Di-NeRF: Distributed NeRF for Collaborative Learning with Relative Pose Refinement

Mahboubeh Asadi[1], Kourosh Zareinia[1], Sajad Saeedi[1]

***Abstract*—Collaborative mapping of unknown environments can be done faster and more robustly than a single robot. However, a collaborative approach requires a distributed paradigm to be scalable and deal with communication issues. This work presents a fully distributed algorithm enabling a group of robots to collectively optimize the parameters of a Neural Radiance Field (NeRF). The algorithm involves the communication of each robot's trained NeRF parameters over a mesh network, where each robot trains its NeRF and has access to its own visual data only. Additionally, the relative poses of all robots are jointly optimized alongside the model parameters, enabling mapping with less accurate relative camera poses. We show that multi-robot systems can benefit from differentiable and robust 3D reconstruction optimized from multiple NeRFs. Experiments on real-world and synthetic data demonstrate the efficiency of the proposed algorithm. See the website of the project for videos of the experiments and supplementary material[2].**

***Index Terms*—Distributed Robot Systems; Mapping; Multi-Robot SLAM**

I. Introduction

THERE is an increasing demand for robots to collaborate on complex tasks, such as mapping an unknown environment. Centralized approaches for the coordination of the robots or processing data face scalability issues, vulnerability to central node failures, and the risk of communication blackouts. A true collaborative robotic system needs to work in a distributed manner [1]. Further, building a high-quality representation in collaborative mapping is needed to make an informed decision. The representations also need to be compact for easy sharing. Neural Radiance Field (NeRF) [2] enables such a representation for a single robot, leveraging advancements in neural networks. Extending the capabilities of NeRF to facilitate multi-robot NeRF in a distributed manner emerges as a natural progression. This extension allows for a reliable and fast collaborative development of high-quality representations of unknown environments (e.g. in search & rescue, surveillance, and monitoring), without a central node. Performing distributed NeRF requires addressing several problems inherent to multi-robot systems [3]. Firstly, determining the relative poses of robots, preferably without requiring them to rendezvous. This is needed to enable the fusions of the local maps (i.e. maps of individual robots) into a global map.

Manuscript received: August, 03, 2024; Accepted September, 17, 2024.

This paper was recommended for publication by Editor Javier Civera upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

[1]All authors are with Toronto Metropolitan University, Canada.

Digital Object Identifier (DOI): see top of this page.

[2]https://sites.google.com/view/di-nerf/home

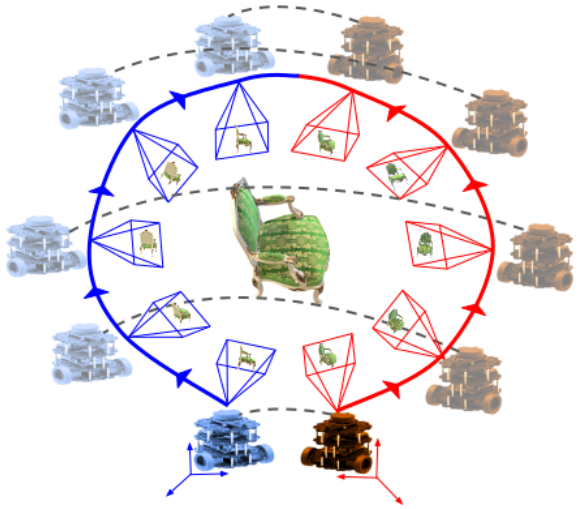

Figure 1. Di-NeRF allows robots to cooperatively optimize local copies of a neural network model without explicitly sharing visual data. In this figure, two robots use Di-NeRF to cooperatively optimize a unified NeRF. Each robot only sees part of the chair, and robots do not know their relative poses. The robots communicate over a wireless network (gray dashed lines) to cooperatively optimize the final network and relative poses.

The relative poses can be determined either via line-of-sight rendezvous [4], which necessitates motion coordination, or by identifying overlaps within local maps [5], requiring additional processing. In large-scale applications, avoiding line-of-sight rendezvous is preferred to prevent further planning constraints. The second challenge is deciding what information to share among robots. While sharing unprocessed raw visual data is resource-intensive, depending on the architecture, sharing compact neural models is a more viable option. However, sharing maps introduces complexity to the subsequent challenge. The third challenge is merging representations, requiring the development of a method to unify local neural maps into a global map. Merging neural maps is more intricate than merging geometric representations. There are many methods for distributed/centralized geometric multi-robot mapping and localization [4], [6]. For learned methods, federated [7] and distributed learning [1] are common for overcoming the issues associated with centralized learning. In these methods, the training process is decentralized. Each robot performs its training, and the results are aggregated in a central node [7]. The need for a central node still constrains the scalability of such methods. A recent work using multiple agents for NeRF is Block-NeRF [8], composed of independent local maps, i.e. NeRF block. In Block-NeRF, the relative poses of the blocks are known to a degree and the task is done in a centralized

manner. For large-scale aerial NeRF mapping, Mega-NeRF [9] partitions training images into NeRF submodules for parallel training. Another closely related work is DiNNO [10], which is a distributed algorithm enabling robots to collaboratively optimize deep neural networks over a mesh network without sharing raw data. In this paper, we present Di-NeRF (Distributed NeRF), an algorithm that builds on the Consensus Alternating Direction Method of Multipliers (C-ADMM) as a versatile distributed optimization method for multi-robot systems. In Di-NeRF, each robot starts building its NeRF by relying on its own data i.e. images and local camera poses from a Structure-from-Motion (SfM) algorithm. By integrating NeRFs from other robots, each robot will build a global NeRF model as if the robot has gathered all data and processed them. The advantages of Di-NeRF are that the robots do not know their exact initial poses, as illustrated graphically in Fig. 1. In Di-NeRF, robots alternate between local optimization of an objective function and communication of intermediate network weights over the wireless network. Di-NeRF can consider different communication graphs (e.g. fully connected, circular, and ring connectivity). The key contributions of Di-NeRF include i) developing fully distributed optimization for 3D reconstruction with RGB images as input, and a NeRF as the backend, enabling fusing NeRFs in training not the rendering process, and ii) optimization of the relative poses of the robots, removing the need to know the prior relative poses accurately. In this work, we make these assumptions: i) a reliable estimate of the agent's trajectory in its local frame is available; ii) there is sufficient overlap between the agents' viewpoints, allowing optimization of the relative poses.

The structure of the paper is as follows. In Sec. II, related work is presented. In Sec. III, we present Di-NeRF, followed by results and conclusions in Sec. IV and V.

II. Literature Review

Multi-robot localization and mapping algorithms utilize various representations such as sparse landmarks [11], dense geometric maps [12], object classes [13], and semantics [14]. With the advent of neural radiance field [2] and its variants [15]–[18], there are efforts to use neural maps for collaborative localization/mapping [8], [19].

Emerging radiance fields, including non-neural representations such as Plenoxels [20], and neural representations, such as NeRF [2], [21] have revolutionized localization and mapping algorithms. Nice SLAM [22], iMAP [23], NeRF-SLAM [24], and [25] are using such representations. The early versions of NeRF representations [18], required pose information often generated via Structure-from-Motion (SfM) algorithms, e.g. COLMAP [26]. This limitation was later addressed in [15], [17], [27], by jointly optimizing camera poses and the scene. It was also shown that a camera view can be localized in a NeRF map, as shown in iNeRF [28].

Extending NeRF to multi-robot scenarios has also been proposed in Block-NeRF [8], NeRFuser [29], and NeRFusion [30]. These methods use blending techniques to render an image, often done centralized, performing inference on multiple NeRF models. Furthermore, it is assumed that there is prior knowledge about the relative pose of the robots. A similar research is federated learning [19], [31], where the training is done in a decentralized optimization, and each node learns a common NeRF in parallel. Then the weights are transferred to a server for aggregation. A memory-efficient multi-robot SLAM approach based on multi-block ADMM is proposed in [32]. In DiNNO [10], a distributed algorithm for collaborative learning of a neural model is presented, demonstrating a range of experiments including multi-agent RL, MNIST classification, and 2D neural implicit mapping with known relative poses. The core of the algorithm is a distributed optimization method known as C-ADMM [1]. To establish a shared reference frame, [33] proposes a method for incremental inference from arbitrary poses. Additionally, im2nerf [16], [34] enables neural radiance field prediction from a single image, addressing view sparsity.

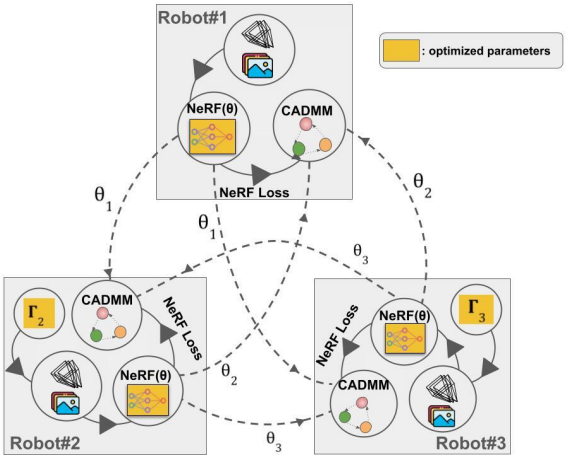

Figure 2. Three connected robots share their local NeRF to build a global NeRF. Each robot adjusts its weight compared to receiving weights using C-ADMM, and all robots except robot R^g (Robot#1, the robot whose coordinate is considered as the global coordinate) optimize for the relative pose T_i^g and depth distortion parameters.

III. Distributed Neural Radiance Field (Di-NeRF)

Here the problem is defined followed by the proposed method for relative pose refinement and distributed NeRF. The overall pipeline for Di-NeRF is shown in Fig. 2. Di-NeRF is a fully distributed algorithm for 3D reconstruction using RGB images. With Di-NeRF, each robot has its local coordinate and only shares the learned models to achieve a consensus while optimizing for the relative poses.

A. Problem Statement

Assume N robots connected to each other with a communication graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consisting of vertices $\mathcal{V} = \{1, ..., N\}$ and edges $\mathcal{E}$, on which pairwise communication can occur. Given a set of images $\boldsymbol{I}_i$ captured by robot i, $i = 1, ..., N$, from a scene with the associated local camera poses, $\boldsymbol{\pi}_i$, the goal of Di-NeRF is to build a scene representation, θ, that enables all the robots to generate realistic images from novel viewpoints. Here, NeRF [18] is employed as the model for 3D reconstruction, θ. Since the poses of each robot are defined in a local coordinate system, it is necessary to perform relative

pose optimization to build a consistent model that accurately represents the entire scene. The problem is expressed as:

$$\theta^*, \boldsymbol{T}^* = \underset{\theta, \mathbf{T}}{\operatorname{argmin}} \sum_{i \in \mathcal{V}} L_i(\theta, T_i^g | \boldsymbol{I}_i, \boldsymbol{\pi}_i), \ \boldsymbol{T} = \{T_1^g, .., T_N^g\}, \quad (1)$$

where L_i represents the NeRF loss function corresponding to the robot i. The relative poses are denoted by $\boldsymbol{T}$, where the pose of the robot i relative to the global coordinate system is represented by $T_i^g = [R^g|t^g]$ within the space of $SE(3)$. Superscript g stands for the global coordinate. Without loss of generality, the first robot coordinate is selected as the global frame (T_1^g is the identity matrix). If this robot fails, another robot is selected as the global frame. The model θ is shared among all robots and needs to be determined collectively, indicated by the summation in Eq. (1). Other variables with subscript i; *i.e.* T_i^g, $\boldsymbol{I}_i$, and $\boldsymbol{\pi}_i$; are local/internal to robot i. The model comprises fully connected neural network layers, described in Sec. IV-H. It learns to generate a color vector $\boldsymbol{c} = (r, g, b)$ and volume density σ for each ray along a 3D location $\boldsymbol{x} = (x, y, z)$ and 2D viewing direction $\boldsymbol{d}$. Here, we use a pinhole camera model, assuming equal focal lengths for x and y directions. The camera ray that starts from camera origin $\boldsymbol{o}$ and passes through each pixel $\boldsymbol{p} = (u, v)$ is in the local coordinate of each robot and should be transferred to the global coordinate system using relative camera poses. In a local camera pose, $\pi_i = [R|t]$, the camera origin is $\boldsymbol{o} = t$ and the ray direction $\boldsymbol{d}$ is as follows [2], [18]:

$$\boldsymbol{d} = R \begin{pmatrix} \frac{u - W/2}{f} & \frac{-v + H/2}{f} & -1 \end{pmatrix}^\top , \quad (2)$$

where f is the focal length and H and W are the image height and width. By using the relative pose $T_i^g = [R^g|t^g]$, the camera pose in the global coordinate will be $\pi_i^g = T_i^g \pi_i = [R^g R | R^g t + t^g]$ which positions the camera origin at $R^g t + t^g$ and defines ray direction $R^g \boldsymbol{d}$. Since the relative poses, $\boldsymbol{T}_i^g$, are not known exactly, the model θ_i and relative poses must be optimized jointly. The optimization can be central, but a distributed approach is preferred for real-world use.

B. Relative Camera Pose Optimization

In multi-robot NeRF, robots do not send raw data over the wireless network. Therefore, using an SfM pipeline to calculate the relative poses for all robots is not easy. Here, the camera poses of each robot are expressed based on a local coordinate system, and gradually all poses transfer to the robot R^g coordinate system (throughout this paper, the robot R^g will be the arbitrary reference agent.). The optimization of the relative poses is performed jointly with the model parameters. Relative camera poses of multi-robot systems can be expressed as a camera-to-world transformation matrix (similar to each robot's local camera pose) $T_i^g = [R^g|t^g] \in SE(3)$, where $R^g \in SO(3)$ and $t^g \in \mathbb{R}^3$ show camera rotation and translation, respectively [15]. Optimizing the translation vector t^g involves the designation of trainable parameters, given its definition in Euclidean space. For camera rotation, which is in $SO(3)$ space, the axis-angle representation is chosen: $\phi := \alpha\omega, \phi \in \mathbb{R}^3$, where ω and α are a normalized rotation axis and a rotation angle, respectively. The rotation matrix R^g can be expressed from Rodrigues' formula:

$$R^g = I + \frac{\sin(\alpha)}{\alpha} \phi^\wedge + \frac{1 - \cos(\alpha)}{\alpha^2} (\phi^\wedge)^2 , \quad (3)$$

where $(.)^\wedge$ is the skew operator that converts a vector ϕ to a skew-symmetric matrix. The relative pose for each robot i is optimized by trainable parameters ϕ_i and t_i.

Training NeRF and the relative camera poses jointly adds another degree of ambiguity to the problem, and causes the NeRF optimization to converge to local minima, especially for large relative poses. To avoid local minima, we use monocular depth estimation [17]. However, the depth maps suffer scale and shift distortions across frames, making mono-depth maps not multi-view consistent. To utilize mono-depth effectively, we explicitly optimize scale and shift parameters during NeRF training by penalizing the difference between rendered depth and mono-depth. For each image I_i^m (out of M_i images) of robot i, we generate mono-depth sequence $D_i^m, m = 0, ..., M_i - 1$ from input images with an off-the-shelf monocular depth network, i.e., DPT [35]. Two linear transformation parameters $\boldsymbol{\psi}_i^m = (\alpha_i^m, \beta_i^m), m = 0 \dots M_i - 1$ for each mono-depth map as scale and shift factors can recover a multi-view consistent depth map $D_i^{m*} = \alpha_i^m D_i^m + \beta_i^m$, for each image m in each robot i. This joint optimization of distortion parameters and NeRF is achieved by adding a depth loss $L_i^{depth} = \sum_m ||D_i^{m*} - \hat{D}_i^m||$ to the RGB loss of NeRF [18], $L_i^{RGB} = \sum_m ||I_i^m - \hat{I}_i^m||$. Here $\hat{D}_i^m$ and $\hat{I}_i^m$ are NeRF-rendered depth/RGB images, described in [17], [18]. Thus, the local NeRF loss for each robot i is defined as:

$$L_i = L_i^{RGB} + \gamma L_i^{depth} , \quad (4)$$

where γ is a weighting factor for depth loss term. With depth information, Eq. (1) is rewritten as:

$$\theta^*, \boldsymbol{T}^*, \boldsymbol{\psi}^* = \underset{\theta, \mathbf{T}, \boldsymbol{\psi}}{\operatorname{argmin}} \sum_{i \in \mathcal{V}} L_i(\theta, T_i^g, \boldsymbol{\psi}_i \mid \boldsymbol{I}_i, \boldsymbol{\pi}_i) . \quad (5)$$

$\boldsymbol{\psi_i} = \{\psi_i^m\}$ is the set of distortion parameters for robot i, and $\boldsymbol{\psi} = \{\boldsymbol{\psi}_i\}$ is the set of all distortion parameters of all robots.

C. Distributed Formulation

In the problem presented in Eq. (5), each agent i has its local cost function L_i, indicating that the optimization must be done in a distributed manner on each agent. However, θ is a global variable and is shared among all agents. This necessitates that the agents should achieve a consensus on the optimal value of θ. In addition to the global variable θ, agents have their local variables, such as T_i^g. To express that the optimization is distributed, Eq. (5) is rewritten as

$$\begin{aligned} \theta^*, \boldsymbol{T}^*, \boldsymbol{\psi}^* = \underset{\theta, \boldsymbol{T}, \boldsymbol{\psi}}{\operatorname{argmin}} \sum_{i \in \mathcal{V}} L_i(\underbrace{\theta_i, T_i^g, \boldsymbol{\psi}_i}_{\Gamma_i} | \boldsymbol{I}_i, \boldsymbol{\pi}_i) \ . \\ \text{s.t.} \quad \theta_i = z_{ij}, \quad \forall j \in \mathcal{N}_i \end{aligned} \quad (6)$$

Each agent solves for its version of the global variable θ, θ_i, a relaxed local variable or primal variable. Assuming $\mathcal{N}_i$ is the set of neighbors of robot i, to achieve a consensus

between agents i and $\forall j \in \mathcal{N}_i$, an auxiliary variable z_{ij} is introduced. This is also known as the 'complicating variable' which ensures that the agents achieve a consensus via 'complicating constraints': i.e. $\theta_i = z_{ij}$ and $\theta_j = z_{ij}$. By defining $\Gamma_i = \{\theta_i, T_i^g, \boldsymbol{\psi}_i\}$ for notation brevity, Eq. (6) is then solved by introducing augmented Lagrangian and redefining the cost function to $\mathcal{L}_i$ – for brevity, $\boldsymbol{I}_i$ and $\boldsymbol{\pi}_i$ were omitted:

$$\mathcal{L}_i(\Gamma_i) = L_i(\Gamma_i) + (\theta_i - z_{ij})^\top y_i + \frac{\rho}{2} \sum_{j \in \mathcal{N}_i} \|\theta_i - z_{ij}\|_2^2 \ , \quad (7)$$

where y_i is the Lagrangian multiplier or dual variable and ρ is penalty factor. The last two terms enforce that constraints are satisfied, but the penalty function ensures that around the optimal point, the objective function is quadratic.

There are various techniques to optimize Eq. (7), such as auxiliary problem principle (APP) [36] and alternating direction method of multipliers (ADMM) [37]. ADMM has improved convergence properties [37]. We use a version of ADMM that is suitable for distributed systems, known as consensus ADMM (C-ADMM) [38] described next.

D. Distributed Optimization of NeRF

The optimization through ADMM alternates between variables, and C-ADMM allows the agents to share the intermediate optimized variables to achieve a consensus. The variables are $\Gamma_i = \{\theta_i, T_i^g, \boldsymbol{\psi}_i\}$, z_{ij}, and y_i. Optimization of z_{ij} has a closed-form solution, see the website of the project, (superscript (k) denotes the step number in updating the variable):

$$z_{ij}^{(k+1)} = \frac{1}{N} \sum \theta_i^{(k)} := \bar{\theta}^{(k)} \ , \quad (8)$$

therefore, the optimization step alternates between minimizing the updated local objective function with respect to primal variables ($\Gamma_i = \{\theta_i, T_i^g, \boldsymbol{\psi}_i\}$) and maximizing the updated local objective function with respect to the dual variable:

- θ_i-minimization

$$\min_{\Gamma_i^{(k)}} L_i(\Gamma_i^{(k)}) + \theta_i^{(k)\top} y_i^{(k)} + \frac{\rho}{2} \sum_{j \in \mathcal{N}_i} \left\| \theta_i^{(k)} - \bar{\theta}^{(k)} \right\|_2^2, \quad (9)$$

- y_i-update (or dual variable update):

$$y_i^{(k)} \leftarrow y_i^{(k-1)} + \rho \sum_{j \in \mathcal{N}_i} (\theta_i^{(k)} - \bar{\theta}^{(k)}), \quad (10)$$

Alg. 1 summarizes the optimization procedures for local primal, dual variables, and relative poses for each robot. In line 1, The initial values for the NeRF parameters $\theta_i^{(0)}$, the dual variable $y_i^{(0)}$, and the depth distortion parameters $\boldsymbol{\psi}_i^{(0)}$ are set to zero. The initial value for the relative poses - three rotation angles R and three translation displacements t - are explained in Sec. IV. The first robot R^g undergoes the NeRF training process (ln 6-7), communicates the learned weights with other robots (ln 10-11), and updates the dual variable (ln 12) for 200 steps. The remaining robots engage in collaborative optimization, simultaneously refining NeRF weights and adjusting relative camera poses and depth distortion parameters in a distributed training framework based on the weights of robot R^g (lines 8-9 for NeRF training, lines 10-12 for communication and dual update). This iterative communication and training process persists until uniformity is achieved in both the weights and optimized relative camera poses across all robots. The relative pose between robot i's coordinate system and robot R^g's coordinate system $T_i^g \in SE(3)$ is optimized jointly in the distributed training (ln 9).

Algorithm 1 Di-NeRF Algorithm

1: **Initialization:** $k \leftarrow 0, \theta_i^{(0)} \in \mathbb{R}^n, y_i^{(0)} = 0, T_i^g \in SE(3)$, $\boldsymbol{\psi_i} \in \mathbb{R}^m$
2: **Internal variables:** $y_i^{(k)}$
3: **Public variables:** $\mathcal{Q}_i^{(k)} = \theta_i^{(k)}$
4: **while** stopping condition not satisfied **do**
5: **for** i in $\mathcal{V}$ **do**
6: **if** robot i is R^g **then**
7: $\theta_i^{(k+1)} = \underset{\theta_i}{\operatorname{argmin}} \{L_i(\theta_i) + \theta_i^\top y_i^{(k)} + \rho \sum_{j \in \mathcal{N}_i} \left\| \theta_i - \bar{\theta}^{(k)} \right\|_2^2 \}$
8: **else**
9: $\Gamma_i^{(k+1)} = \underset{\Gamma}{\operatorname{argmin}} \{L_i(\Gamma_i) + \theta_i^\top y_i^{(k)} + \rho \sum_{j \in \mathcal{N}_i} \left\| \theta_i - \bar{\theta}^{(k)} \right\|_2^2 \}$
10: **Communicate** $\mathcal{Q}_i^{(k)}$ to all j in $\mathcal{N}_i$
11: **Receive** $\mathcal{Q}_j^{(k)}$ from all j in $\mathcal{N}_i$
12: $y_i^{(k+1)} = y_i^{(k)} + \rho \sum_{j \in \mathcal{N}_i} (\theta_i^{(k+1)} - \theta_j^{(k+1)})$
13: $k \leftarrow k + 1$

E. Convergence Properties

The convergence of C-ADMM methods typically requires the dual variables' sum to converge to zero, a condition challenging in unreliable networks. Moreover, the nonlinearity and nonconvexity of neural networks, especially NeRF, preclude guaranteed global solutions and linear rates [10], [38], [39]. Despite these issues, Alg. 1 has proven effective in practice for distributed NeRF training, as shown in Sec. IV, converging to solutions equivalent to when a robot is doing all processing.

IV. Experiments

This section presents the experimental results on synthetic and real-world datasets, including 1) comparing Di-NeRF and a baseline, 2) examining the impact of the number of robots and communication graphs, 3) analyzing the impact of the amount of overlap between the robots' trajectories on convergence, 4) evaluation using Waymo Block-NeRF dataset [8], 5) convergence of the relative pose with respect to different initial values and ground truth relative pose, 6) examining no overlap setup, 7) assessing reference robot failure, and 8) an ablation study. The architecture of NeRF is based on the instant neural graphics primitives (iNGP) [18], described in Sec IV-H. For all comparisons, the '*baseline*' is training one model with all images, with known poses, equivalent to the standard NeRF training of iNGP. For the relative poses, the initial values for both the translational and rotational components were set to zero, except for convergence analyses (Sec. IV-E and Sec. IV-I, Fig. 14). The ground truth relative poses assume each robot's

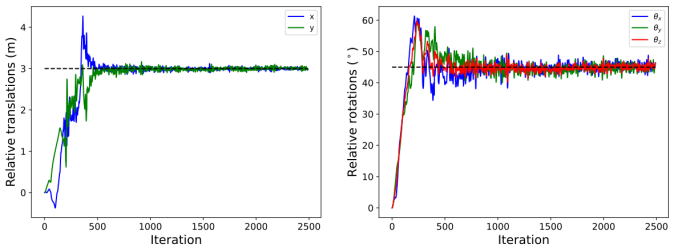

Figure 3. Optimizing the relative poses for two robots for the chair sequence [2]. The relative translation and rotation estimates are plotted. The desired translation/rotation is marked by a dashed line.

local origin is its first frame, chosen for the largest relative pose. In reality, any image in the robot's trajectory could serve as the origin.

A. Di-NeRF vs Baseline NeRF on Synthetic Dataset

In this experiment, we use Di-NeRF to learn a model in a distributed manner and compare the results with the baseline, iNGP [18]. In this experiment, for Di-NeRF, the robots do not have access to a global coordinate frame, so the relative poses are unknown. We also compare Di-NeRF with PixelNeRF [16]. PixelNeRF works with one/fewer images. So part of the data from both robots that produce the best results is considered for PixelNeRF. Here, the Chair, Hotdog, and Lego sequences from the synthetic NeRF dataset [2] are used. We also use the long sequence of Barn from the real-world Tanks and Temples Benchmark [40]. The configurations for all experiments and the baseline are the same, described in Sec. IV-H. The baseline contains 100-150 images for synthetic sequences and 300 for the Barn sequence. For multiple robots, images are divided with controlled overlaps. All experiments follow official preprocessing and train/test splits.

The metrics used to compare the performance of Di-NeRF with the baseline solution are the average rendering i) PSNRs, ii) SSIM, and iii) relative pose accuracy (in degrees and cm). The pose accuracy is done against the ground truth. For the synthetic sequences, two robots split the dataset, with robot R^1 observing the chair's front and robot R^2 the back, without trajectory overlap. This pattern applies to other sequences. The dataset is divided, and COLMAP estimates poses for each segment, with different coordinate systems. Di-NeRF calculates the relative pose. Fig. 3 shows the convergence of the relative pose for the chair sequence.

Fig. 4 shows the qualitative results. It illustrates the Chair sequence, comparing individual training, PixelNeRF, the baseline, and Di-NeRF for two robots. Since robot R^1 has no image from the back of the chair, it fails to reconstruct the back. Similarly, robot R^2 is unable to accurately reconstruct the front of the chair. This issue is evident in the first column of Fig. 4. The second column is PixelNeRF. The third column shows the baseline. Di-NeRF enables all robots to render the whole scene, see Fig. 4-(last column). Fig. 5 presents a comparison of PSNR and L2 loss convergences for Di-NeRF and the baseline training. The convergences exhibit nearly identical behavior.

Table I shows the quantitative results for PixelNeRF, a centralized method, the baseline, and Di-NeRF. The metrics separated by the / sign show the metric values for each robot. The *centralized* column refers to sending all local images and local poses to a server for training a NeRF model. The relative poses are calculated from RootSIFT features of a set of keyframes from each local set. The global poses then are calculated to train a model. Di-NeRF achieves similar quality compared to the baseline and centralized methods. For iNGP and PixelNeRF, image poses are in the global frame. Table I shows that while PixelNeRF is designed for a few input images, Di-NeRF outperforms it with results matching centralized and baseline approaches.

B. Di-NeRF vs Baseline NeRF on Real-world Dataset

This experiment shows the performance of Di-NeRF not only on real-world data, but also when the number of robots increases from two to five. This scenario applies to large-scale environments. Here, the Barn sequence [40] is used. There are 300 images in this sequence. Images are divided by the number of robots such that there is no common image in the local data of each robot, and the robots' trajectories have zero overlaps. Fig. 6 shows the rendered images from each robot's distributedly trained NeRF. Each robot can render a view from any point (e.g. Robot R^3 has never seen the large bay doors but renders an image when requested, as seen in Fig. 6 (b)).

Next, the number of robots in this experiment is altered. In Fig. 7, the trajectory for each robot is shown alongside the rendered images for each robot after collaborative training is completed. Each color represents the trajectory of a different robot. Table II presents the quantitative results. The rendering qualities and accuracy of relative camera pose are compared to the baseline method. With Di-NeRF, the outcomes closely resemble those achieved with the baseline method. In Table III, we experiment with different connectivity graphs of robots. Five robots are used with the Barn sequence. In the star connection, R^1 is in the center. In Table III, the average results for 5 robots are shown.

C. Trajectory Overlap Analysis

In this experiment, the sensitivity of the convergence of the relative poses with respect to the overlaps between the views of the robots is analyzed. For this experiment, two robots are considered. There are no common images between the robots, and the overlaps are only in the trajectories. Fig. 8 shows the top-down view of the overlaps, ranging from 5% to 40%, i.e. [5, 10, 20, 30, 40]%. The overlaps are determined using x and y coordinates of camera poses, assuming the poses are known in a coordinate frame. To divide the views among N robots, first, the $x-y$ plane containing the desired object (the Chair, in this case) is divided into N sectors, centered on the object. This will generate a 0% overlap for N robots, which then is extended to increase the overlap, as shown in Fig. 8.

For two robots, once the overlaps are determined, the poses of robot R^2 are manipulated to create a relative displacement with respect to R^1, by translating them for 3m along each axis and rotating them for 30° around each axis. Then optimization is performed, assuming an initial estimate for the relative pose with zero for each translation component and each rotation.

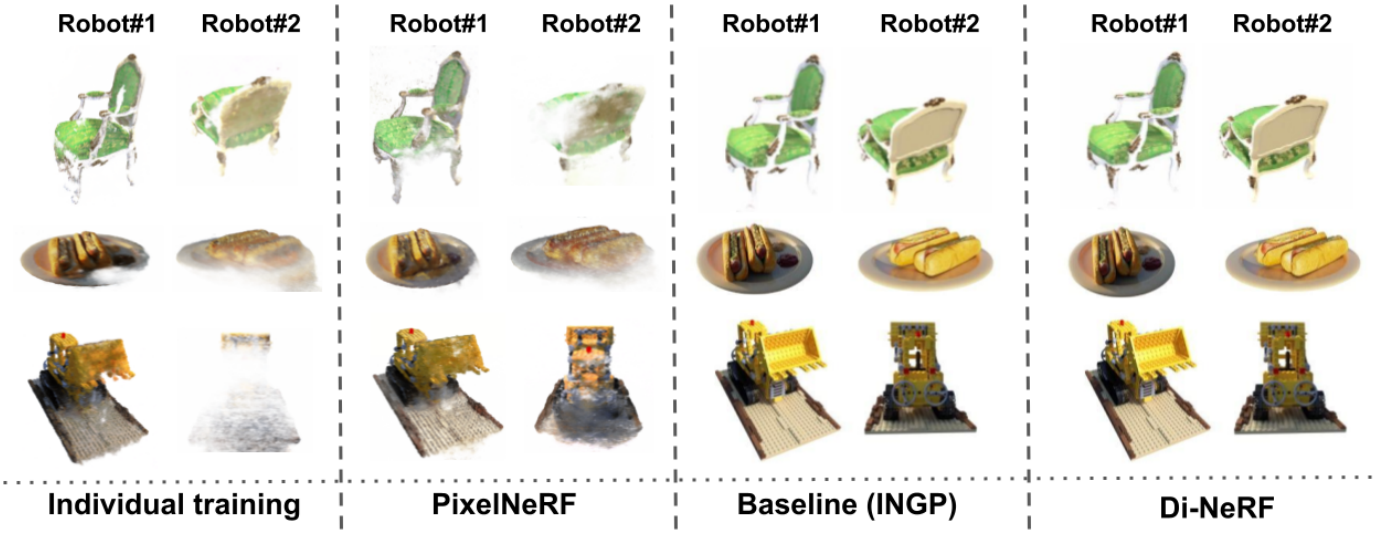

Figure 4. Two robots learn the scene collaboratively where each robot only sees a part of the scene. Individual training results (using iNGP and PixelNeRF) have poor quality for some areas, but Di-NeRF maintains good quality, similar to the baseline. In each column, the left and right images are for robots R^1 and R^2, respectively.

Table I

IMAGE AND RELATIVE CAMERA POSE METRIC PERFORMANCES FOR TWO ROBOTS. PSNR AND SSIM ARE REPORTED FOR EACH ROBOT. THE RELATIVE POSE ACCURACY IS SHOWN FOR R^2. R^1 IS THE GLOBAL FRAME. LOWER/HIGHER BETTER IS INDICATED BY ↓/↑ RESPECTIVELY.

Dataset	T_i^g	Di-NeRF				Centralized				Baseline		PixelNeRF	
		PSNR↑	SSIM↑	$\delta R^\circ \downarrow$	$\delta t(cm) \downarrow$	PSNR	SSIM	$\delta\ R^\circ$	$\delta t(cm)$	PSNR	SSIM	PSNR	SSIM
Chair	Known	31.93/32.05	0.941/0.945	NA	NA	33.02	0.952	NA	NA	33.02	0.952	24.61	0.682
	Unknown	32.11/31.89	0.933/0.939	1.2	2.02	32.63	0.923	0.11	0.86				
Hotdog	Known	31.02/31.10	0.951/0.952	NA	NA	32.63	0.963	NA	NA	32.63	0.963	24.11	0.695
	Unknown	30.25/30.50	0.931/0.917	1.30	1.78	32.21	0.958	0.54	0.81				
Lego	Known	29.00/28.91	0.936/0.937	NA	NA	29.67	0.941	NA	NA	29.67	0.941	25.07	0.706
	Unknown	29.11/29.19	0.932/0.938	1.42	1.22	29.71	0.943	0.51	0.76				
Barn	Known	29.01/28.79	0.884/0.886	NA	NA	29.94	0.894	NA	NA	29.94	0.894	24.16	0.659
	Unknown	29.09/29.23	0.854/0.843	1.42	1.70	29.66	0.891	0.59	0.94				

Table II

IMAGE AND RELATIVE CAMERA POSE METRIC PERFORMANCES FOR DIFFERENT NUMBERS OF ROBOTS FOR THE BARN SEQUENCE.

	Di-NeRF	Baseline	Di-NeRF	Baseline	Relative Camera Pose Error	
#Robots	PSNR ↑		SSIM ↑		$\delta R^\circ \downarrow$	$\delta t(cm) \downarrow$
2	29.21/29.11	29.94	0.886/0.839	0.894	1.34	1.76
3	29.13/29.21/29.08		0.836/0.878/0.873		2.12/1.89	1.05/1.82
4	29.35/29.39/29.33/29.17		0.856/0.838/0.828/0.856		1.35/1.90/2.10	1.83/2.00/3.01
5	29.67/29.44/29.15/29.45/29.41		0.845/0.878/0.852/0.887/0.869		2.70/3.20/3.10/3.00	2.11/2.65/2.39/2.02

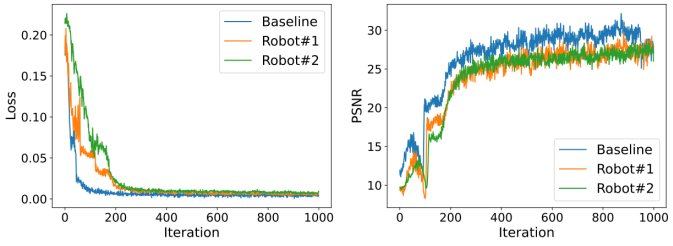

Figure 5. NeRF validation Loss and PSNR for the chair sequence.

Table III

MEAN VALIDATION RESULT OF DI-NERF FOR 5 ROBOTS, WITH DIFFERENT CONNECTIVITIES, FOR THE BARN SEQUENCE. DI-NERF WORKS WELL EVEN IF THE NETWORK OF ROBOTS IS NOT FULLY CONNECTED. LOWER/HIGHER BETTER IS INDICATED BY ↓/↑ RESPECTIVELY.

Communication	PSNR↑	SSIM↑	$\delta R^\circ \downarrow$	$\delta t(cm) \downarrow$
Fully Connected	29.42	0.866	3.00	2.29
Ring	29.19	0.835	4.56	2.33
Star	29.16	0.862	4.91	3.02
Line	29.10	0.853	5.01	2.97

Fig. 9 shows the convergence results for different overlaps, for 2000 steps of Di-NeRF training. The ground-truth relative poses are marked with GT. For each overlap, the plot shows the average estimates of translation and rotation components. The training configurations are the same for all overlaps. It is observed that large overlaps improve the convergence.

D. Waymo dataset - Unbounded Scenes

The San Francisco Mission Bay dataset (Waymo) [8] captures 12,000 images over 100 seconds along a 1.08 km route, using 12 cameras mounted on a car in an urban environment with dynamic objects and reflective surfaces. A 286m data segment with 233 images from a single roof-mounted camera providing a full surround view was selected for this experiment (Fig. 10). The sequence was divided into six segments to

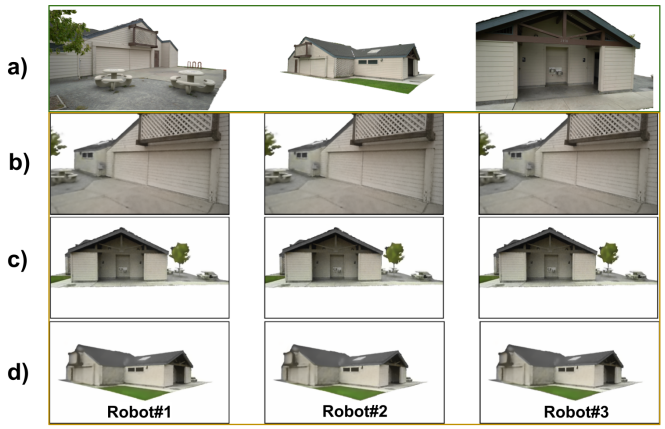

Figure 6. (a) sample images, from the Barn sequence, provided to each robot; (b, c, d) sample images generated by Di-NeRF, once the robots collaboratively learn the scene. The images are strategically chosen—each is only visible in the raw data of a specific robot but can be rendered by all robots.

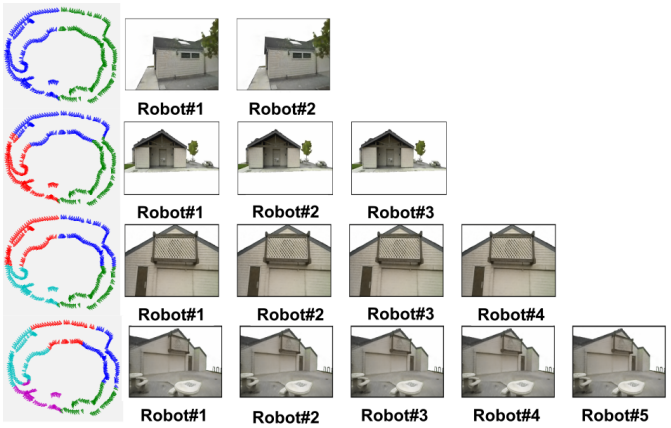

Figure 7. Di-NeRF for different numbers of robots (all fully connected); (left) the allocation of frames and poses to the robots; (right) none of the robots

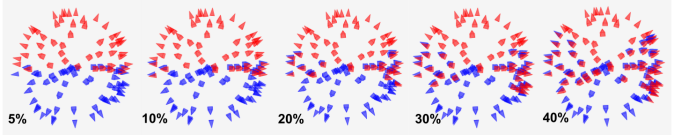

Figure 8. Segmenting the Chair sequence with different overlaps.

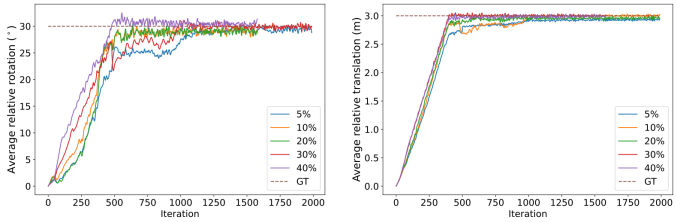

Figure 9. Optimizing the relative poses for two robots with varying overlaps.

resemble a multi-agent setup. Initial relative poses are offset by 1m from actual values. For larger offsets, the algorithm fails due to lack of overlap.

Fig. 11 shows the rendering result for the baseline and Di-NeRF. All robots can render a view that they have never seen directly. The average values of PSNR and SSIM are 25.60 and 0.831 over 6 robots and 25.39 and 0.848 for the baseline. Table IV shows detailed results. We divided the Waymo dataset into segments to benchmark Di-NeRF against Block-NeRF [8]. Table V shows the results. The objective of this experiment is not to surpass Block-NeRF [8], which is centralized. The goal is to achieve comparable results between distributed and

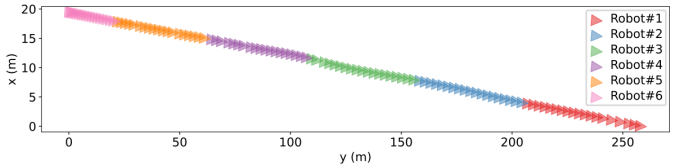

Figure 10. A sequence of a Waymo trajectory is divided into six segments, i.e. robots, and used to evaluate Di-NeRF.

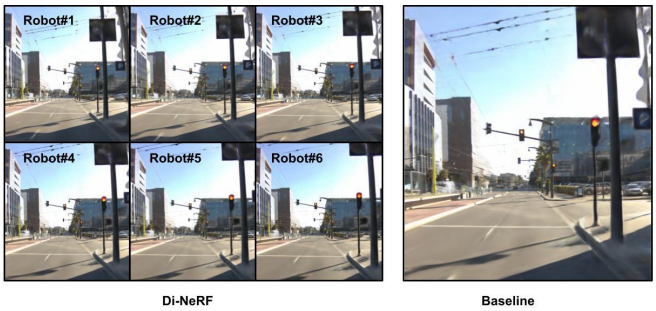

Figure 11. Rendered images from Di-NeRF on the Waymo dataset, all six robots demonstrated the capability to render similar images with quality comparable to that achieved through centralized processing.

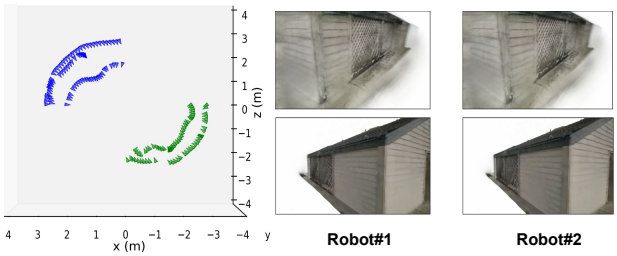

Figure 12. (Left) Trajectories of two robots from the Barn sequence showing no overlaps in views. Rendered views after Di-NeRF training: (Top Right) with unknown relative pose; (Bottom Right) with a known relative pose.

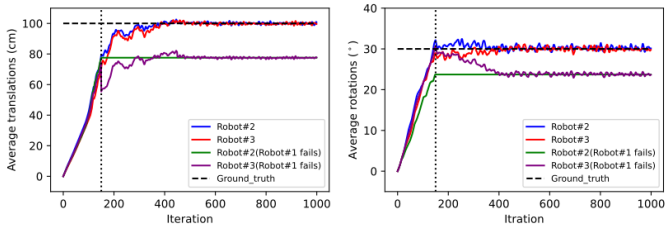

Figure 13. Average values for relative rotations and translations for R^2 and R^3, in two cases: R^1 doesn't fail (red and blue lines) and R^1 fails (green and purple lines) after 150 iterations. When R^1 fails, the global coordinate is transferred to R^2, and R^3 is optimizing for relative pose with respect to R^2.

centralized systems.

E. Convergence of the Relative Poses

In this section, convergence analysis with different initial relative pose values for the Barn sequence is provided. For simplicity, two robots are considered. The actual relative translation is 3m in all directions (i.e. x, y, and z), and the average initial values for three translations start from 0m to 2m (off by -1m to 1m), with increments of 0.02m. Similarly, the rotation ground truth is 45°, and the initial value ranges from 0° to 90° (off by -45° to 45°), with increments of 0.9°. The translation error averages (0.86, 0.76)cm, and the rotation error (0.32, 0.38)°, as shown by the blue curve in Fig. 14. Experimentally, the algorithm converges even with initial values far from the ground truth.

Table IV
DI-NERF AND BASELINE (BSLN.) PERFORMANCES FOR THE MISSION BAY SEQUENCE FOR 6 ROBOTS. $R\#1$ IS THE REFERENCE FRAME.

	R#1	R#2	R#3	R#4	R#5	R#6	Bsln.
PSNR↑	25.65	25.24	25.56	25.67	25.78	25.70	25.39
SSIM ↑	0.826	0.854	0.815	0.819	0.838	0.839	0.848
$\delta R^\circ \downarrow$	-	2.89	2.86	2.81	2.91	2.79	-
δt(cm)↓	-	4.2	4.0	4.5	4.2	4.7	-

Table V
DI-NERF AND BLOCK-NERF, ON THE MISSION BAY SEQUENCE.

Block-NeRF (centralized)			Di-NeRF (distributed)		
No. Blocks	PSNR	SSIM	No. Agents	PSNR	SSIM
2	**25.65**	**0.871**	2	25.22	0.866
8	24.56	**0.892**	8	**25.41**	0.854
16	**27.1**	**0.903**	16	25.34	0.861

F. No Overlap Setup

Even when there is no direct path overlap, scenes may be viewed from varying angles, allowing for relative pose calculation. For scenarios with no view overlaps, we have included an experiment using the Barn sequence. Fig. 12 depicts the trajectories for two robots with results. Di-NeRF struggles to compute relative poses with no view overlaps. However, knowing the relative poses, Di-NeRF merges the two models in a distributed fashion despite no absolute overlap.

G. Reference Robot Failure

If the robot chosen as the global coordinate frame fails or does not communicate, the global frame is switched to another robot. Fig. 13 shows an experiment where after 150 iterations, the reference robot is changed. Di-NeRF successfully optimizes the relative pose with respect to a new origin.

H. Technical Details of the Experiments

In this paper, three datasets are used to evaluate Di-NeRF: the synthetic dataset [2] (800×800 pixels), the Tanks and Temples [40] (1920×1080 pixels), and the Waymo dataset [8] (1217×1096 pixels). For all datasets, the batch size is 2048, the learning rate is 0.01, the weight for depth loss γ in Eq. (4) is 0.01 and the value of ρ in Eq. (9) is 0.001. ρ is the weight for the quadratic term and is the step size in the gradient ascent for the dual variable optimization. The number of steps before a communication round is 200. There are 5 communication rounds in all experiments.

Hardware and Software Configuration: The training was conducted on one NVIDIA RTX A5000 GPU with 24 GB of memory. We utilized PyTorch as our deep learning framework.

NeRF Network Configuration: The network θ has 8 layers with 256 channels each. The hashmap size is 2^{15} for synthetic sequences and 2^{19} for the Barn and Waymo sequences. Table VI shows the training times for a fully connected graph. The memory required for a NeRF model is shown in Table VII. A robot needs 40.7 MB to exchange models, which can be reduced by compressing them when bandwidth is limited.

I. Ablation Study

We conducted an ablation study to examine the effects of incorporating monocular depth estimation into relative pose optimization. The findings are detailed in Fig 14. While joint optimization of NeRF and relative pose often faces convergence issues with inaccurate initial position error estimates, the use of monocular depth estimation significantly mitigates this problem by correcting depth distortions.

Table VI
DI-NERF TRAINING TIME COMPARISON (SEC/ITR/ROBOT).

Dataset	**#Robots**	**Di-NeRF**	**Di-NeRF;known T_i^g**	**Baseline**
Synthetic Dataset	2	4.27	0.98	0.94
Tank and Temple	5	4.78	0.98	0.93
Mission Bay	6	4.73	0.97	0.95

Table VII
IMAGES AND NERF SIZE WITH THE REQUIRED BANDWIDTH

Sequence	Seq. Size (MB)	Model Size (MB)	BW per Link
Chair	31.04	4.07	40.7
Lego	42.9	4.07	40.7
Hotdog	40.9	4.07	40.7
Barn	481.5	64.07	640.7
Waymo	428.2	64.07	640.7

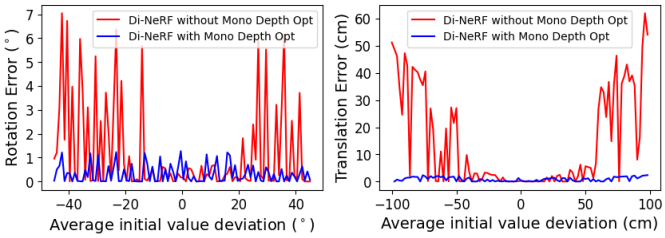

Figure 14. Impact of initial translation and rotation values on relative pose convergence (the Barn sequence) and performance differences between two approaches in optimizing the Di-NeRF model. The initial translation deviation from ground truth ranges from -1m to 1m, and for rotation from -45° to 45°. The first approach (shown in blue) integrates monocular depth estimation into the relative pose optimization, while the second approach (depicted in red) does not incorporate monocular depth estimation.

V. CONCLUSION AND FUTURE WORK

We introduced Di-NeRF, the first fully distributed NeRF used for multi-robot systems and 3D scenes. We demonstrate its versatility across synthetic and real datasets. Di-NeRF harnesses the advantages of NeRF in distributed learning, relying on RGB input for relative pose optimization. Our analyses reveal its applicability to varying numbers of robots, diverse connection types, and distinct trajectory overlaps.

In this work, the relative camera extrinsics are optimized. Optimizing the intrinsics is useful when cameras on the robots have different settings. Also, the base NeRF model for each robot has limitations in unbounded scenes. A model, designed for unbounded scenes, can be integrated into this project. Finally, we assume the local camera poses are known but with different origins. By jointly optimizing NeRF and local camera poses, one can eliminate this requirement. Also, by optimizing the architecture of NeRF, Di-NeRF can be efficient in bandwidth use compared to directly sending RGB images. This demonstrates Di-NeRF's advantage in distributed systems, where communication is localized, reducing bandwidth needs unlike in centralized systems that require comprehensive data transmission. The core strength of Di-NeRF lies in facilitating collaborative mapping in challenging environments, emphasiz-

ing the distributed nature of learning and system scalability where broad communication may not be feasible.

REFERENCES

[1] T. Halsted, O. Shorinwa, J. Yu, and M. Schwager, "A survey of distributed optimization methods for multi-robot systems," *arXiv*, 2023.

[2] 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," *Communications of the ACM*, 2021.

[3] S. Saeedi, M. Trentini, M. Seto, and H. Li, "Multiple-robot simultaneous localization and mapping: A review," *JFR*, vol. 33, pp. 3–46, 2016.

[4] R. Murai, J. Ortiz, S. Saeedi, P. H. J. Kelly, and A. J. Davison, "A robot web for distributed many-device localization," *IEEE T-RO*, 2024.

[5] N. Stathoulopoulos, A. Koval, A.-a. Agha-mohammadi, and G. Nikolakopoulos, "FRAME: Fast and robust autonomous 3D point cloud map-merging for egocentric multi-robot exploration," in *ICRA*, 2023, pp. 3483–3489.

[6] Y. Tian and J. P. How, "Spectral sparsification for communication-efficient collaborative rotation and translation estimation," *IEEE T-RO*, vol. 40, pp. 257–276, 2024.

[7] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE SPM*, vol. 37, no. 3, pp. 50–60, 2020.

[8] M. Tancik, V. Casser, X. Yan, S. Pradhan, B. P. Mildenhall, P. Srinivasan, J. T. Barron, and H. Kretzschmar, "Block-NeRF: Scalable large scene neural view synthesis," in *2022 IEEE/CVF CVPR*, 2022, pp. 8238–8248.

[9] H. Turki, D. Ramanan, and M. Satyanarayanan, "Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs," in *Proceedings of the IEEE/CVF*, 2022, pp. 12 922–12 931.

[10] J. Yu, J. A. Vincent, and M. Schwager, "DiNNO : Distributed neural network optimization for multi-robot collaborative learning," *IEEE RA-L*, vol. 7, no. 2, pp. 1896–1903, 2022.

[11] A. Cunningham, V. Indelman, and F. Dellaert, "DDF-SAM 2.0: Consistent distributed smoothing and mapping," in *2013 IEEE ICRA*. IEEE, 2013, pp. 5220–5227.

[12] M. J. Schuster, K. Schmid, C. Brand, and M. Beetz, "Distributed stereo vision-based 6D localization and mapping for multi-robot teams," *JFR*, vol. 36, no. 2, pp. 305–332, 2019.

[13] V. Tchuiev and V. Indelman, "Distributed consistent multi-robot semantic localization and mapping," *IEEE RA-L*, vol. 5, p. 4649–4656, 2020.

[14] Y. Tian, Y. Chang, F. H. Arias, C. Nieto-Granda, J. P. How, and L. Carlone, "Kimera-multi: Robust, distributed, dense metric-semantic SLAM for multi-robot systems," *IEEE T-OR*, vol. 38, no. 4, 2022.

[15] Z. Wang, S. Wu, W. Xie, M. Chen, and V. A. Prisacariu, "NeRF−−: Neural radiance fields without known camera parameters," *arXiv*, 2021.

[16] A. Yu, V. Ye, M. Tancik, and A. Kanazawa, "pixelNeRF: Neural radiance fields from one or few images," in *CVPR*, 2021, pp. 4578–4587.

[17] W. Bian, Z. Wang, K. Li, J.-W. Bian, and V. A. Prisacariu, "Nope-NeRF: Optimising neural radiance field with no pose prior," in *Proceedings of the IEEE/CVF CVPR*, 2023, pp. 4160–4169.

[18] 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.

[19] T. Suzuki, "Federated learning for large-scale scene modeling with neural radiance fields," *arXiv preprint arXiv:2309.06030*, 2023.

[20] Sara Fridovich-Keil and Alex Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa, "Plenoxels: Radiance fields without neural networks," in *CVPR*, 2022.

[21] R. Wu, B. Mildenhall, P. Henzler, K. Park, R. Gao, D. Watson, P. P. Srinivasan, D. Verbin, J. T. Barron, B. Poole *et al.*, "Reconfusion: 3d reconstruction with diffusion priors," in *IEEE/CVF*, 2024.

[22] 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 *IEEE/CVF CVPR*, June 2022.

[23] E. Sucar, S. Liu, J. Ortiz, and A. Davison, "iMAP: Implicit mapping and positioning in real-time," in *ICCV*, 2021.

[24] A. Rosinol, J. J. Leonard, and L. Carlone, "NeRF-SLAM: Real-time dense monocular slam with neural radiance fields," in *2023 IEEE/RSJ IROS*. IEEE, 2023, pp. 3437–3444.

[25] A. L. Teigen, Y. Park, A. Stahl, and R. Mester, "RGB-D mapping and tracking in a Plenoxel radiance field," in *WACV*, 2024, pp. 3342–3351.

[26] J. L. Schönberger and J.-M. Frahm, "Structure-from-Motion Revisited," in *CVPR*, 2016.

[27] Y. Jeong, S. Ahn, C. Choy, A. Anandkumar, M. Cho, and J. Park, "Self-calibrating neural radiance fields," in *ICCV*, 2021.

[28] L. Yen-Chen, P. Florence, J. T. Barron, A. Rodriguez, P. Isola, and T.-Y. Lin, "iNeRF: Inverting neural radiance fields for pose estimation," in *IEEE/RSJ IROS*, 2021.

[29] J. Fang, S. Lin, I. Vasiljevic, V. Guizilini, R. Ambrus, A. Gaidon, G. Shakhnarovich, and M. R. Walter, "NeRFuser: Large-scale scene representation by NeRF fusion," *arXiv*, 2023.

[30] X. Zhang, S. Bi, K. Sunkavalli, H. Su, and Z. Xu, "NeRFusion: Fusing radiance fields for large-scale scene reconstruction," in *IEEE/CVF CVPR*, 2022, pp. 5449–5458.

[31] L. Holden, F. Dayoub, D. Harvey, and T.-J. Chin, "Federated neural radiance fields," *arXiv*, 2023.

[32] S. Choudhary, L. Carlone, H. I. Christensen, and F. Dellaert, "Exactly sparse memory efficient slam using the multi-block alternating direction method of multipliers," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 1349–1356.

[33] V. Indelman, E. Nelson, J. Dong, N. Michael, and F. Dellaert, "Incremental distributed inference from arbitrary poses and unknown data association: Using collaborating robots to establish a common reference," *IEEE Control Systems Magazine*, vol. 36, no. 2, pp. 41–74, 2016.

[34] L. Mi, A. Kundu, D. Ross, F. Dellaert, N. Snavely, and A. Fathi, "im2nerf: Image to neural radiance field in the wild," *arXiv preprint arXiv:2209.04061*, 2022.

[35] R. Ranftl, A. Bochkovskiy, and V. Koltun, "Vision transformers for dense prediction," in *IEEE/CVF*, 2021, pp. 12 179–12 188.

[36] A. Losi and M. Russo, "On the Application of the Auxiliary Problem Principle," *J. Optim. Theory Appl.*, vol. 117, no. 2, pp. 377–396, 2003.

[37] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

[38] O. Shorinwa, T. Halsted, J. Yu, and M. Schwager, "Distributed optimization methods for multi-robot systems: Part 2—a survey," *IEEE RAM*, 2024.

[39] S. Fridovich-Keil, F. Valdivia, G. Wetzstein, B. Recht, and M. Soltanolkotabi, "Gradient descent provably solves nonlinear tomographic reconstruction," *ArXiv*, 2023.

[40] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun, "Tanks and Temples: Benchmarking large-scale scene reconstruction," *ACM TOG*, vol. 36, no. 4, 2017.

[41] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

VI. SUPPLEMENTARY MATERIAL

A. The Auxiliary parameter in Di-NeRF

Considering the distributed optimization in Di-NeRF that is subject to the constraint $\theta_i = z_{ij}$, the following sub-problem optimization can be written for each robot as follows using the auxiliary variable z_{ij}:

$$\Gamma_i^{(k+1)} = \underset{\Gamma}{\operatorname{argmin}} \tag{11}$$

$$\{L_i\left(\Gamma_i\right) + \theta_i^\top y_i^{(k)} + \rho \sum_{j \in \mathcal{N}_i} \left\| \theta_i - \bar{\theta}^{(k)} \right\|_2^2\} \tag{12}$$

where L_i is the local loss function for each robot, and $\Gamma_i = \{\theta_i, T_i^g, \boldsymbol{\psi}_i\}$). θ_i is the network parameters of the robot i which is communicating with the robot j. The parameter ρ is the weight for the quadratic term $\sum_{j \in \mathcal{N}_i} \|\theta_i - z_{ij}\|_2^2$ and the step size in the gradient ascent of the dual variable y_i. $\mathcal{N}_i$ is the set of neighbours of robot i, and T_i^g is the relative pose of robot i with respect to the global coordinate. According to (12), the local loss function for each robot can be updated according to the following [41]:

$$\mathcal{L}_i(\Gamma_i) = L_i(\Gamma_i) + \theta_i^\top y_i + \frac{\rho}{2} \sum_{j \in \mathcal{N}_i} \|\theta_i - z_{ij}\|_2^2 \ , \tag{13}$$

where $\mathcal{L}_i$ is the updated loss function for each robot based on Di-NeRF. The dual variable y_i update according to C-ADMM is as follows:

$$y_i^{(k+1)} = y_i^{(k)} + \rho(\theta_i^{(k+1)} - z_{ij}^{(k+1)}), \tag{14}$$

To update the auxiliary parameter z_{ij} given $\theta_i - z_{ij} = 0$ and (12), the z update subproblem is as follows:

$$\min_{z^{(k+1)}} \quad -\sum_{i\in\mathcal{N}_i} y_i^{(k)} z^{(k+1)} + \frac{1}{2}\sum_{i\in\mathcal{N}_i} \left\|\theta_i^{(k+1)} - z^{(k+1)}\right\|^2 \tag{15}$$

By taking the first derivative from (15), the following equation can be driven for z_{ij}:

$$z^{(k+1)} = \frac{1}{\mathcal{N}} \sum_{i\in\mathcal{N}_i} (\theta_i^{(k+1)} + \frac{1}{\rho} y_i^{(k)}) \tag{16}$$

Dual variable update is an equality, not an optimization problem and is called a central collector or fusion center. Equation (16) can be simplified further by writing it as:

$$z^{(k+1)} = \bar{\theta}^{(k+1)} + \frac{1}{\rho}\bar{y}^{(k)} \tag{17}$$

Substitution of (17) into the average value of y_i over $\mathcal{N}_i$ in (14) (i.e. $\bar{y}_i^{(k+1)} = \bar{y}_i^{(k)} + \rho(\bar{\theta}_i^{(k+1)} - z^{(k+1)})$), yields $\bar{y}_i^{(k+1)} = 0$. Further substitution of this result into (17) leads to the following final equation:

$$z_{ij}^{(k+1)} = \frac{1}{N}\sum \theta_i^{(k)} := \bar{\theta}^{(k)} \ , \tag{18}$$

During each iteration, k, every robot, indexed by i, independently solves its own subproblem to determine the value of the global variable $\theta_i^{(k)}$. The robot incurs a penalty proportional to the deviation of its variable from the mean value of the global variable, as computed from all robots in the preceding iteration.