

# INF: Implicit Neural Fusion for LiDAR and Camera

Shuyi Zhou<sup>1,2</sup>, Shuxiang Xie<sup>1,2</sup>, Ryoichi Ishikawa<sup>1</sup>, Ken Sakurada<sup>2</sup>, Masaki Onishi<sup>2</sup> and Takeshi Oishi<sup>1</sup>

**Abstract**—Sensor fusion has become a popular topic in robotics. However, conventional fusion methods encounter many difficulties, such as data representation differences, sensor variations, and extrinsic calibration. For example, the calibration methods used for LiDAR-camera fusion often require manual operation and auxiliary calibration targets. Implicit neural representations (INRs) have been developed for 3D scenes, and the volume density distribution involved in an INR unifies the scene information obtained by different types of sensors. Therefore, we propose implicit neural fusion (INF) for LiDAR and camera. INF first trains a neural density field of the target scene using LiDAR frames. Then, a separate neural color field is trained using camera images and the trained neural density field. Along with the training process, INF both estimates LiDAR poses and optimizes extrinsic parameters. Our experiments demonstrate the high accuracy and stable performance of the proposed method.

## I. INTRODUCTION

With the rapid advancement in sensor technology, integrating measurements from multimodal sensors has become increasingly significant in many applications. The outputs of multiple sensors can be combined to reduce the uncertainty, resolve the ambiguity, and increase the robustness of a system [1]. LiDARs and cameras are commonly used sensors in robotics. A LiDAR can provide precise geometric information, whereas a camera can provide visual information about a scene. Combining LiDAR and camera data can help perform tasks such as simultaneous localization and mapping (SLAM) [2], object recognition, and 3D reconstruction.

Sensor calibration, which ensures that all sensors function correctly in a unified coordinate system, is a fundamental process during sensor fusion. However, many challenges hinder conventional calibration and fusion methods. In many cases, manual operation and auxiliary calibration targets are required. Popular solutions involve the use of single or multiple plane boards as targets [3], [4], [5], [6]. Some researchers have accomplished target-less LiDAR-camera calibration [7], [8], [9], [10]. However, the accuracy and robustness of these methods depend largely on scene complexity [7], and pre-trained models or additional prior knowledge is sometimes required [9], [10]. Therefore, automatic calibration between LiDARs and cameras remains challenging.

Following the trend of neural radiance fields (NeRF) [11], we consider the use of implicit neural representations (INRs) to solve the above problems in sensor fusion. The

<sup>1</sup>The authors are with The Institute of Industrial Science, The University of Tokyo, Japan. Emails: {zhoushuyi495, shxxie, ishikawa, oishi}@cvl.iis.u-tokyo.ac.jp

<sup>2</sup>The authors are with The National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan. Emails: {k.sakurada, onishi-masaki}@aist.go.jp

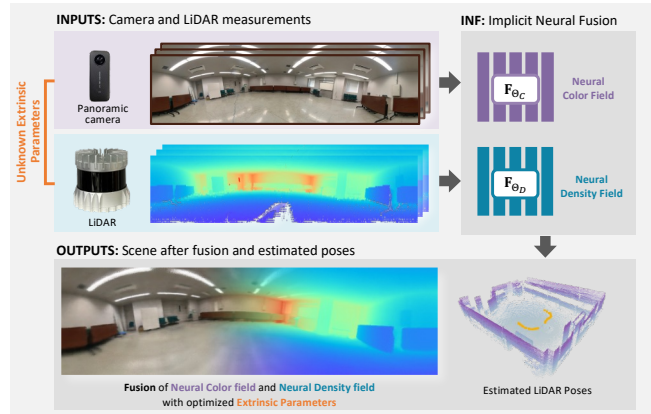


Fig. 1. Overview of INF. The proposed method takes camera and LiDAR observations as inputs to refine a neural color field and a neural density field. INF estimates the LiDAR pose of each frame and optimizes the extrinsic parameters to further accomplish sensor fusion.

volume density distribution given by an INR characterizes the geometric information of a space regardless of sensor type. Therefore, by obtaining a reliable volume density field, we may align all INRs using a unified volume density field and realize sensor fusion.

In this paper, we propose implicit neural fusion (INF) for LiDAR and camera measurements. Fig. 1 shows an overview of the proposed method. We first use LiDAR frames to generate a neural density field because LiDARs provide reliable, high-quality geometric information. In this step, INF is able to estimate the LiDAR pose for each frame as well. Then, we combine camera images and the trained neural density field to obtain and further refine a color field. Meanwhile, we optimize extrinsic parameters for fusion between LiDAR and camera.

Our contributions are summarized as follows:

- Provision of an implicit LiDAR-camera extrinsic parameter calibration method that does not require auxiliary calibration objects
- Introduction of a weighting technique for depth loss that improves the quality of neural density fields and the accuracy of LiDAR poses estimation
- Proposal of a sequential LiDAR pose estimation method using neural volume density fields

## II. RELATED WORK

This section gives a brief review of studies on conventional LiDAR-camera fusion techniques, basic INR concepts, and camera pose estimation methods using INRs.

### A. LiDAR-Camera Calibration

Based on the need for auxiliary targets, we classify LiDAR-camera calibration methods as target-based and target-less methods [12].

**Target-based Methods:** Target-based methods require a specified target in the scene. In most cases, the target can be a smooth plane with a feature pattern (such as checkerboard) [3], [13], [14], [15] or markers [16], [17] or some solid colors [17], [18]. The target can have different shapes, such as rectangle [3], [4], [13], [14], [15], [19], triangle [20], and sphere [18]. Applying multiple targets can improve calibration performance [5], [6], [21]. The advantages of target-based methods are high stability and accuracy. However, preparing calibration targets is inconvenient and takes time and cost. There are no such objects in most public dataset and real-life application scenarios.

**Target-less Methods:** Target-less methods can be classified as appearance-, motion-, and geometry-based methods [7]: (1). *Appearance-based methods* detect and pair appearance features between a camera and a LiDAR. For instance, Pandey et al. [22] maximized the mutual information (MI) between image color and LiDAR reflectivity; Levinson and Thrun [23] and Yuan et al. [24] detected and aligned the edges from both sensors' measurements; Schneider et al. [9] applied CNN to detect features from both of camera and LiDAR frames; Koide et al. [8] utilized SuperGlue to match the features; [25] Zhu et al. [26] and Jiang et al. [27] matched semantic information between the sensors. (2). *Motion-based methods* use the hand-eye calibration approach. They first calculate the motion of camera and LiDAR separately. Then, the relative pose of the sensors can be obtained by comparing the trajectories of both sensors [7]. (3). *Geometry-based methods* extract 3D geometric information of the scene from multiple camera frames and directly find the correspondence in LiDAR scans [28]. CalibNet [10] and CalibRCNN [29] used supervised neural networks to create depth map from images and compare it with LiDAR depths to align scenes.

Appearance and geometry based methods require explicit 2D or 3D features; that is, they are highly dependent on the target scene. Motion-based methods can hardly achieve high accuracy when the possible motions are restricted. While conventional works aim to match information from different domain data, our work uses a unified INR to align data derived from two sensors within the same domain.

### B. Implicit Neural Representations

In addition to traditional approaches, such as point-based [30], [31], mesh-based [32], [33], and voxel-based [34] methods, researchers have recently proposed continuous, differentiable, implicitly defined networks to represent 3D scenes. Park et al. and Mescheder et al. use signed distance functions or occupancy networks to describe surfaces implicitly [35], [36]. NeRF adopts volume rendering to encode appearance and features inside neural networks [11]. It samples multiple 3D points according to the camera position; these sample points further pass through a fully connected network to obtain corresponding radiance and density values.

[11]. Following certain works on NeRF, many pieces of research, such as [37], [38], [39], propose several techniques that improve the performance of NeRF.

However, to our best knowledge, few methods have been developed for integrating depth and color information obtained from independent sensors. Moreover, the NeRF and its variants usually require the camera poses as inputs. These poses are often estimated using structure from motion (SfM) techniques; this additional process may harm the compactness of the whole system.

### C. Camera Pose Estimation with INRs

Researchers have attempted to solve camera pose estimation problems using INRs. Many methods use the continuity of implicit neural networks to optimize input camera poses directly [40], [41], [42]. Jeong et al. implements camera distortion models in their framework to optimize both camera intrinsic and extrinsic parameters simultaneously [40]. The implementation of pose parameters also varies. Jeong et al. applies a 6-vector representation for rotation [40], while Lin et al. parameterized poses with the  $\mathfrak{se}(3)$  Lie algebra [41]. Both methods manage to converge.

Although the above mentioned methods demonstrate the feasibility of pose estimation, most existing methods perform targeting using RGB images. LiDAR depth is highly accurate and lightweight; thus, training speed and performance may be improved by the use of LiDAR data in pose estimation using INRs.

## III. IMPLICIT LiDAR-CAMERA FUSION

We propose INF, an Implicit Neural Fusion system, which integrates LiDAR and camera data to estimate the extrinsic parameters of the sensors without calibration targets. Our method takes sequential LiDAR and camera measurements of a scene as inputs, and outputs the optimized extrinsic parameters, neural density field, and neural color field of the scene. In this section, we describe the general framework and workflow of the proposed fusion system and explain the details in procedures.

### A. Assumptions

We assume the use of a LiDAR and a camera combined with a rigid joint. Thus, the extrinsic parameters between the LiDAR and the camera should remain the same regardless of how the LiDAR-camera system moves. We also assume that each pair of camera frame and LiDAR frame is taken at the same time, in other words, temporally synchronized. The target scene must be static; that is, it should have no moving objects.

### B. Problem Definition

The LiDAR and camera at the same frame are capturing the scene from different locations and angles. The physical distance between the optical centers of a LiDAR and a camera makes scenes reconstructed by the two sensors misaligned. Therefore, we need to estimate the extrinsic parameters between the LiDAR and the camera to fuse the two reconstructed scenes together.

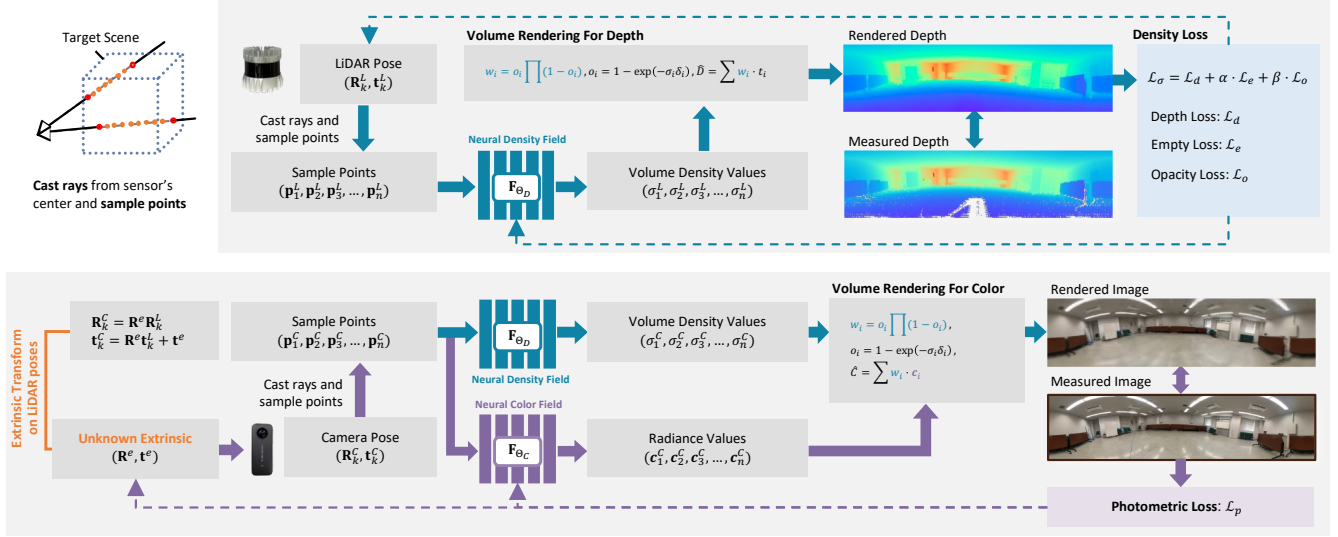


Fig. 2. General workflow of INF. We use LiDAR measurement data to train a neural density field that can represent the space geometry. Notice that LiDAR poses can also be estimated in this process. We also use camera data and trained density field to generate and refine a neural color field for color representation. Meanwhile, the extrinsic parameters can be optimized, which is the key part of sensor fusion. Processes shown by dashed lines indicate back-propagation, update and optimization.

We use LiDAR poses as bases. Although SfM techniques can powerfully reconstruct 3D scenes and estimate camera poses from image inputs, LiDAR point cloud alignment methods, such as the iterative closest point (ICP) algorithm still holds many advantages. First, LiDAR measurements can provide an absolute scale. Second, aligning LiDAR data takes lower computational costs. Third, LiDAR pose registration methods are more robust in textureless scenarios, especially in indoor cases. Since the extrinsic parameters will not vary during the measurement, we may derive the camera poses using LiDAR poses and extrinsic parameters. Instead of optimizing every independent camera pose, optimizing only the extrinsic parameters makes the whole process easier.

We first define the extrinsic parameters as  $(\mathbf{R}^e, \mathbf{t}^e)$ . We also let  $\{\mathcal{P}^L\}$  be the set containing poses for all LiDAR frames. Define pose of  $k$ -th LiDAR frame pose as  $\mathcal{P}_k^L = (\mathbf{R}_k^L, \mathbf{t}_k^L)$ , and we can have  $\mathcal{P}_k^L \in \{\mathcal{P}^L\}$ . Then we may derive camera pose by applying  $(\mathbf{R}^e, \mathbf{t}^e)$  on  $\mathcal{P}_k^L$ . Let  $k$ -th camera pose be  $(\mathbf{R}_k^C, \mathbf{t}_k^C)$ , we have

$$\mathbf{R}_k^C = \mathbf{R}^e \mathbf{R}_k^L, \mathbf{t}_k^C = \mathbf{R}^e \mathbf{t}_k^L + \mathbf{t}^e. \quad (1)$$

Following the settings in [41], all poses are parameterized with the  $\mathfrak{se}(3)$  Lie algebra in this study. We further define two fully-connected neural networks as scene representations: Neural Density Field  $\mathbf{F}_{\Theta_D} : \mathbf{x} \rightarrow \sigma$ , and Neural Color Field  $\mathbf{F}_{\Theta_C} : \mathbf{x} \rightarrow \mathbf{c}$ .  $\mathbf{F}_{\Theta_D}$  is used to represent the volume density distribution and  $\mathbf{F}_{\Theta_C}$  is used to represent the color of every particles in the space.

Our INF system takes LiDAR frames  $\{\mathcal{F}_1^L, \mathcal{F}_2^L \dots, \mathcal{F}_N^L\}$ , camera frames  $\{\mathcal{F}_1^C, \mathcal{F}_2^C \dots, \mathcal{F}_N^C\}$ , and manually set initial value of  $(\mathbf{R}^e, \mathbf{t}^e)$  as inputs. By taking advantages of implicit neural representation, INF outputs LiDAR poses  $\{\mathcal{P}^L\}$ , optimized extrinsic parameters  $(\mathbf{R}^e, \mathbf{t}^e)$ , and refined neural networks  $\mathbf{F}_{\Theta_D}$  and  $\mathbf{F}_{\Theta_C}$ . Note that,  $(\mathbf{R}^e, \mathbf{t}^e)$  are applied

to transform camera poses to world coordinates. Therefore, optimizing  $(\mathbf{R}^e, \mathbf{t}^e)$  is equivalent to fusing  $\mathbf{F}_{\Theta_D}$  and  $\mathbf{F}_{\Theta_C}$ .

### C. Framework and Procedures

Fig. 2 gives an overview of the proposed INF system. Here, we explain the training procedures of the mentioned neural networks and parameters.

1) *Training of Neural Density Field*: We train a neural density field  $\mathbf{F}_{\Theta_D}$  using LiDAR measurement data. Notice that, LiDAR poses  $\{\mathcal{P}^L\}$  can be obtained by two means. First,  $\{\mathcal{P}^L\}$  is given by an Iterative Closest Point (ICP) method. Second,  $\{\mathcal{P}^L\}$  is estimated together with  $\mathbf{F}_{\Theta_D}$ . The second method does not need output from pre-alignment so all procedures are implemented with INRs.

Take  $k$ -th LiDAR frame  $\mathcal{P}_k^L = (\mathbf{R}_k^L, \mathbf{t}_k^L)$  as an example. Hereafter, subscript  $k$  will be omitted for simplicity. First, we use  $\mathcal{P}^L$  to calculate the laser rays in the world coordinates and then follow the techniques applied in [11] to sample several 3D points along the directions of laser rays. These samples are input into the defined  $\mathbf{F}_{\Theta_D}$  after positional encoding. The corresponding volume density value for each sample is the output of  $\mathbf{F}_{\Theta_D}$ . Next, volume rendering is applied to obtain the estimated depth  $\hat{D}$ . We use our designed loss functions to compare  $\hat{D}$  with observed depth  $D$ . Finally, we propagate the loss values back to optimize  $\mathbf{F}_{\Theta_D}$ . Notice that the loss functions will be the same if  $\{\mathcal{P}^L\}$  is optimized here. The details of LiDAR pose estimation will be discussed in the following section.

2) *Training of Neural Color Field*: We train a neural color field  $\mathbf{F}_{\Theta_C}$  and optimize the extrinsic parameters simultaneously. Instead of directly defining camera poses, we use the derived camera poses as shown in equation 1. Then with the derived camera poses, we can obtain sample points along the camera rays. These sampled points are input to both  $\mathbf{F}_{\Theta_D}$

and  $\mathbf{F}_{\Theta_C}$  to get density values and radiance values. After volume rendering, estimated pixel color  $\hat{\mathbf{C}}$  is compared with observed color  $\mathbf{C}$  to obtain photometric loss. We may give the initial value of  $(\mathbf{R}^e, \mathbf{t}^e)$  manually before training. The most straightforward setting is *zero*, which means the camera is at the same pose and position of the LiDAR. This initial value setting may be inaccurate, leading to imperfect scene reconstruction and fusion. However, by back-propagating the loss function, we gradually optimize both  $\mathbf{F}_{\Theta_C}$  and  $(\mathbf{R}^e, \mathbf{t}^e)$ .

#### IV. IMPLICIT NEURAL REPRESENTATIONS FOR CALIBRATION AND SCENE FUSION

In this section, we describe color and depth rendering, the loss design for  $\mathbf{F}_{\Theta_C}$  and  $\mathbf{F}_{\Theta_D}$ , and LiDAR poses estimation techniques.

##### A. Color and Depth Rendering

As a core concept of NeRFs [11], *volume rendering* has been extensively discussed. Besides RGB color rendering, volume rendering is widely applied to obtain estimated depths from volume density samples [38], [42]. In this study, we use both color and depth rendering.

Volume rendering is completed through sampling, querying, and integrating. We first sample 3D points along rays starting from the LiDAR/camera centers. Then, we query the neural networks to obtain the corresponding properties, such as radiance and volume density. Let  $\mathbf{p}_i = (x_i, y_i, z_i)$  be a 3D point, then we have the RGB color vector  $\mathbf{c}_i = \mathbf{F}_{\Theta_C}(\mathbf{p}_i)$  and the volume density value  $\sigma_i = \mathbf{F}_{\Theta_D}(\mathbf{p}_i)$ . The rendered color and depth for each ray can be obtained by integrating the different properties. Color rendering follows  $\hat{\mathbf{C}} = \sum w_i \cdot \mathbf{c}_i$ , whereas depth rendering follows  $\hat{D} = \sum w_i \cdot t_i$ .  $\mathbf{c}_i$  and  $t_i$  represent the color and depth value, respectively, for each sample.  $w_i$  denotes the transparency weight according to the setting in [11], [42], and it is calculated as  $w_i = o_i \prod_{j=1}^{i-1} (1 - o_j)$ , where  $o_i = 1 - \exp(-\sigma_i \delta_i)$ . In this way, both color  $\hat{\mathbf{C}}$  and depth  $\hat{D}$  are calculated.

##### B. Loss for Neural Color Field

After obtaining rendered RGB vector  $\hat{\mathbf{C}}$ , we compare it with the observed color  $\mathbf{C}$  to update  $\mathbf{F}_{\Theta_C}$ . Given  $n$  rays in total, the photometric loss  $\mathcal{L}_p$  is obtained as,

$$\mathcal{L}_p = \frac{1}{n} \sum_{i=1}^n (\mathbf{C}_i - \hat{\mathbf{C}}_i)^2. \quad (2)$$

$\mathbf{F}_{\Theta_C}$  and  $(\mathbf{R}^e, \mathbf{t}^e)$  are optimized simultaneously. The optimization process here is described as

$$\Theta_C, \mathbf{R}^e, \mathbf{t}^e = \arg \min_{\Theta_C, \mathbf{R}^e, \mathbf{t}^e} \mathcal{L}_p. \quad (3)$$

##### C. Loss for Neural Density Field

We design density loss as a combination of three parts: depth loss  $\mathcal{L}_d$ , empty loss  $\mathcal{L}_e$ , and opacity loss  $\mathcal{L}_o$ . The total loss for density field training is

$$\mathcal{L}_\sigma = \mathcal{L}_d + \alpha \cdot \mathcal{L}_e + \beta \cdot \mathcal{L}_o, \quad (4)$$

where  $\alpha$  and  $\beta$  are used to scale the losses. The optimization process related to the neural density field is

$$\Theta_D, \{\mathcal{P}^L\} = \arg \min_{\Theta_D, \{\mathcal{P}^L\}} \mathcal{L}_\sigma. \quad (5)$$

**Depth Loss:** Depth loss compares the ground truth depth  $D_i$  and rendered depth  $\hat{D}_i$  as applied in many research [38], [43]. Instead of directly using L1 norm as  $|D_i - \hat{D}_i|$  like in [38], [43], we use a weighting technique to emphasize the points near edges. Many LiDAR points are sampled at smooth planes, such as walls and grounds, which contribute too much to the loss. However, these points help little to the convergence of  $\{\mathcal{P}^L\}$  because of the point-to-point nature of the error metric. On the contrary, the points located around discontinuous parts, such as edges, are more significant. Therefore, we propose to weigh the depth loss according to whether the LiDAR ray reaches around the edges.

The proposed weighting technique characterizes the extent of discontinuity for LiDAR points using the normal vector difference of neighboring points. Let  $\eta_i$  be the depth weight of  $i$ -th ray,  $\mathbf{p}_i$  be the point reached by  $i$ -th ray, and  $\mathbf{n}_i$  be the normal vector with unit length at the location of  $\mathbf{p}_i$ . We label the rays according to their vertical and horizontal order in a scan. Let  $(j, k)$  be the point index of  $\mathbf{p}_i$  in the LiDAR frame. Then we define  $\Phi_i$  as the set of normal vectors of neighboring points of  $\mathbf{p}_i$ . The neighboring points are defined based on their index, such as  $(j-1, k-1), (j-1, k) \dots (j+1, k+1)$ . Thus  $|\Phi_i| = 8$ . The depth weight  $\eta_i$  is calculated as,

$$\eta_i = \frac{\lambda}{2} \left( 1 - \frac{1}{|\Phi_i|} \sum_{\mathbf{n}_\phi \in \Phi_i} \langle \mathbf{n}_i, \mathbf{n}_\phi \rangle \right) + (1 - \lambda). \quad (6)$$

$\langle \cdot, \cdot \rangle$  is a vector dot product,  $\lambda \in [0, 1]$  is a hyper-parameter representing how much edge points are emphasized. Larger  $\lambda$  means the higher priority of edge points. Further, for all  $n$  rays, we define the depth loss as,

$$\mathcal{L}_d = \frac{1}{n} \sum_{i=1}^n \eta_i |D_i - \hat{D}_i|. \quad (7)$$

**Empty Loss:** Since  $i$ -th LiDAR ray reaches  $\mathbf{p}_i$ , then no obstacle should lie between the LiDAR center and  $\mathbf{p}_i$ . This means that the transparency weights should be 0 for the sample points with depth  $t < D_i$ . Consider  $i$ -th ray among all  $n$  rays, where  $K$  samples exist. Let the transparency weight of  $k$ -th sample point on  $i$ -th ray be  $w_{(i,k)}$ , and let the corresponding depth value be  $t_{(i,k)}$ . We define the  $\lambda$ -th sample point as the point that satisfies  $t_{(i,\lambda)} < D_i - \epsilon < t_{(i,\lambda+1)}$  on  $i$ -th ray.  $\epsilon$  is a very small value modeling the error caused by discrete sampling. Then we may write the empty loss as,

$$\mathcal{L}_e = \frac{1}{n} \sum_{i=1}^n \sum_{k=1}^{\lambda} (w_{(i,k)})^2. \quad (8)$$

**Opacity Loss:** LiDAR can only reach points within the scanning range. For example, the scanning range of Ouster OS0 LiDAR is  $0.25 \sim 35\text{m}$ . If one laser does not detect

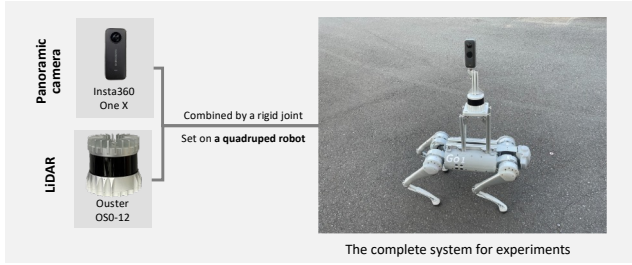


Fig. 3. Experiment setup. We use a panoramic camera and a LiDAR. They are connected with a rigid joint and set on a quadruped robot.

any point within this range, the output depth value will be 0. Opacity loss describes this phenomenon. We denote the opacity for  $i$ -th ray among  $n$  rays as  $y_i$ . For the rays reaching objects,  $y_i$  should be 1; for the rays that directly go outside the range,  $y_i$  should be 0. The opacity calculation is similar to the empty loss calculation, which is  $\hat{y}_i = \sum_{k=1}^K w_{(i,k)}$ , when there are  $K$  sample points on  $i$ -th ray. We use binary entropy loss to characterize the opacity difference between rays,

$$\mathcal{L}_o = -\frac{1}{N} \sum_{i=1}^N (y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log (1 - \hat{y}_i)). \quad (9)$$

#### D. LiDAR Pose Estimation

As stated in Section III-C, instead of using LiDAR poses given by ICP methods, we can estimate  $\{\mathcal{P}^L\}$  together with  $\mathbf{F}_{\Theta_D}$ . In this way, we may increase the compactness of the system without losing accuracy in LiDAR pose estimation.

Motivated by conventional SLAM systems, we apply the concept of local maps and keyframes to reduce accumulated errors. For LiDAR frames  $\{\mathcal{F}_1^L, \mathcal{F}_2^L \dots, \mathcal{F}_N^L\}$ , we define the keyframes as  $\{\mathcal{F}_{\kappa_1}^L, \mathcal{F}_{\kappa_2}^L \dots, \mathcal{F}_{\kappa_n}^L\}$ . Frames  $\mathcal{F}_i^L$ , where  $\kappa_a \leq i < \kappa_{a+1}$ , will form a local map. We set  $\kappa_1 = 1$ , and select keyframes based on the distance between the current frame and the previous keyframe. If the distance is larger than  $\tau$ , we will consider the current frame as a keyframe.

We optimize within each local map to refine the density field and estimate relative poses between normal frames and the keyframe. The detailed process is shown in Alg. 1.

## V. EXPERIMENTS

### A. Experimental Setup

In our experiments, we use Ouster OS0-128, whose resolution is  $1024 \times 128$ , and Insta360 One X, whose resolution is  $6080 \times 3040$ . We fix the camera on the LiDAR using a rigid joint on a Unitree Go1 quadruped robot as shown in Fig. 3. When taking a shot, we move the quadruped robot by a small step and then capture a camera image and a LiDAR scan simultaneously after the robot stays still to ensure temporal synchronization.

Fig. 4 shows the three captured scenes. The first scene is a classroom with no occlusions; the second scene is a meeting room with various objects, such as desks and chairs; the third

### Algorithm 1 LiDAR Poses Optimization

---

**Input:**  $\{\mathcal{F}_1^L, \mathcal{F}_2^L \dots, \mathcal{F}_N^L\}, \mathbf{F}_{\Theta_D}, \tau$   
**Output:** Optimized  $\Theta_D, \{\mathcal{P}^L\}$

- 1:  $i \leftarrow 1$  ▷ LiDAR frame index
- 2: **while**  $i < N$  **do**
- 3:   **Optimize**  $\mathbf{F}_{\Theta_D}$  **with**  $\mathcal{F}_i^L$
- 4:    $d \leftarrow 0$  ▷ distance  $d$  to previous keyframe
- 5:    $c \leftarrow i$  ▷ current frame index  $c$
- 6:   **while**  $d < \tau$  **do**
- 7:      $c \leftarrow c + 1$
- 8:     **Estimate**  $\mathcal{P}_c^L$  **with**  $\mathbf{F}_{\Theta_D}$  **and**  $\mathcal{F}_c^L$
- 9:      $\mathcal{P}_{c+1}^L \leftarrow \mathcal{P}_c^L$  ▷ set initial pose of next frame
- 10:      $d \leftarrow \|\mathcal{P}_c^L - \mathcal{P}_{c-1}^L\|_2$
- 11:   **end while**
- 12:    $\{\mathcal{F}_l\} \leftarrow \{\mathcal{F}_i^L, \dots, \mathcal{F}_{c-1}^L\}$  ▷ local map
- 13:   **Optimize**  $\mathbf{F}_{\Theta_D}$  **and**  $\{\mathcal{P}_{i+1}^L, \dots, \mathcal{P}_{c-1}^L\}$  **with**  $\{\mathcal{F}_l\}$
- 14:    $i \leftarrow c - 1$  ▷ next keyframe
- 15: **end while**

---

scene is an outdoor scene with trees and buildings. We use 30 frames for each scene in the following experiments.

### B. Ground Truth of Extrinsic Parameters

We obtain the ground-truth LiDAR-camera extrinsic parameters by minimizing the reprojection error  $\mathcal{E}_r$  of the manually selected corresponding points between the camera frames  $\mathbf{p}_i^C$  and LiDAR frames  $\mathbf{p}_i^L$ . The residual function is

$$\mathcal{E}_r = \sum_{i=1}^N T(\|\pi(\mathbf{R}^e \mathbf{p}_i^L + \mathbf{t}^e) - \mathbf{p}_i^C\|_2), \quad (10)$$

where  $\pi(\cdot)$  is the camera projection function and  $N$  is the total number of selected points.  $T(\cdot)$  is the Tukey loss function; it aims to avoid the error caused by manual selection. The reprojection errors of 90% of selected points are within 30 pixels.

## VI. RESULTS

### A. Calibration Results

The LiDAR-camera extrinsic parameters vary from scene to scene. However, we assume the camera and LiDAR are at the same relative pose throughout every single scene. The initial values for  $(\mathbf{R}^e, \mathbf{t}^e)$  are about  $10^\circ/0.2\text{m}$  from ground truth values. The calibration results of different scenes are shown in Table I, where errors are calculated using absolute value. Fig. 6 (left) also gives the visualization of scenes after fusion.

Given the absence of specific target objects in our data, we use targetless methods based on appearance, motion, and geometry to perform comparative experiments. The compared methods are described below.

(1). **Mutual Info:** The mutual information between LiDAR scan reflectivity and the camera image color is minimized [22].



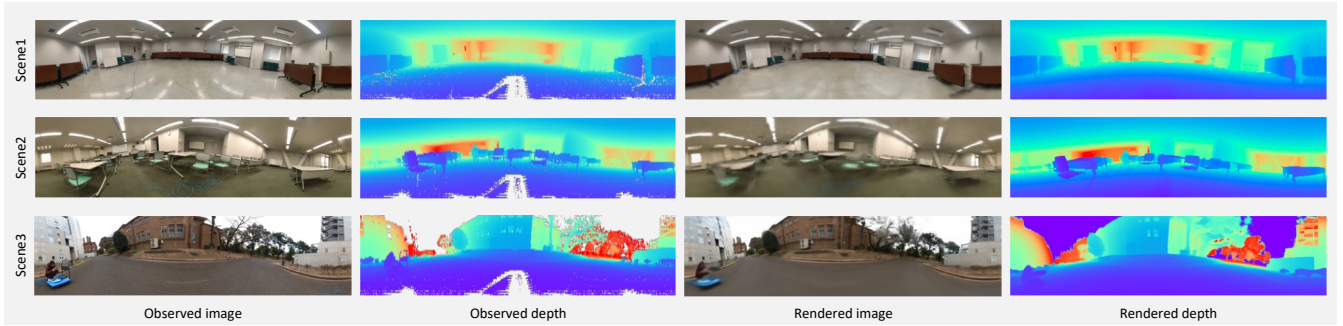


Fig. 4. Observed and rendered scenes. RGB images taken by panoramic camera and point clouds captured by LiDAR for all three scenes are shown on the left part. On the right side, we also show images rendered by the neural color field and depth rendered by the neural density field.

TABLE I  
RESULTS OF EXTRINSIC CALIBRATION USING DIFFERENT METHODS

	Scene1: Classroom				Scene2: Meeting Room				Scene3: Outdoor			
	Translation [m] & Rotation [°] Error				Translation [m] & Rotation [°] Error				Translation [m] & Rotation [°] Error			
	x	y	z	r	x	y	z	r	x	y	z	r
Mutual Info[22]	0.072	0.032	0.259	0.190	0.105	0.031	0.155	16.902	0.055	<b>0.015</b>	0.181	2.702
Jump Edge[23]	0.019	0.031	0.271	3.288	0.055	0.048	0.171	16.015	<b>0.001</b>	0.078	0.336	7.151
Motion[7]	0.196	0.065	1.450	4.863	0.143	0.043	0.728	6.512	0.063	0.075	0.783	0.868
Continuous Edge[24]	0.619	0.332	0.738	21.473	0.085	0.094	0.222	17.690	0.546	3.090	2.313	71.215
INF w/ g.t. pose (ours)	<b>0.005</b>	0.006	0.020	<b>0.299</b>	<b>0.004</b>	0.010	<b>0.009</b>	<b>0.385</b>	0.013	0.017	0.020	<b>0.285</b>
INF w/o g.t. pose (ours)	0.009	<b>0.004</b>	<b>0.015</b>	0.680	0.009	<b>0.002</b>	0.016	0.511	0.015	0.037	<b>0.010</b>	0.500

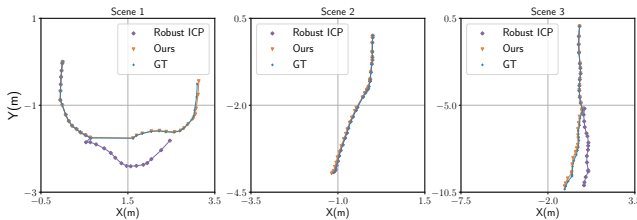


Fig. 5. Comparison between estimated LiDAR poses using different methods. We compare the results obtained from INF (Ours), and traditional ICP methods without manual initialization.

TABLE II  
RESULTS OF LiDAR POSE ESTIMATION (APE & RPE)

	Method	APE / RPE	
		Translation [m]	Rotation [°]
Scene1	Ours	<b>0.038 / 0.009</b>	<b>0.704 / 0.133</b>
	Robust ICP	1.173 / 0.268	27.536 / 7.222
Scene2	Ours	0.036 / 0.007	0.668 / 0.142
	Robust ICP	<b>0.017 / 0.002</b>	<b>0.099 / 0.029</b>
Scene3	Ours	<b>0.161 / 0.121</b>	<b>0.509 / 0.162</b>
	Robust ICP	0.660 / 0.131	9.256 / 1.657

- (2). **Jump Edge:** The edges in the LiDAR and camera frames are matched. Edges are detected by considering the discontinuity of LiDAR points [23].
- (3). **Motion:** This is a hand-eye calibration approach. The method uses ground-truth LiDAR poses and alternatively iterates the updating of extrinsic parameters and camera poses [7].
- (4). **Continuous Edge:** LiDAR point cloud edges are detected by cutting the point cloud into voxels and fitting

the points into planes. Then, the edges in the LiDAR and camera frames are matched. This edge detection method requires a high-resolution LiDAR [24].

- (5). **INF w/ ICP LiDAR pose (ours):** Proposed INF system with the LiDAR poses given by ICP as inputs.
- (6). **INF w/o ICP LiDAR pose (ours):** Proposed INF system without the LiDAR poses given by ICP. Instead, the LiDAR poses are estimated simultaneously with the density field.

Generally speaking, our method has the highest accuracy among all methods. Jump Edge [23] and Mutual Info [22] give small translation errors in the x and y axes, but these methods could not converge to the correct position in the z-axis and the rotation. The proposed method is also robust enough to converge in the different scenes, whereas the other methods, such as Mutual Info and Jump Edge, fall into local minima easily. Continuous Edge does not function well due to the relatively low resolution of the LiDAR used. Moreover, the compared methods do not handle translation in the z direction properly, whereas our method performs well in such a case. Fig. 4 indicates the good rendering quality by INF of both the neural color and neural density fields. Fig. 6 (left) shows that two INRs are aligned accurately.

### B. LiDAR Poses Estimation Results

We evaluate our estimated LiDAR poses using ground-truth poses. We also compare our method with an ICP method implemented by the public library Open3D [44], [45], [46], which is denoted as Robust ICP. The results are shown in Table II. Fig. 5 also gives a comparison between the LiDAR trajectory estimated by different methods and the

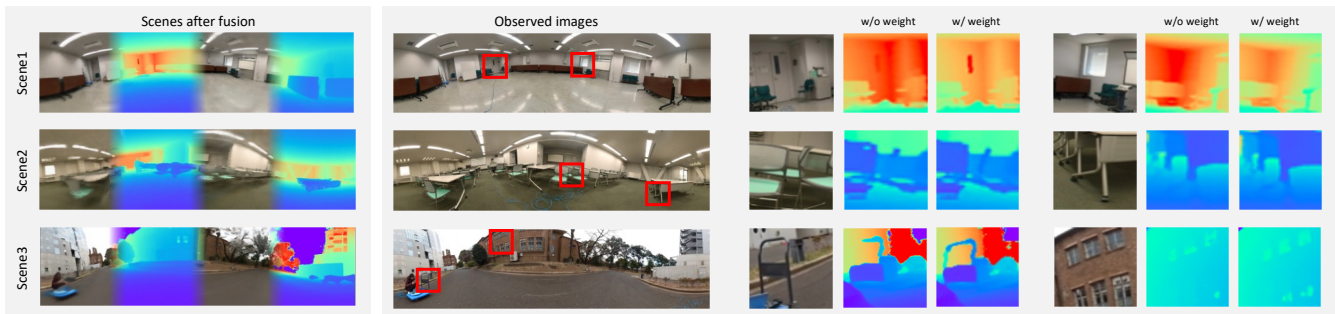


Fig. 6. **Left:** Fusion of neural density field and neural color field. The two INRs are aligned. **Right:** Qualitative evaluation of the rendered depth with or without weight introduced in section IV-C. We can observe detailed structure more clearly when the weighting technique is involved.

ground truth. Results show that INF holds higher accuracy compared with Robust ICP in scene 1 and scene 3. Robust ICP cannot handle large displacement between frames, while INF maintains good performance. Robust ICP aligns point clouds directly, while our method aligns them with the trained neural density field; the accuracy of the neural density field affects the accuracy of pose estimation. In scene 2, though Robust ICP has a lower error, INF also shows reasonable accuracy.

### C. Ablation Study on Depth Weight

An ablation study is carried out for the weighting technique in Section IV-C for LiDAR pose estimation and LiDAR-camera calibration. The qualitative result is shown in Table III. We also evaluate the rendered results qualitatively as shown in Fig. 6 (right). The performance difference with and without weighting indicates the effectiveness of our methods. In Fig. 6 (right), we can also observe the improvement in scene reconstruction, especially in the detailed place with complex geometric features. Table III indicates the reduction in pose error after applying the weighting technique in indoor scenes. In the outdoor case, the APE becomes larger with the weighting technique due to the existence of noises such as trees. However, the error is still small considering the total moving distance. Therefore, we may consider the proposed weighting technique helps the system with accuracy.

### D. Ablation Study on Initial Value Setting

We also test the convergence of our method with respect to different initial values, including initial rotation values and initial translation values in both indoor and outdoor scenes. The result is shown in Fig. 7. In our experiment, the convergence speed becomes a problem when a large initial bias is applied. However, our method can converge even with an initial translation of 0.7m or an initial rotation of 80° away from the correct value. Compared with other neural network based methods, such as 1.5m/20° in RegNet [9], 0.25m/7.5° in CalibNet [10] and 0.25m/10° in CalibRCNN [29], INF shows relatively high robustness against different initial value settings.

TABLE III  
RESULTS OF ABLATION STUDY FOR WEIGHTING

		Error of Translation [m] / Rotation [°]	
		APE of LiDAR Pose Estimation	Calibration Error
Scene1	w/	<b>0.038 / 0.704</b>	<b>0.018 / 0.647</b>
	w/o	0.109 / 0.883	0.030 / 0.733
Scene2	w/	<b>0.036 / 0.668</b>	<b>0.018 / 0.511</b>
	w/o	0.054 / 0.764	0.020 / <b>0.431</b>
Scene3	w/	0.161 / 0.509	<b>0.041 / 0.500</b>
	w/o	<b>0.133 / 0.371</b>	0.047 / <b>0.348</b>

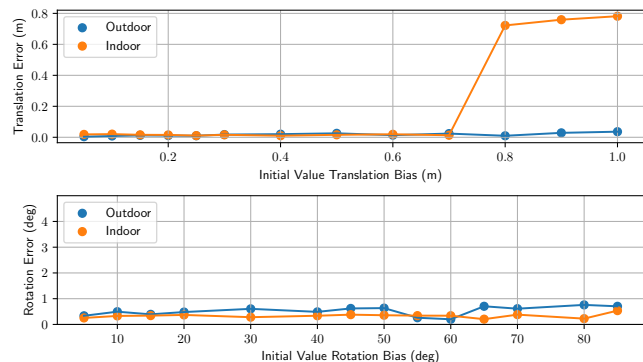


Fig. 7. Calibration error with respect to initial extrinsic parameters bias. INF manages to converge even initial extrinsic parameters are set relatively far from ground truth.

## VII. CONCLUSION

In this paper, we propose INF for LiDAR-camera extrinsic calibration without auxiliary calibration targets. To our best knowledge, this work is the first to attempt to solve this problem using INRs. The proposed system exhibits good performance and robustness on several real-world datasets, and the proposed techniques are proven effective. Furthermore, unlike some target-less calibration and fusion methods, INF does not require prior knowledge or pre-trained models. Thus, our method could suit various scenarios.

Future work will aim to improve the convergence speed of INF. We will closely examine and try to make good use of the geometric features hidden inside neural representations. We also plan to implement temporal synchronization in the system to enrich the input types of data. In addition,

we aim to extend INF to other types of sensors, such as radar, temperature sensors, and event cameras. The sensor fusion process shall become more compact and convenient by leveraging the implicitness of the system.

#### ACKNOWLEDGEMENT

This work was partially supported by JST, PRESTO Grant Number JPMJPR22C4, Japan.

#### REFERENCES

- [1] J. K. Hackett and M. Shah, "Multi-sensor fusion: a perspective," in *Proceedings, ICRA*. IEEE, 1990, pp. 1324–1330.
- [2] H. Zhong, H. Wang, Z. Wu, C. Zhang, Y. Zheng, and T. Tang, "A survey of lidar and camera fusion enhancement," *Procedia Computer Science*, vol. 183, pp. 579–588, 2021.
- [3] S. Verma, J. S. Berrio, S. Worrall, and E. Nebot, "Automatic extrinsic calibration between a camera and a 3d lidar using 3d point and plane correspondences," in *2019 ITSC*. IEEE, 2019, pp. 3906–3912.
- [4] Y. Zhao, K. Huang, H. Lu, and J. Xiao, "Extrinsic calibration of a small fov lidar and a camera," in *2020 CAC*. IEEE, 2020, pp. 3915–3920.
- [5] S. Sim, J. Sock, and K. Kwak, "Indirect correspondence-based robust extrinsic calibration of lidar and camera," *Sensors*, vol. 16, no. 6, p. 933, 2016.
- [6] H. Yang, X. Liu, and I. Patras, "A simple and effective extrinsic calibration method of a camera and a single line scanning lidar," in *ICPR2012*. IEEE, 2012, pp. 1439–1442.
- [7] R. Ishikawa, T. Oishi, and K. Ikeuchi, "Lidar and camera calibration using motions estimated by sensor fusion odometry," in *2018 IROS*. IEEE, 2018, pp. 7342–7349.
- [8] K. Koide, S. Oishi, M. Yokozuka, and A. Banno, "General, single-shot, target-less, and automatic lidar-camera extrinsic calibration toolbox," *arXiv preprint arXiv:2302.05094*, 2023.
- [9] N. Schneider, F. Piewak, C. Stiller, and U. Franke, "Regnet: Multimodal sensor registration using deep neural networks," in *2017 IEEE intelligent vehicles symposium (IV)*. IEEE, 2017, pp. 1803–1810.
- [10] G. Iyer, R. K. Ram, J. K. Murthy, and K. M. Krishna, "Calibnet: Geometrically supervised extrinsic calibration using 3d spatial transformer networks," in *2018 IROS*. IEEE, 2018, pp. 1110–1117.
- [11] 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 *ECCV*, 2020.
- [12] G. Yan, Z. Liu, C. Wang, C. Shi, P. Wei, X. Cai, T. Ma, Z. Liu, Z. Zhong, Y. Liu, *et al.*, "OpenCalib: A multi-sensor calibration toolbox for autonomous driving," *Software Impacts*, vol. 14, p. 100393, 2022.
- [13] Q. Zhang and R. Pless, "Extrinsic calibration of a camera and laser range finder (improves camera calibration)," in *IROS*, vol. 3. IEEE, 2004, pp. 2301–2306.
- [14] E.-S. Kim and S.-Y. Park, "Extrinsic calibration of a camera-lidar multi sensor system using a planar chessboard," in *2019 ICUFN*. IEEE, 2019, pp. 89–91.
- [15] W. Wang, K. Sakurada, and N. Kawaguchi, "Reflectance intensity assisted automatic and accurate extrinsic calibration of 3d lidar and panoramic camera using a printed chessboard," *Remote Sensing*, vol. 9, no. 8, 2017.
- [16] Y. Li, W. Tian, X. You, K. Li, J. Yuan, X. Chen, and L. Pan, "Application of 3d-lidar & camera extrinsic calibration in urban rail transit," in *2020 ICITE*. IEEE, 2020, pp. 456–460.
- [17] Z. Chai, Y. Sun, and Z. Xiong, "A novel method for lidar camera calibration by plane fitting," in *2018 AIM*. IEEE, 2018, pp. 286–291.
- [18] G.-M. Lee, J.-H. Lee, and S.-Y. Park, "Calibration of vlp-16 lidar and multi-view cameras using a ball for 360 degree 3d color map acquisition," in *2017 MFI*. IEEE, 2017, pp. 64–69.
- [19] S. Mishra, P. R. Osteen, G. Pandey, and S. Saripalli, "Experimental evaluation of 3d-lidar camera extrinsic calibration," in *2020 IROS*. IEEE, 2020, pp. 9020–9026.
- [20] Q. Ye, L. Shu, and W. Zhang, "Extrinsic calibration of a monocular camera and a single line scanning lidar," in *2019 ICMA*. IEEE, 2019, pp. 1047–1054.
- [21] A. Geiger, F. Moosmann, Ö. Car, and B. Schuster, "Automatic camera and range sensor calibration using a single shot," in *2012 ICRA*. IEEE, 2012, pp. 3936–3943.
- [22] G. Pandey, J. R. McBride, S. Savarese, and R. M. Eustice, "Automatic targetless extrinsic calibration of a 3d lidar and camera by maximizing mutual information," in *AAAI*, 2012.
- [23] J. Levinson and S. Thrun, "Automatic online calibration of cameras and lasers," in *Robotics: science and systems*, vol. 2, no. 7. Berlin, Germany, 2013.
- [24] C. Yuan, X. Liu, X. Hong, and F. Zhang, "Pixel-level extrinsic self calibration of high resolution lidar and camera in targetless environments," *RA-L*, vol. 6, no. 4, pp. 7517–7524, 2021.
- [25] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superglue: Learning feature matching with graph neural networks," in *CVPR*, 2020, pp. 4938–4947.
- [26] Y. Zhu, C. Li, and Y. Zhang, "Online camera-lidar calibration with sensor semantic information," in *2020 ICRA*. IEEE, 2020, pp. 4970–4976.
- [27] P. Jiang, P. Osteen, and S. Saripalli, "Calibrating lidar and camera using semantic mutual information," *arXiv preprint arXiv:2104.12023*, 2021.
- [28] H.-J. Chien, R. Klette, N. Schneider, and U. Franke, "Visual odometry driven online calibration for monocular lidar-camera systems," in *2016 ICPR*. IEEE, 2016, pp. 2848–2853.
- [29] J. Shi, Z. Zhu, J. Zhang, R. Liu, Z. Wang, S. Chen, and H. Liu, "Calibrnn: Calibrating camera and lidar by recurrent convolutional neural network and geometric constraints," in *2020 IROS*. IEEE, 2020, pp. 10197–10202.
- [30] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, "Learning representations and generative models for 3d point clouds," in *International conference on machine learning*. PMLR, 2018, pp. 40–49.
- [31] H. Fan, H. Su, and L. J. Guibas, "A point set generation network for 3d object reconstruction from a single image," in *CVPR*, 2017, pp. 605–613.
- [32] A. Ranjan, T. Bolkart, S. Sanyal, and M. J. Black, "Generating 3d faces using convolutional mesh autoencoders," in *ECCV*, 2018, pp. 704–720.
- [33] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang, "Pixel2mesh: Generating 3d mesh models from single rgb images," in *ECCV*, 2018, pp. 52–67.
- [34] Y. Liao, S. Donne, and A. Geiger, "Deep marching cubes: Learning explicit surface representations," in *CVPR*, 2018, pp. 2916–2925.
- [35] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 165–174.
- [36] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space," in *CVPR*, 2019, pp. 4460–4470.
- [37] A. Yu, V. Ye, M. Tancik, and A. Kanazawa, "pixelnerf: Neural radiance fields from one or few images," in *CVPR*, 2021, pp. 4578–4587.
- [38] K. Deng, A. Liu, J.-Y. Zhu, and D. Ramanan, "Depth-supervised nerf: Fewer views and faster training for free," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12882–12891.
- [39] 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 *ICCV*, 2021, pp. 5855–5864.
- [40] Y. Jeong, S. Ahn, C. Choy, A. Anandkumar, M. Cho, and J. Park, "Self-calibrating neural radiance fields," in *ICCV*, 2021, pp. 5846–5854.
- [41] C.-H. Lin, W.-C. Ma, A. Torralba, and S. Lucey, "Barf: Bundle-adjusting neural radiance fields," in *ICCV*, 2021, pp. 5741–5751.
- [42] 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.
- [43] K. Rematas, A. Liu, P. P. Srinivasan, J. T. Barron, A. Tagliasacchi, T. Funkhouser, and V. Ferrari, "Urban radiance fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12932–12942.
- [44] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.
- [45] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Sensor fusion IV: control paradigms and data structures*, vol. 1611. Spie, 1992, pp. 586–606.
- [46] Y. Chen and G. Medioni, "Object modelling by registration of multiple range images," *Image and vision computing*, vol. 10, no. 3, pp. 145–155, 1992.