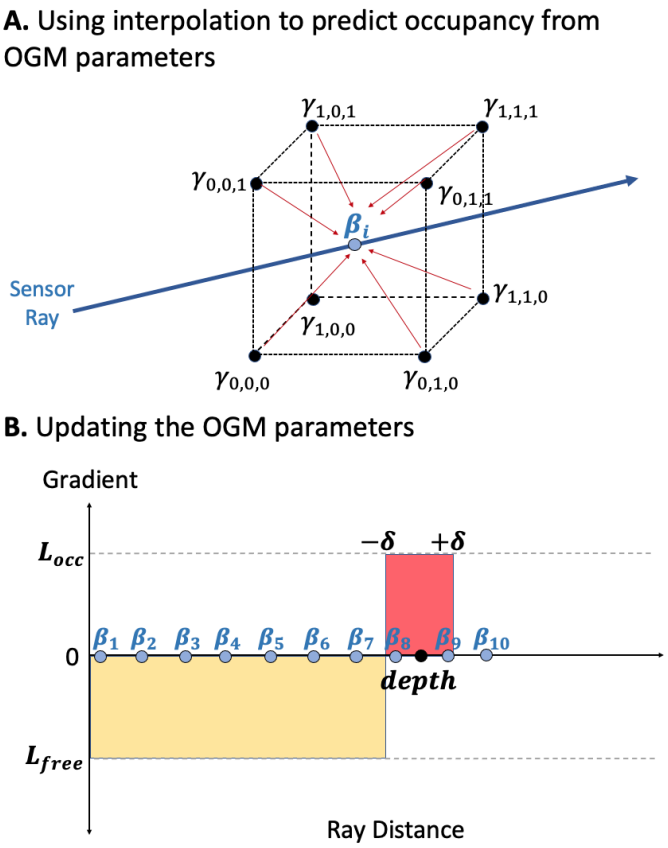

Fig. 3: Occupancy Grid Mapping as Stochastic Gradient Descent. Panel A describes the interpolation process to predict occupancy values at each ray sample from the neighboring γ variables in the OGM. Panel B describes how the γ variables of the OGM are updated using stochastic gradient descent.

where l_0 is the prior of occupancy represented in log-odds, $\mathcal{S}$ is the inverse sensor model that returns the measurement update for grid element k given z_n and the known pose of the sensor [21]. We propose to treat the log-odds variables l_k as learnable parameters $\gamma_k \in \mathcal{O}$ initialized to l_0. We can then re-write the map update equation Eq. (7) from OGM to match a parameter update equation from SGD as follows:

$$\gamma_k := \gamma_k - \alpha \nabla \mathcal{L}(\mathcal{O}, z_n) \tag{8}$$

where α is the learning rate and $\mathcal{L}$ is an objective function operating on the parameters $\mathcal{O}$ and a measurement z_n. Matching the two equations, we get:

$$-\alpha \nabla \mathcal{L}(\mathcal{O}, z_n) = \mathcal{S}(k, z_n) - \gamma_0 \tag{9}$$

Under this new setting, performing OGM translates to optimizing an unknown objective function $\mathcal{L}$, whose gradients play the role of the desired map update provided by the inverse measurement model $\mathcal{S}$.

A trained OGM, $\mathcal{O}$, can be used to efficiently sample points along sensor rays $\vec{r}(t)$ during volumetric rendering in the following manner. Given the ray origin $\vec{o}$ and direction $\vec{d}$, we uniformly sample $N/2$ points $\vec{\zeta_j} = \vec{o} + \zeta_j \vec{d}$ along this ray where N is the total number of samples needed for volumetric rendering. We compute the log-odds values at these continuous coordinates, $l_{\vec{\zeta_j}}$, using trilinear interpolation of the discrete

grid $\mathcal{O}$. The corresponding occupancy is then computed as

$$p_{\vec{\zeta}_j} = \frac{1}{1 + \exp(-l_{\vec{\zeta}_j})} \tag{10}$$

Note that a value of $p = 0.5$ corresponds to unknown regions. We clamp and re-scale these probabilities from $[0.5, 1.0]$ to $[0.0, 1.0]$ and perform importance sampling to get $N/2$ additional samples around occupied regions. Finally, both the uniform samples and the additional samples are concatenated to construct N samples $\vec{\beta}_j = \vec{o} + \beta_j \vec{d}$ that are then used in volumetric rendering.

In order to train $\mathcal{O}$, we define a function g that defines the gradients of $l_{\vec{\beta}_j}$ as follows

$$\frac{\partial \mathcal{L}}{\partial l_{\vec{\beta}_j}} = g(\beta_j, z_n) = l_{\texttt{free}} \mathcal{U}((z_t - \delta) - \beta_j) - l_{\texttt{occ}} \mathcal{U}(\beta_j - (z_t - \delta)) \mathcal{U}((z_t + \delta) - \beta_j)$$

where $\mathcal{U}$ is the Heaviside step function, which takes the value of 1 for positive inputs and zero otherwise, and δ is a hyperparameter. These gradients can then be backpropagated through the trilinear interpolation operator to compute updates for γ_k, as shown pictorially in Panel B of Fig. 3, which shows an illustration of g. Note that we do not explicitly define $\mathcal{L}(\mathcal{O})$.

G. Training Details

We perform training of the two NeRF MLPs in three stages. In Stage 1, we only train the sigma MLP using LiDAR data for 2500 iterations with a batch size of 1024 rays per iteration. Note that rendering a LiDAR ray is significantly faster than rendering a camera ray since only the sigma MLP needs to be evaluated. In Stage 2, we freeze the weights of the sigma MLP and train the color MLP using camera data for 2500 iterations with 1024 rays per iteration. Finally, in Stage 3, we jointly fine-tune the full model for 10000 iterations with 1024 LiDAR rays and 1024 camera rays per iteration. In stages 1 and 3, we also train the OGM using the same batch of LiDAR data. We make one update with the OGM optimizer after accumulating gradients every 10 updates to the NeRF model. With the above settings, the model takes less than 12 minutes and 45 seconds on a single NVIDIA A100 GPU.

IV. Experimental Results

In this section we present image-only results for Novel View Synthesis and Depth Prediction tasks. For video results of Fig. 4 and Fig. 5, please see attached supplementary video.

A. Dataset and Training/Evaluation Protocol

For all experiments, we train and evaluate the proposed model and baselines on four short sequences from the KITTI dataset [22]. Each scene consists of three rectified stereo image pairs and three LiDAR scans corresponding to consecutive time steps, $t-1, t$ and $t+1$. For training each baseline, we use camera poses and intrinsics estimated by COLMAP [23], [24]. Since our proposed method operates in metric space, we use the pose and calibration information provided by the KITTI dataset to train CLONeR. For each scene, the two LiDAR scans and two left camera images at time $t-1$ and $t+1$, referred to as Image 000 and Image 002, are used for training. Note that only CLONeR uses the two LiDAR scans corresponding to time $t-1$ and $t+1$ for training. Quantitative and qualitative evaluation is performed on the remaining four images - left image at time t and all three right images. We refer to these test images as Image 001, Image 003, Image 004, and Image 005, respectively.

TABLE I: Quantitative results for novel view synthesis.

Scene	Algorithm	PSNR ↑	MSSIM ↑	LPIPS ↓
Scene 1	NeRF	16.99	0.73	0.14
	MIP-NeRF	13.51	0.47	0.24
	DS-NeRF*	20.39	0.87	0.09
	CLONeR	19.94	0.85	0.11
Scene 2	NeRF	19.90	0.87	0.08
	MIP-NeRF	14.31	0.62	0.15
	DS-NeRF*	14.97	0.70	0.13
	CLONeR	20.76	0.89	0.08
Scene 3	NeRF	20.02	0.89	0.05
	MIP-NeRF	15.28	0.73	0.10
	DS-NeRF*	22.63	0.94	0.04
	CLONeR	21.40	0.92	0.05
Scene 4	NeRF	19.95	0.87	0.08
	MIP-NeRF	14.52	0.62	0.16
	DS-NeRF*	18.55	0.83	0.09
	CLONeR	19.62	0.87	0.09
Average across Scenes	NeRF	19.21	0.84	0.09
	MIP-NeRF	14.41	0.61	0.16
	DS-NeRF*	19.14	0.83	0.09
	CLONeR	**20.43**	**0.87**	**0.08**

TABLE II: Quantitative results for dense depth prediction.

Scene	Algorithm	μSILog ↓	absErrRel ↓	sqErrRel ↓
Scene 1	NeRF	0.37	0.26	0.11
	MIP-NeRF	0.57	0.38	0.21
	DS-NeRF*	0.09	0.04	0.01
	CLONeR	0.09	0.07	0.01
Scene 2	NeRF	0.61	0.57	3.37
	MIP-NeRF	0.51	0.33	0.17
	DS-NeRF*	0.72	0.70	0.96
	CLONeR	0.07	0.06	0.01
Scene 3	NeRF	0.54	0.51	1.46
	MIP-NeRF	0.59	0.40	0.22
	DS-NeRF*	0.09	0.05	0.01
	CLONeR	0.09	0.07	0.01
Scene 4	NeRF	0.31	0.20	0.10
	MIP-NeRF	0.56	0.35	0.18
	DS-NeRF*	0.28	0.09	0.05
	CLONeR	0.14	0.08	0.03
Average across Scenes	NeRF	0.46	0.39	1.26
	MIP-NeRF	0.56	0.36	0.20
	DS-NeRF*	0.30	0.22	0.26
	CLONeR	**0.10**	**0.07**	**0.01**

B. Baseline methods

We compare our method against the following state-of-the-art methods: NeRF [1], Mip-NeRF [15], and DS-NeRF [10]. We train each of these models using the provided hyperparameters within each of the cited code bases. For NeRF, we used the original Tensorflow implementation. For Mip-NeRF, we use the available pytorch-lightning implementation [25]. For DS-NeRF, we used the original pytorch implementation [10]. Note that DS-NeRF is trained using 3D keypoints estimated by COLMAP computed on all 6 images. In this setting, COLMAP would have likely used information from the 4 test images as well to estimate 3D keypoints visible from the 2 training images. We attempted to train DS-NeRF using only keypoints that were estimated from the 2 training images but

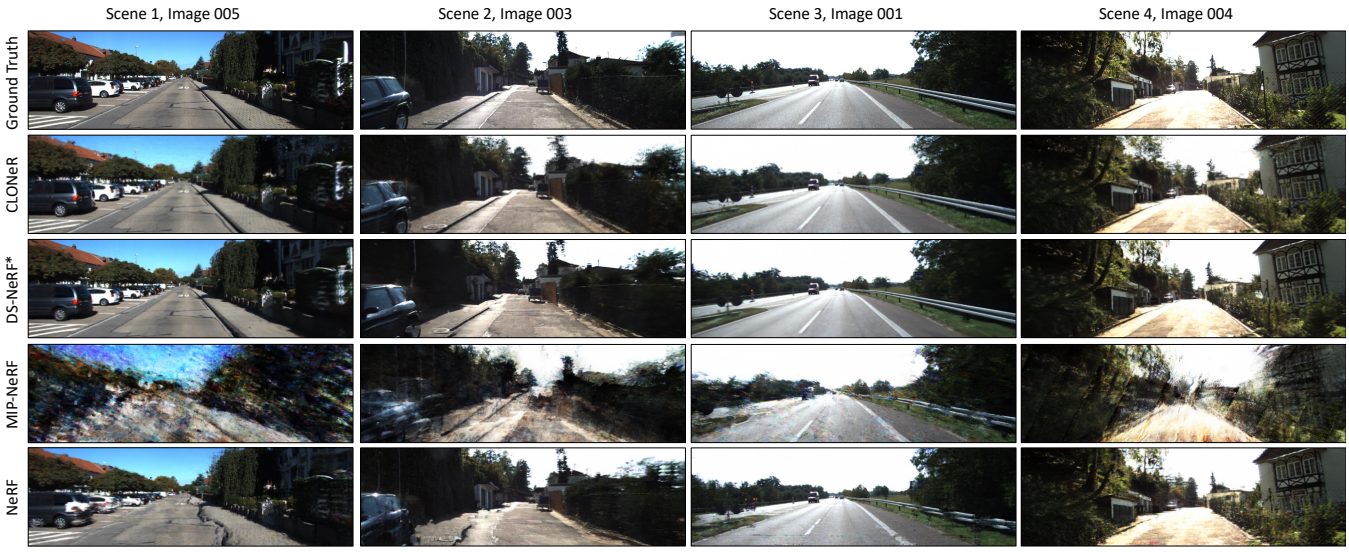

Fig. 4: Qualitative comparison of novel view synthesis results for 4 different scenes. Best viewed using a monitor, please zoom in to view details.

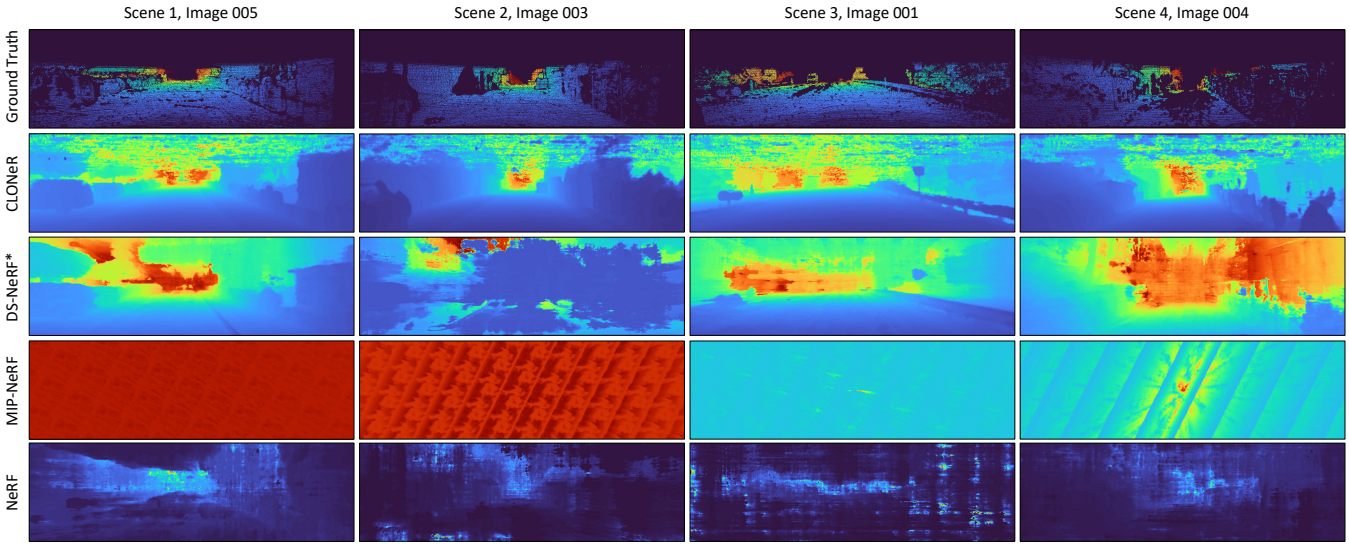

Fig. 5: Qualitative comparison of rendered depth for 4 different scenes. Ground truth depth is from the KITTI depth prediction benchmark. Best viewed using a monitor, please zoom in to view details.

the training did not converge. This gives an advantage to DS-NeRF compared to the other methods. We add an asterisk to all DS-NeRF results to denote this advantage.

C. Novel View Synthesis Results

We report three standard metrics for quantitative evaluation in Table I: PSNR, MSSIM, and LPIPS [26]. CLONeR outperforms all baselines across the three metrics averaged over the four scenes. Qualitative results on selected images from each of the four scenes are shown in Fig. 4. While all methods were able to recover general scene details, Mip-NeRF performed significantly worse than others. Both CLONeR and DS-NeRF achieve similar level of detail in the rendered novel views. However, poor depth learning in DS-NeRF results in objects being rendered at the wrong location. For example, in the Scene 2 column of Fig. 4, the trailer at the middle of the image is rendered incorrectly.

D. Depth Prediction Results

For quantitative evaluation, we use metrics and pseudo-ground truth depth maps from the KITTI depth prediction benchmark. While CLONeR operates in metric space, all the baselines can only estimate depth up-to scale (as determined by the COLMAP poses). Thus, we follow the evaluation protocol in [27] and align the median depth of the predicted depth maps with the ground truth before computing the metrics. As seen in Table II, CLONeR outperforms all baselines across the three metrics when averaged across the four scenes.

The pseudo-ground truth depth maps used in the quantitative evaluation along with qualitative results are shown in Fig. 5. Note that the pseudo-ground truth depth maps are sparse and contain many empty pixels where there is no ground truth. A visual inspection of the qualitative results demonstrates the improved performance of CLONeR compared to baselines

on all four scenes. MIP-NeRF suffers from significant planar artifacts, potentially from the geometry ambiguity failure mode that NeRF models experience when trained on sparse views [6]. The depth maps from NeRF lack any structural detail. While DS-NeRF learns detailed depth maps for scene 1 and 3, it struggles for scene 2 and 4. In comparison, CLONeR captures greater details in all scenes along with fine structures in the regions where LiDAR data is present.

E. Ablation Experiments

TABLE III: The different ablation experiments.

Type	Decoupled	LiDAR	LOS / Term	OGM	CF
Ablation A	✗	✗	N/A	✗	✗
Ablation B	✗	✓	✓/ ✗	✓	✗
Ablation C	✓	✓	✗/ ✓	✓	✗
Ablation D	✓	✓	✓/ ✗	✗	✓
Proposed	✓	✓	✓/ ✗	✓	✗

To analyze the different components of the proposed work, we consider 5 ablation experiments, described in Table III. We present the quantitative results in Table IV. The proposed method demonstrates comparable performance across ablation experiments for novel view synthesis, but yields the highest performance for depth prediction. In Figure 6, we show how the different features of CLONeR eliminate the failure modes that NeRFs have in outdoor scenes. Ablation A and B, which both use coupled NeRFs, experience the failure mode where both saturation and hard cast shadows are learned as separate objects or planes; these regions are highlighted in the yellow callout box and red callout box (respectively) in each depth map of Fig. 6. Ablation A, B, and C all experience the failure mode where fine structures cannot be learned. We observe that all the structures in the yellow callout box for these depth maps are noticeably blurry, which highlights not only the importance of LiDAR but also the proposed volumetric carving losses. Ablation D experiences the failure mode where both global and local smooth scene structure is difficult to learn without proper sampling in a large outdoor scene, highlighting the importance of the proposed OGM. Finally, we observe that all ablation models struggle to learn correct scene geometry in the upper parts of the image.

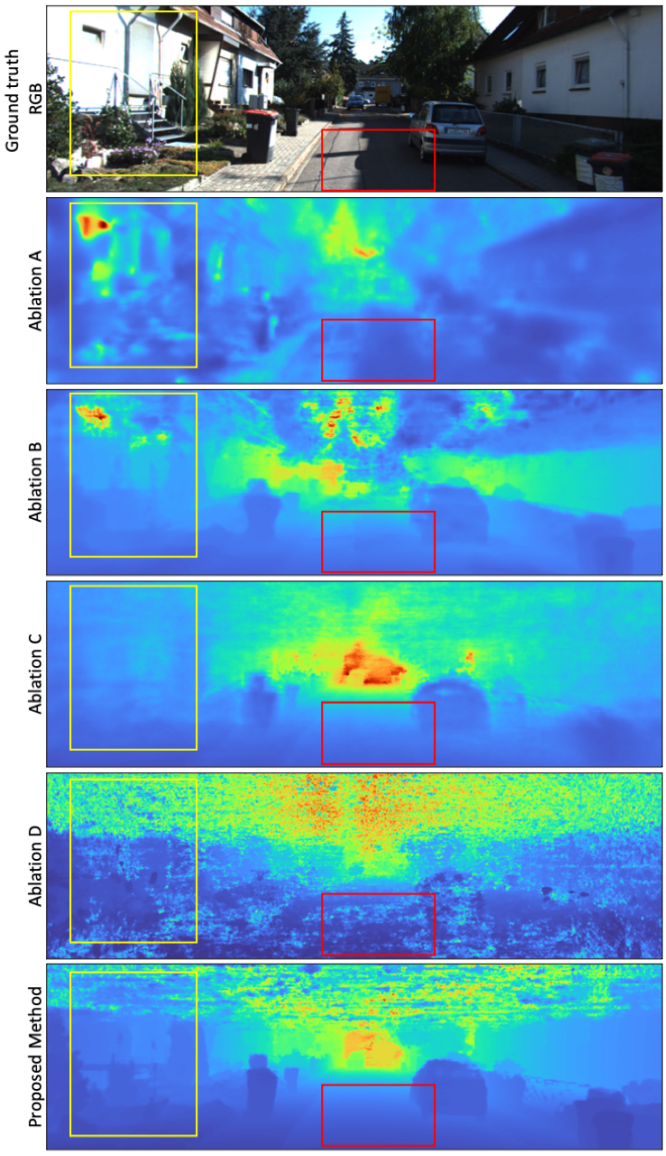

Fig. 6: Qualitative comparison of rendered depth maps from a test image for different ablations. The yellow boxes highlight regions of saturation and red boxes highlight hard cast shadows

V. Conclusion

This paper tackles the problem of novel view synthesis and dense depth reconstruction for large outdoor scenes with sparse input views. To this end, we proposed CLONeR, which extends NeRFs to operate on camera and lidar data and leverages occupancy grid mapping for efficient sampling, thus replacing the standard coarse NeRF MLP. Through both quantitative and qualitative experiments on the KITTI dataset, we demonstrate that the proposed method outperforms state-of-the-art algorithms on both tasks. Future work will investigate improving the performance of the proposed method in regions where lidar data is not present, but camera data is available, namely the sky and regions of the scene where lidar cannot be collected, such as through glass. Ultimately, this method opens up significant opportunity to begin using NeRFs to learn dense maps of outdoor scenes and can allow researchers to augment existing datasets with novel views and corresponding depth maps.

References

[1] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *European conference on computer vision*. Springer, 2020, pp. 405–421.

[2] E. R. Chan, M. Monteiro, P. Kellnhofer, J. Wu, and G. Wetzstein, "Pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 5799–5809.

[3] Z. Li, S. Niklaus, N. Snavely, and O. Wang, "Neural scene flow fields for space-time view synthesis of dynamic scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6498–6508.

[4] X. Zhang, P. P. Srinivasan, B. Deng, P. Debevec, W. T. Freeman, and J. T. Barron, "Nerfactor: Neural factorization of shape and reflectance under an unknown illumination," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 6, pp. 1–18, 2021.

TABLE IV: Ablation Quantitative evaluation for Novel View Synthesis and Depth estimation

Scene	Ablation	Novel View Synthesis		
		PSNR ↑	MSSIM ↑	LPIPS ↓
Scene 1	Ablation A	18.02	0.74	0.14
	Ablation B	14.77	0.66	0.21
	Ablation C	16.58	0.67	0.19
	Ablation D	17.90	0.77	0.13
	Proposed	19.67	0.86	0.09
Scene 2	Ablation A	21.10	0.87	0.08
	Ablation B	15.35	0.77	0.12
	Ablation C	16.65	0.76	0.13
	Ablation D	19.21	0.87	0.08
	Proposed	20.53	0.89	0.07
Scene 3	Ablation A	18.94	0.84	0.06
	Ablation B	17.05	0.86	0.08
	Ablation C	18.23	0.83	0.08
	Ablation D	13.13	0.75	0.16
	Proposed	21.04	0.92	0.04
Scene 4	Ablation A	19.65	0.84	0.08
	Ablation B	15.52	0.73	0.14
	Ablation C	17.71	0.80	0.11
	Ablation D	18.32	0.83	0.09
	Proposed	19.41	0.87	0.08
Average across scenes	Ablation A	19.42	0.83	0.09
	Ablation B	15.67	0.75	0.14
	Ablation C	17.29	0.77	0.13
	Ablation D	17.14	0.80	0.12
	Proposed	**20.16**	**0.89**	**0.07**

Scene	Ablation	Depth Prediction		
		SILog ↓	absErrRel ↓	sqrErrRel ↓
Scene 1	Ablation A	0.46	0.32	0.15
	Ablation B	0.57	0.28	0.11
	Ablation C	0.11	0.08	0.02
	Ablation D	0.41	0.37	0.27
	Proposed	0.09	0.07	0.01
Scene 2	Ablation A	0.50	0.37	0.21
	Ablation B	0.32	0.11	0.04
	Ablation C	0.12	0.09	0.02
	Ablation D	0.33	0.29	0.24
	Proposed	0.07	0.06	0.01
Scene 3	Ablation A	0.50	0.36	0.19
	Ablation B	0.40	0.12	0.05
	Ablation C	0.15	0.10	0.04
	Ablation D	0.34	0.30	0.24
	Proposed	0.10	0.06	0.02
Scene 4	Ablation A	0.54	0.42	0.24
	Ablation B	0.54	0.21	0.10
	Ablation C	0.14	0.09	0.03
	Ablation D	0.39	0.35	0.24
	Proposed	0.14	0.08	0.03
Average across scenes	Ablation A	0.50	0.37	0.20
	Ablation B	0.46	0.18	0.08
	Ablation C	0.13	0.09	0.03
	Ablation D	0.37	0.33	0.26
	Proposed	**0.10**	**0.07**	**0.02**

[5] E. Sucar, S. Liu, J. Ortiz, and A. J. Davison, "imap: Implicit mapping and positioning in real-time," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 6229–6238.

[6] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, "Mip-nerf 360: Unbounded anti-aliased neural radiance fields," *arXiv preprint arXiv:2111.12077*, 2021.

[7] K. Zhang, G. Riegler, N. Snavely, and V. Koltun, "Nerf++: Analyzing and improving neural radiance fields," *arXiv preprint arXiv:2010.07492*, 2020.

[8] R. Martin-Brualla, N. Radwan, M. S. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth, "Nerf in the wild: Neural radiance fields for unconstrained photo collections," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7210–7219.

[9] K. Rematas, A. Liu, P. P. Srinivasan, J. T. Barron, A. Tagliasacchi, T. Funkhouser, and V. Ferrari, "Urban radiance fields," *arXiv preprint arXiv:2111.14643*, 2021.

[10] K. Deng, A. Liu, J.-Y. Zhu, and D. Ramanan, "Depth-supervised nerf: Fewer views and faster training for free," *arXiv preprint arXiv:2107.02791*, 2021.

[11] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys, "Nice-slam: Neural implicit scalable encoding for slam," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022.

[12] T. Neff, P. Stadlbauer, M. Parger, A. Kurz, C. R. Alla Chaitanya, A. Kaplanyan, and M. Steinberger, "Donerf: Towards real-time rendering of neural radiance fields using depth oracle networks," *arXiv e-prints*, pp. arXiv–2103, 2021.

[13] B. Attal, E. Laidlaw, A. Gokaslan, C. Kim, C. Richardt, J. Tompkin, and M. O'Toole, "Törf: Time-of-flight radiance fields for dynamic scene view synthesis," *Advances in neural information processing systems*, vol. 34, pp. 26 289–26 301, 2021.

[14] J. Ost, I. Laradji, A. Newell, Y. Bahat, and F. Heide, "Neural point light fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 18 419–18 429.

[15] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan, "Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5855–5864.

[16] M. Tancik, V. Casser, X. Yan, S. Pradhan, B. Mildenhall, P. P. Srinivasan, J. T. Barron, and H. Kretzschmar, "Block-nerf: Scalable large scene neural view synthesis," *arXiv preprint arXiv:2202.05263*, 2022.

[17] M. Piala and R. Clark, "Terminerf: Ray termination prediction for efficient neural rendering," in *2021 International Conference on 3D Vision (3DV)*. IEEE, 2021, pp. 1106–1114.

[18] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa, "Plenoctrees for real-time rendering of neural radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5752–5761.

[19] C. Sun, M. Sun, and H.-T. Chen, "Improved direct voxel grid optimization for radiance fields reconstruction," *arXiv preprint arXiv:2206.05085*, 2022.

[20] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Trans. Graph.*, vol. 41, no. 4, pp. 102:1–102:15, Jul. 2022. [Online]. Available: https://doi.org/10.1145/3528223.3530127

[21] S. Thrun, "Probabilistic robotics," *Communications of the ACM*, vol. 45, no. 3, pp. 52–57, 2002.

[22] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 3354–3361.

[23] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm, "Pixelwise view selection for unstructured multi-view stereo," in *European Conference on Computer Vision (ECCV)*, 2016.

[24] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[25] J. Huang, "Unofficial pytorch-lightening implementation of mipnerf," 2022. [Online]. Available: $https://github.com/hjxwhy/mipnerf_pl$

[26] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *CVPR*, 2018.

[27] L. Wang, Y. Wang, L. Wang, Y. Zhan, Y. Wang, and H. Lu, "Can scale-consistent monocular depth be learned in a self-supervised scale-invariant manner?" in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12 727–12 736.