Robotics: Science and Systems 2025
Los Angeles, CA, USA, June 21–25, 2025

RAMEN: Real-time Asynchronous Multi-agent Neural Implicit Mapping

Hongrui Zhao
ICON Lab
Department of Aerospace Engineering
University of Illinois Urbana-Champaign
Email: hongrui5@illinois.edu

Boris Ivanovic
NVIDIA Research
Email: bivanovic@nvidia.com

Negar Mehr
ICON Lab
Department of Mechanical Engineering
University of California Berkeley
Email: negar@berkeley.edu

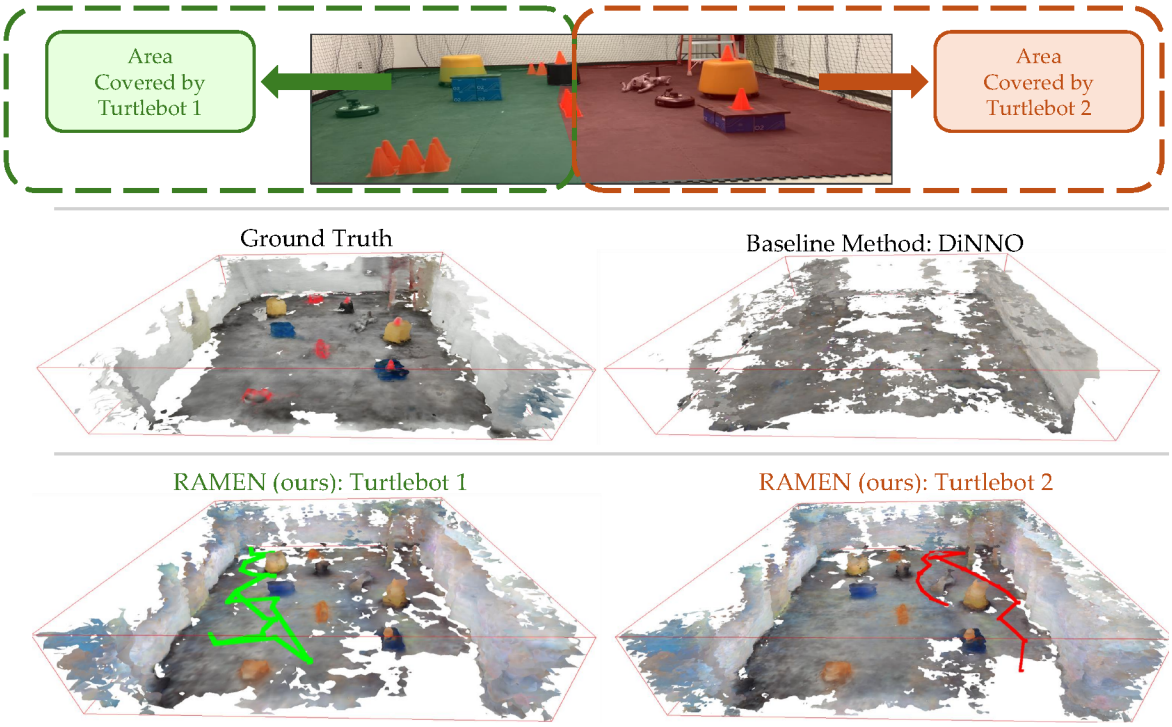

Fig. 1: In a challenging real-world experiment with limited communication (agents can only exchange information every 30 seconds), our method RAMEN enables each turtlebot to successfully map the full scene while only physically visiting half of the scene (explored areas and trajectories are colored accordingly). Our method achieves accuracy comparable to the ground truth while the baseline method (DiNNO) fails to converge.

***Abstract*—Multi-agent neural implicit mapping allows robots to collaboratively capture and reconstruct complex environments with high fidelity. However, existing approaches often rely on synchronous communication, which is impractical in real-world scenarios with limited bandwidth and potential communication interruptions. This paper introduces RAMEN: Real-time Asynchronous Multi-agEnt Neural implicit mapping, a novel approach designed to address this challenge. RAMEN employs an uncertainty-weighted multi-agent consensus optimization algorithm that accounts for communication disruptions. When communication is lost between a pair of agents, each agent retains only an outdated copy of its neighbor's map, with the uncertainty of this copy increasing over time since the last communication. Using gradient update information, we quantify the uncertainty associated with each parameter of the neural network map. Neural network maps from different agents are brought to consensus on the basis of their levels of uncertainty, with consensus biased towards network parameters with lower uncertainty. To achieve this, we derive a weighted variant of the decentralized consensus alternating direction method of multipliers (C-ADMM) algorithm, facilitating robust collaboration among agents with varying communication and update frequencies. Through extensive evaluations on real-world datasets and robot hardware experiments, we demonstrate RAMEN's superior mapping performance under challenging communication conditions. Supplementary videos and codes are available at https://iconlab.negarmehr.com/RAMEN/.**

I. Introduction

Multi-robot systems rely on efficient communication and accurate map merging to explore and understand their environment. By sharing individual observations, agents gain a comprehensive understanding of the scene, enabling effective

trajectory planning and task coordination. This capability is crucial for diverse applications like autonomous driving [1, 2, 3], warehouse automation [4], and multi-UAV search and survey missions [5]. Neural implicit maps, such as Neural Radiance Fields (NeRF) [6], are emerging as a powerful tool for multi-robot mapping, offering a compelling alternative to traditional map representations [7, 8]. Neural implicit maps efficiently store complex geometries and realistic appearance information (including color and lighting) in compact neural networks, requiring minimal storage space on robots. Furthermore, high-level semantic information, such as object classes, can be readily integrated into neural implicit maps [9, 10].

However, challenges arise when multiple robots perform neural implicit mapping in real-time. While conventional map representations (e.g., point clouds, occupancy grids) are easily merged [11, 12], neural network parameters cannot be simply combined. Existing multi-agent neural implicit mapping methods often utilize distributed optimization algorithms, such as distributed stochastic gradient descent (DSGD) or C-ADMM, to gradually bring different neural networks into *consensus* [13, 14, 15]. This consensus results in each robot possessing an identical neural network representing the collective sensor observations. The consensus optimization, however, is vulnerable to divergence when communication is asynchronous.

Previous works [16] reveal that existing distributed optimization algorithms, such as the C-ADMM employed by the state-of-the-art multi-agent neural implicit mapping methods DiNNO [13] and Di-NeRF [15], are not robust to asynchronous communication or message loss. These algorithms require perfect synchronization, where agents consistently receive the latest network parameters from neighbors before each optimization step. However, real-world robot communication is often asynchronous: updates may be delayed or lost due to bandwidth limitations, best-effort protocols (e.g., UDP), obstacles between agents, or agents moving out of range. Consequently, as communication failures increase, multi-agent mapping can produce inaccurate reconstructions or even diverge entirely, as is the case for DiNNO [13] in Fig. 1. Existing approaches to handle such communication disruptions often rely on barriers that halt optimization to wait for the slowest agent [17, 18]. This is impractical for real-time robotic applications where continuous mapping is crucial for planning.

In this paper, we introduce RAMEN, Real-time Asynchronous Multi-agEnt Neural implicit mapping, to address these challenges. RAMEN explicitly addresses the uncertainty inherent in distributed multi-agent systems by employing a frequency-based epistemic uncertainty quantification method. This method leverages gradient update information to assess the reliability of each agent's neural implicit map, recognizing that infrequent updates due to communication loss lead to higher uncertainty in the neural reconstruction. To achieve consensus between agents, we derive an uncertainty-weighted decentralized C-ADMM algorithm which enables RAMEN to effectively fuse knowledge from multiple robots while favoring more reliable information. To the best of our knowledge, this work presents the first real-world demonstration of multi-robot neural implicit mapping. Through extensive simulation and hardware experiments, we show that RAMEN successfully constructs maps in real-time despite communication disruptions, outperforming existing methods that fail under such conditions. A summary of our key contributions is:

1. We propose RAMEN, a multi-agent neural implicit mapping method with frequency-based uncertainty to address communication disruptions.
2. We derive an uncertainty-weighted decentralized C-ADMM algorithm, allowing RAMEN to prioritize reliable information during map consensus optimization.
3. We demonstrate the first real-world implementation of multi-robot neural implicit mapping, showcasing the robustness of RAMEN under challenging communication conditions.

II. Related Works

Real-time Neural Implicit Mapping. Neural implicit mapping methods based on NeRF utilize RGB or RGBD images to train neural networks for various scene prediction tasks, including color, volume density, occupancy, and signed distance. Examples include iSDF [19] for real-time signed distance field reconstruction, H2-Mapping [20] for efficient color and signed distance prediction on edge devices, and GO-Surf [21] for fast surface reconstruction using multi-resolution features. Some approaches, like iMAP [22] and NICE-SLAM [23], integrate camera pose tracking with scene representation, while Co-SLAM [24] further enhances reconstruction speed. However, these methods primarily focus on centralized training with a single agent. In contrast, this paper builds upon Co-SLAM and introduces an asynchronous distributed training framework specifically designed for multi-robot systems.

Multi-agent Mapping. Existing multi-agent mapping methods involve transmitting either explicit local maps (e.g., 3D point clouds with image features [25, 26, 12]) or raw sensor data to a central server for map alignment and fusion [27]. Alternatively, methods like MAANS [28] employ a central server to fuse bird's-eye view images predicted by each agent. Recent approaches leverage neural implicit mapping to reduce communication costs by sharing compact neural networks instead of raw data [13, 15, 14]. However, existing neural implicit mapping methods require synchronous communication, hindering their applicability in real-world robotic scenarios where communication is often asynchronous. Bandwidth limitations, best-effort protocols (e.g., UDP), and obstacles can lead to delayed or lost updates. Our proposed RAMEN method addresses this gap by enabling robust multi-agent neural implicit mapping under asynchronous communication.

Distributed Neural Network Optimization. Distributed optimization of neural networks allows multiple agents to collaboratively train a model without relying on a central server [29]. Common approaches include distributed stochastic gradient descent (DSGD) [30], which combines local gradients with neighbors' parameters, and distributed stochastic gradient tracking (DSGT) [31], which improves convergence by estimating the global gradient. Other methods, like the

decentralized linearized alternating direction method (DLM) [32] and consensus alternating direction method of multipliers (C-ADMM) [13], optimize local objectives while enforcing agreement among agents. However, these methods typically require synchronous communication, limiting their practicality. While techniques like partial barriers and bounded delay [17, 18] allow for some degree of tolerance to communication disruptions by introducing waiting periods, they are primarily designed for distributed training with data servers and are not ideal for robotic tasks requiring continuous real-time updates. RAMEN tackles communication challenges without relying on such barriers, enabling efficient and robust multi-robot collaborative learning.

Uncertainty in Neural Implicit Maps. Quantifying uncertainty in neural implicit maps is crucial for tasks like next-best view selection, model refinement, and artifact removal. Existing methods for estimating neural network parameter uncertainty often rely on computationally expensive techniques like deep ensembles [33] or variational Bayesian neural networks [34], which require significant modifications to the training pipeline. While alternative approaches estimate spatial uncertainty by perturbing input coordinates [35] or network parameters [36], they do not directly address the uncertainty of each learnable parameter. In contrast, RAMEN efficiently quantifies parameter uncertainty by combining a frequency-based heuristic [37] with a multi-resolution hash grid [24]. This requires minimal extra computation and allows us to leverage uncertainty information for robust distributed mapping.

III. Preliminaries

This section provides the necessary background for our proposed approach. We first introduce Co-SLAM [24], a single-agent neural implicit mapping method that forms the foundation of RAMEN. We then discuss existing multi-agent neural implicit mapping approaches, assuming synchronous communications. This discussion highlights the limitations of current methods and motivates the need for a system like RAMEN, which can handle the asynchronous communications common in real-world robotic scenarios.

A. Single-agent Neural Implicit Mapping

Co-SLAM, a single-agent neural implicit mapping method, represents scenes implicitly using three components: a multi-resolution hash-based feature grid [38] which we denote by $V_\alpha = \{V_\alpha^l\}_{l=1}^L$, a geometry MLP decoder F_τ, and a color MLP decoder F_ϕ. The feature grid V_α comprises L levels, each storing feature vectors at grid vertices with resolutions increasing in a geometric progression from coarsest to finest. Higher resolution levels have more vertices placed across the scene, thus capture finer details. We denote the learnable parameters of the feature grid and decoders as $\mathbf{\Theta} = \{\alpha, \tau, \phi\}$.

Given the world coordinate $\mathbf{x}$ of a point, Co-SLAM's neural implicit map predicts its corresponding RGB value $\mathbf{c}$ and truncated signed distance function (SDF) value s. The SDF value s represents the minimum distance of a given point to the surface boundary of a geometric shape, with its sign indicating whether the point lies inside (-) or outside (+) the surface. To enhance the smoothness of the scene geometry, one-blob encoding [39] $\gamma(\cdot)$ is applied to $\mathbf{x}$ before passing it to the geometry decoder F_τ. With the encoded coordinate $\gamma(\mathbf{x})$ and feature vectors from the corresponding vertices $V_\alpha(\mathbf{x})$, the geometry decoder outputs a feature vector $\mathbf{h}$ and the SDF value s:

$$F_\tau\left(\gamma(\mathbf{x}), V_\alpha(\mathbf{x})\right) \rightarrow (\mathbf{h}, s). \tag{1}$$

The color decoder then outputs the RGB value $\mathbf{c}$:

$$F_\phi\left(\gamma(\mathbf{x}), \mathbf{h}\right) \rightarrow \mathbf{c}. \tag{2}$$

Co-SLAM renders RGB and depth images from its neural implicit map. With the camera intrinsic calibration matrix K and pose P, each pixel with 2D image plane coordinates (u, v) is associated with a ray $\mathbf{r}$ in the direction $PK^{-1}(u, v, 1)^\top$. Given the camera origin $\mathbf{o}$ and ray direction $\mathbf{r}$, we sample M points $\mathbf{x}_i = \mathbf{o} + d_i\mathbf{r}$ along the ray, where d_i is the distance from the camera origin. For each point $\mathbf{x}_i$, we obtain the corresponding s_i and $\mathbf{c}_i$ using (1) and (2). The pixel color $\hat{\mathbf{c}}$ and depth $\hat{d}$ are then rendered as:

$$\hat{\mathbf{c}} = \frac{1}{\sum_{i=1}^M w_i} \sum_{i=1}^M w_i \mathbf{c}_i, \tag{3a}$$

$$\hat{d} = \frac{1}{\sum_{i=1}^M w_i} \sum_{i=1}^M w_i \mathbf{d}_i, \tag{3b}$$

where w_i represents the weight assigned to each sampled point along the ray, peaking at the surface of a geometric shape [40]. Given a truncated distance tr (we truncate the SDF value s to tr for points far from the surface), we compute w_i with two Sigmoid functions $\sigma(\cdot)$:

$$w_i = \sigma\left(\frac{s_i}{tr}\right)\sigma\left(-\frac{s_i}{tr}\right). \tag{4}$$

We train the neural implicit map by minimizing the difference between rendered RGBD images and the images from robot's camera. We use notation R to denote the actual images collected by the robot. The objective function is then given as:

$$L^{obj}(\mathbf{\Theta}, R) = L_{rgb} + L_d + L_{sdf} + L_{fs} + L_{smooth}, \tag{5}$$

where L_{rgb} and L_d are L2 losses between the rendered and observed RGB and depth images, respectively; L_{sdf} is the loss for SDF values; L_{fs} encourages points far from the surface to have a SDF value equal to the truncated distance tr, and L_{smooth} promotes smoothness in the reconstructed geometry. For a detailed explanation of each loss term, please refer to [24].

B. Synchronous Multi-agent Mapping

We now discuss how Co-SLAM can be extended into synchronous multi-agent mapping. In a multi-agent setup, a group of agents collaboratively map their environment and share information within a communication graph $G = (\mathcal{V}, \mathcal{E})$. Each agent is represented as a node $i \in \mathcal{V}$, and their communication

connectivity is defined by a set of edges $\mathcal{E}$. Each agent i can communicate with its neighbors $N_i = \{j \in \mathcal{V} \mid (i,j) \in \mathcal{E}\}$, sharing its learned parameters $\boldsymbol{\Theta}_i$ instead of raw images. Each agent i captures posed RGBD images R_i with its onboard camera. During each mapping iteration, agent i samples pixels from R_i and minimizes its local objective function L_i^{obj}:

$$L_i^{obj}(\boldsymbol{\Theta}_i, R_i) = L_{rgb_i} + L_{d_i} + L_{sdf_i} + L_{fs_i} + L_{smooth_i}, \quad (6)$$

To achieve consensus, each agent also aligns its model parameters $\boldsymbol{\Theta}_i$ with those of its neighbors $\boldsymbol{\Theta}_j$, for all $j \in N_i$. Through consensus, agents converge on a shared neural map, allowing them to exchange environmental information. This problem is formulated as minimizing the sum of local objectives while ensuring agreement among the neural maps [41]:

$$\min_{\{\boldsymbol{\Theta}_1,\cdots\}\{\mathbf{z}_{ij},\cdots\}} \sum_{i\in\mathcal{V}} L_i^{obj}(\boldsymbol{\Theta}_i, R_i), \quad (7a)$$

$$\text{s.t. } \boldsymbol{\Theta}_i = \mathbf{z}_{ij}, \boldsymbol{\Theta}_j = \mathbf{z}_{ij}, \ \forall (i,j) \in \mathcal{E}, \quad (7b)$$

where $\mathbf{z}_{ij}$ is a local auxiliary variable imposing consensus on neighboring agents i and j. This formulation represents a *decentralized consensus problem* (or graph consensus problem), where agreement is enforced only between pairs of neighbors. In contrast, a *global consensus problem* requires all $\boldsymbol{\Theta}_i$ to agree with a global variable $\mathbf{z}$, managed by a central fusion center [42]. Decentralized consensus is more suitable for multi-robot systems, as it avoids the single point of failure and vulnerability associated with a fusion center.

To solve (7), we first formulate the augmented Lagrangian:

$$\mathcal{L}^a = \sum_{i\in\mathcal{V}} \Bigg(L_i^{obj}(\boldsymbol{\Theta}_i, R_i) + \sum_{j\in N_i} \boldsymbol{\lambda}_{ij}^\top (\boldsymbol{\Theta}_i - \mathbf{z}_{ij}) + \mathbf{v}_{ij}^\top (\boldsymbol{\Theta}_j - \mathbf{z}_{ij}) + \frac{\rho}{2}\|\boldsymbol{\Theta}_i - \mathbf{z}_{ij}\|_2^2 + \frac{\rho}{2}\|\boldsymbol{\Theta}_j - \mathbf{z}_{ij}\|_2^2 \Bigg), \quad (8)$$

where $\boldsymbol{\lambda}_{ij}$ and $\mathbf{v}_{ij}$ are dual variables (or Lagrange multipliers) penalizing violations of consensus constraints and bringing $\boldsymbol{\Theta}_i$ and $\boldsymbol{\Theta}_j$ to $\mathbf{z}_{ij}$, ρ is a positive penalty parameter which determines how fast we want to achieve consensus, and $\|\cdot\|_2$ is L2 norm. Additionally, we define $\mathbf{p}_i = \sum_{j\in N_i} \boldsymbol{\lambda}_{ij} + \mathbf{v}_{ji}$, which combines all dual variables associated with $\boldsymbol{\Theta}_i$.

Multi-agent mapping employs an iterative algorithm to continually update neural implicit maps. Superscript t is used to denote the values of variables at iteration t. For agent i, at mapping iteration t, we apply ADMM [42] to update $\boldsymbol{\Theta}_i$ and $\mathbf{p}_i$ in an alternating fashion to minimize (8):

$$\boldsymbol{\Theta}_i^{t+1} = \underset{\boldsymbol{\Theta}_i}{\text{argmin}} \ L_i^{obj}(\boldsymbol{\Theta}_i, R_i) + \boldsymbol{\Theta}_i^\top \mathbf{p}_i^t + \rho \sum_{j\in N_i} \left\| \boldsymbol{\Theta}_i - \frac{\boldsymbol{\Theta}_i^t + \boldsymbol{\Theta}_j^t}{2} \right\|_2^2, \quad (9a)$$

$$\mathbf{p}_i^{t+1} = \mathbf{p}_i^t + \rho \sum_{j\in N_i} (\boldsymbol{\Theta}_i^{t+1} - \boldsymbol{\Theta}_j^{t+1}). \quad (9b)$$

We refer to (9a) as the *primal update* and (9b) as the *dual update*. Instead of performing the exact minimization in the primal update, we approximate $\boldsymbol{\Theta}_i^{t+1}$ by taking few steps of stochastic gradient descent. This approach, used in DiNNO [13] and Di-NeRF [15] for synchronous multi-agent mapping, requires continuous communication between agents i and j if $(i,j) \in \mathcal{E}$ initially. This ensures that the latest $\boldsymbol{\Theta}_i$ and $\boldsymbol{\Theta}_j$ are always available for (9). However, this requirement is impractical for real-world robotic systems. In the next section, we introduce an uncertainty-weighted version of (9) to address this limitation.

IV. Asynchronous Multi-agent Neural Implicit Mapping

This section details the core components of RAMEN's asynchronous multi-agent mapping framework. In our context, asynchronism arises from two primary sources: communication drops and variations in processing speed. This work primarily addresses communication drops, as they pose a significant challenge for both homogeneous and heterogeneous robot teams. We first derive an uncertainty-weighted version C-ADMM algorithm, utilizing uncertainty to favor reliable information in the consensus optimization process. Then, we introduce our proposed frequency-based epistemic uncertainty quantification method, detailing how it assesses the reliability of each agent's contributions to the collaborative mapping process.

A. Uncertainty-Weighted Decentralized C-ADMM

Asynchronous communication presents two key challenges:

1) *Mitigating the influence of uncertain parts of outdated models*: When communication between agents i and j is disrupted, agent i at iteration t may possess an outdated copy of agent j's model parameters, denoted as $\boldsymbol{\Theta}_j^{t_{old}}$, where $t_{old} \ll t$. This outdated model may represent some regions of the environment with high uncertainty and poor quality due to lack of mapping updates. Therefore, it is crucial to limit the influence of uncertain parts of $\boldsymbol{\Theta}_j^{t_{old}}$ on agent i's current model $\boldsymbol{\Theta}_i^t$ to avoid degrading its accuracy.
2) *Leveraging valuable information from outdated models*: Despite being outdated, $\boldsymbol{\Theta}_j^{t_{old}}$ may still contain valuable information for agent i, such as details about regions explored by agent j but not yet visited by agent i. We want to identify the neural network parameters that encode these information and assign them higher weights when enforcing consensus.

To address these challenges, we derive an uncertainty-weighted decentralized C-ADMM algorithm that prioritizes reliable neural network parameters during the consensus optimization process. While various weighted C-ADMM algorithms exist [43, 44, 45, 46], this paper introduces, to the best of our knowledge, the first decentralized C-ADMM algorithm that individually weights each optimization parameter according to its associated uncertainty. This allows for fine-grained control over the influence of each parameter during the consensus process, enabling more robust and accurate multi-agent mapping.

Following [43], we start by writing the weighted augmented Lagrangian based on (8):

$$\mathcal{L}^{wa} = \sum_{i\in\mathcal{V}} \Bigg(L_i^{obj}(\boldsymbol{\Theta}_i, R_i) + \sum_{j\in N_i} \boldsymbol{\lambda}_{ij}^\top(\boldsymbol{\Theta}_i - \mathbf{z}_{ij}) + \mathbf{v}_{ij}^\top(\boldsymbol{\Theta}_j - \mathbf{z}_{ij}) + \frac{\rho}{2}\|\boldsymbol{\Theta}_i - \mathbf{z}_{ij}\|^2_{\mathbf{W}_{ij}} + \frac{\rho}{2}\|\boldsymbol{\Theta}_j - \mathbf{z}_{ij}\|^2_{\mathbf{W}_{ji}} \Bigg), \tag{10}$$

where we use a weighted norm $\|\mathbf{x}\|^2_{\mathbf{W}} = \mathbf{x}^\top\mathbf{W}\mathbf{x}$ instead of L2 norm in (8). Note that $\mathbf{W}_{ij}^t$, $\mathbf{W}_{ji}^t$ are weight matrices reflecting the relative reliability of information shared between agents i and j at iteration t. Intuitively, a higher weight indicates greater confidence/lower uncertainty in the corresponding information.

Applying ADMM [42] to (10), at each iteration t for agent i, we alternatively update local consensus variable $\mathbf{z}_{ij}$, neural implicit map $\boldsymbol{\Theta}_i$, and consensus-enforcing dual variables $\boldsymbol{\lambda}_{ij}, \mathbf{v}_{ij}$. We obtain $\mathbf{z}_{ij}^{t+1}$ by setting the derivative of (10) with respect to $\mathbf{z}_{ij}$ to zero:

$$\mathbf{z}_{ij}^{t+1} = (\mathbf{W}_{ij}^t + \mathbf{W}_{ji}^t)^{-1}(\mathbf{W}_{ij}^t\boldsymbol{\Theta}_i^t + \mathbf{W}_{ji}^t\boldsymbol{\Theta}_j^t) + \frac{1}{\rho}(\mathbf{W}_{ij}^t + \mathbf{W}_{ji}^t)^{-1}(\boldsymbol{\lambda}_{ij}^t + \mathbf{v}_{ij}^t). \tag{11}$$

After obtaining $\mathbf{z}_{ij}^{t+1}$ and $\boldsymbol{\Theta}_i^{t+1}$, we compute the dual variables $\boldsymbol{\lambda}_{ij}^{t+1}, \mathbf{v}_{ij}^{t+1}$ by performing gradient ascents to penalize consensus violations:

$$\boldsymbol{\lambda}_{ij}^{t+1} = \boldsymbol{\lambda}_{ij}^t + \rho\mathbf{W}_{ij}^t\nabla_{\boldsymbol{\lambda}_{ij}}\mathcal{L}^{wa}, \tag{12a}$$
$$\mathbf{v}_{ij}^{t+1} = \mathbf{v}_{ij}^t + \rho\mathbf{W}_{ji}^t\nabla_{\mathbf{v}_{ij}}\mathcal{L}^{wa}, \tag{12b}$$

where the gradients are:

$$\nabla_{\boldsymbol{\lambda}_{ij}}\mathcal{L}^{wa} = \boldsymbol{\Theta}_i^{t+1} - \mathbf{z}_{ij}^{t+1}, \tag{13a}$$
$$\nabla_{\mathbf{v}_{ij}}\mathcal{L}^{wa} = \boldsymbol{\Theta}_j^{t+1} - \mathbf{z}_{ij}^{t+1}, \tag{13b}$$

In (12), $\mathbf{W}_{ij}^t$ and $\mathbf{W}_{ji}^t$ apply strong penalty if $\boldsymbol{\Theta}_i^{t+1}$ and $\boldsymbol{\Theta}_j^{t+1}$ diverge from the reliable information in $\mathbf{z}_{ij}^{t+1}$.

Initializing the dual variables as $\boldsymbol{\lambda}_{ij}^0 = \mathbf{v}_{ij}^0 = \mathbf{0}$, we can observe that $\boldsymbol{\lambda}_{ij}^t = -\mathbf{v}_{ij}^t$ by combining (11), (12a), and (12b). We also notice $\boldsymbol{\lambda}_{ij}^t = \mathbf{v}_{ji}^t$. This allows us to rewrite (11) as:

$$\mathbf{z}_{ij}^{t+1} = (\mathbf{W}_{ij}^t + \mathbf{W}_{ji}^t)^{-1}(\mathbf{W}_{ij}^t\boldsymbol{\Theta}_i^t + \mathbf{W}_{ji}^t\boldsymbol{\Theta}_j^t), \tag{14}$$

Note that (14) shows $\mathbf{z}_{ij}^{t+1} = \mathbf{z}_{ji}^{t+1}$, and (14) can be interpreted as a weighted average between $\boldsymbol{\Theta}_i^t$ and $\boldsymbol{\Theta}_j^t$, where higher weights are assigned to parameters with lower uncertainty. Thus, $\mathbf{z}_{ij}^{t+1}$ weights the parameters based on how reliable they are, serving as the consensus target for both agents.

To compute gradients for obtaining $\boldsymbol{\Theta}_i^{t+1}$, we are interested in all terms associated with $\boldsymbol{\Theta}_i$ (including $\mathbf{v}_{ji}^\top\boldsymbol{\Theta}_i = \boldsymbol{\lambda}_{ij}^\top\boldsymbol{\Theta}_i$ and $\frac{\rho}{2}\|\boldsymbol{\Theta}_i - \mathbf{z}_{ji}\|^2_{\mathbf{W}_{ij}} = \frac{\rho}{2}\|\boldsymbol{\Theta}_i - \mathbf{z}_{ij}\|^2_{\mathbf{W}_{ij}}$ from the j^{th} element of $\mathcal{L}^{wa}$). We find $\boldsymbol{\Theta}_i^{t+1}$ that minimizes (10) to be:

$$\boldsymbol{\Theta}_i^{t+1} = \underset{\boldsymbol{\Theta}_i}{\operatorname{argmin}}\ L_i^{obj}(\boldsymbol{\Theta}_i, R_i) + \sum_{j\in N_i} 2\boldsymbol{\Theta}_i^\top\boldsymbol{\lambda}_{ij}^t + \rho\|\boldsymbol{\Theta}_i - (\mathbf{W}_{ij}^t + \mathbf{W}_{ji}^t)^{-1}(\mathbf{W}_{ij}^t\boldsymbol{\Theta}_i^t + \mathbf{W}_{ji}^t\boldsymbol{\Theta}_j^t)\|^2_{\mathbf{W}_{ij}^t}, \tag{15}$$

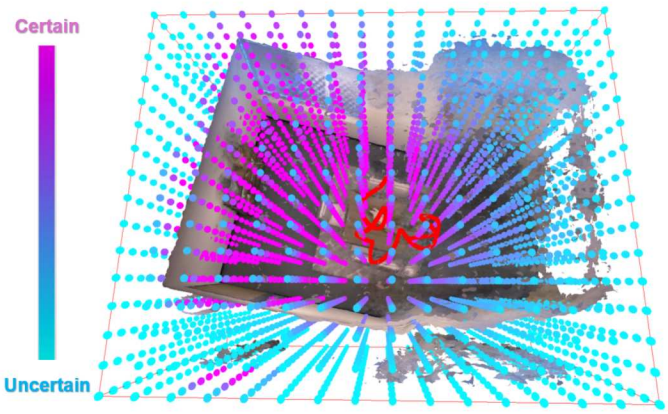

Fig. 2: This visualization shows the uncertainty associated with the parameters in the coarsest feature grid $V_\alpha^{l=1}$ of the neural implicit map. We represent this uncertainty by overlaying color-coded vertices on the reconstructed 3D mesh. The robot's trajectory is shown in red. The robot's trajectory and field-of-view are limited to the left half of the room. As expected, high uncertainty (blue vertices) corresponds to the right half of the map not explored by the robot where the reconstructed geometry is inaccurate.

This gives us the primal update of the weighted decentralized C-ADMM. By defining $\mathbf{p}_i^t := 2\sum_{j\in N_i}\boldsymbol{\lambda}_{ij}^t$, we can rewrite (15) as:

$$\boldsymbol{\Theta}_i^{t+1} = \underset{\boldsymbol{\Theta}_i}{\operatorname{argmin}}\ L_i^{obj}(\boldsymbol{\Theta}_i, R_i) + \boldsymbol{\Theta}_i^\top\mathbf{p}_i^t + \rho\sum_{j\in N_i}\|\boldsymbol{\Theta}_i - (\mathbf{W}_{ij}^t + \mathbf{W}_{ji}^t)^{-1}(\mathbf{W}_{ij}^t\boldsymbol{\Theta}_i^t + \mathbf{W}_{ji}^t\boldsymbol{\Theta}_j^t)\|^2_{\mathbf{W}_{ij}^t}, \tag{16}$$

Using (12a), we perform dual update to get $\mathbf{p}_i^{t+1}$:

$$\mathbf{p}_i^{t+1} = \mathbf{p}_i^t + 2\rho\mathbf{W}_{ij}^t\sum_{j\in N_i}(\mathbf{W}_{ij}^t + \mathbf{W}_{ji}^t)^{-1}(\mathbf{W}_{ji}^t\boldsymbol{\Theta}_i^{t+1} - \mathbf{W}_{ji}^t\boldsymbol{\Theta}_j^{t+1}), \tag{17}$$

After going into the next iteration, we need to update the weights which reflect how reliable the network parameters are. We will discuss how we can update the weights in the next subsection.

B. Frequency-based Epistemic Uncertainty

We quantify the *epistemic* uncertainty associated with each model parameter to obtain weights for our C-ADMM algorithm. Epistemic uncertainty reflects the model's lack of knowledge about the environment it is mapping [35]. We base our simple-yet-effective uncertainty quantification method on two key observations:

1) The uncertainty of an area decreases monotonically with the frequency of observations [37].
2) Unlike the highly correlated parameters in a standard MLP, each feature vector in the feature grid V_α corresponds to a unique vertex in the grid [35]. Updates

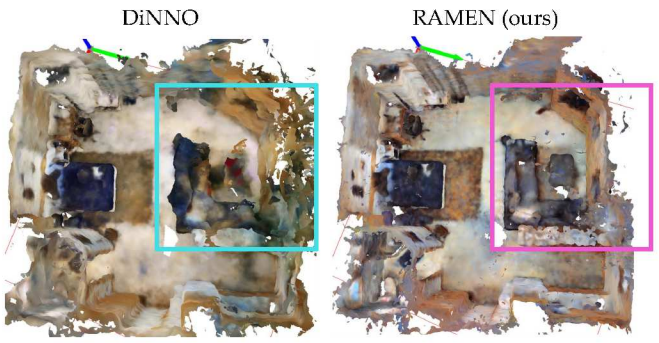

Fig. 3: Comparison of scene reconstructions from DiNNO and RAMEN (ours) on ScanNet "scene0000" with only 50% communication success rate. RAMEN produces more accurate and detailed geometry in the highlighted region.

to a vertex primarily occur when the robot visits its corresponding spatial location [38].

Based on these two observations, *we associate the number of gradient updates a feature grid parameter receives with its epistemic uncertainty*. More updates indicate more frequent visits to the corresponding location, implying lower uncertainty. Due to the correlated nature of parameters in MLP decoders F_τ, F_ϕ and their lack of direct spatial correspondence, we pretrain the decoders using the NICE-SLAM apartment dataset [23] and keep their parameters fixed during mapping. Thus, unlike Co-SLAM, we only optimize the parameters α in the feature grid V_α, resulting in $\boldsymbol{\Theta} = \{\alpha\}$ with n learnable parameters. We flatten $\boldsymbol{\Theta}$ into a $n \times 1$ vector in our computations.

Our proposed frequency-based epistemic uncertainty quantifier is formulated as:

$$\mathbf{u}_i^t = \sum_{k=0}^{t} \operatorname{sgn}\left(\operatorname{abs}\left(\frac{\partial L_{i,k}^{obj}}{\partial \boldsymbol{\Theta}_i^k}\right)\right), \tag{18}$$

where $\mathbf{u}_i^t$, a $n \times 1$ vector, represents the uncertainty associated with each feature grid parameter at iteration t. $L_{i,k}^{obj}$ is the Co-SLAM reconstruction objective loss (6) of agent i at iteartion k; $\operatorname{abs}(\cdot)$ is the absolute value function; $\operatorname{sgn}(\cdot)$ is the sign function returning 1 when the given value > 0 and returning 0 otherwise. $\mathbf{u}_i^t$ tracks the count of non-zero gradient updates for each parameter, establishing a link between the frequency of spatial location visits and their uncertainty. Fig 2 visualizes the uncertainty of the parameters in the coarsest feature grid $V_\alpha^{l=1}$ in an example using our proposed method, demonstrating that our method accurately captures the uncertainty of the neural implicit map.

Since the number of gradient updates could be infinite, we cannot directly use $\mathbf{u}_i^t$ as weights for consensus optimization in IV-A. Simply normalizing $\mathbf{u}_i^{t_i}$ and $\mathbf{u}_j^{t_j}$ independently to a range of $[0, 1]$ would eliminate the proportional relationship between their elements. For example, the maximum value in $\mathbf{u}_i^{t_i}$ might be substantially larger than the maximum in $\mathbf{u}_j^{t_j}$, a difference that would be lost after independent normalization. To preserve these proportional relationships and ensure that the weights reflect the relative uncertainties, we employ a linear scaling and shifting operation. Given $\mathbf{u}_i^{t_i}$ and $\mathbf{u}_j^{t_j}$ for agent i and its neighbor j, we obtain $n \times n$ weight matrices $\mathbf{W}_{ij}$ and $\mathbf{W}_{ji}$ as follows:

$$\mathbf{u}_{sum} := \mathbf{u}_i^{t_i} + \mathbf{u}_j^{t_j}, \tag{19a}$$

$$\epsilon = \frac{\beta_u - \beta_l}{\max(\mathbf{u}_{sum}) - \min(\mathbf{u}_{sum})}, \tag{19b}$$

$$\zeta = \beta_l - \epsilon \cdot \min(\mathbf{u}_{sum}), \tag{19c}$$

$$\mathbf{W}_{ij} = \operatorname{diag}(\epsilon \cdot \mathbf{u}_i^{t_i} + \zeta), \mathbf{W}_{ji} = \operatorname{diag}(\epsilon \cdot \mathbf{u}_j^{t_j} + \zeta), \tag{19d}$$

where β_u and β_l are the user-specified upper and lower bounds of the weights; $\max(\cdot)$ and $\min(\cdot)$ return the maximum and the minimum elements of the input vector; $\operatorname{diag}(\cdot)$ returns a $n \times n$ matrix using the input as its diagonal elements; ϵ (19b) and ζ (19c) are scaling and shifting terms for normalization. The normalization (19) preserves the proportional relationship between the uncertainty values of agents i and j by applying the same ϵ and ζ to both $\mathbf{u}_i^{t_i}$ and $\mathbf{u}_j^{t_j}$. These weight matrices are then incorporated into the weighted decentralized C-ADMM algorithm to effectively leverage information from neighboring models while accounting for their uncertainty.

The entire multi-agent mapping process of RAMEN is summarized in Algorithm 1. All agents use the same initialization $\boldsymbol{\Theta}_{\text{initial}}$ for their neural implicit maps. Note that, instead of the exact minimization in (16), we obtain $\boldsymbol{\Theta}_i^{t+1}$ by taking B steps of stochastic gradient descent, where B is a user-defined constant.

Algorithm 1 RAMEN: Real-time Asynchronous Multi-agent Neural Implicit Mapping

for $i \in \mathcal{V}$ **do**
  $\mathbf{p}_i^0 = 0$
  $\boldsymbol{\Theta}_i^0 = \boldsymbol{\Theta}_{\text{initial}}$
end for
while true **do**
  for $i \in \mathcal{V}$ **do**
    Capture images and add to database R_i
    Send $\boldsymbol{\Theta}_i^t$ and $\mathbf{u}_i^t$ to neighbors N_i
  end for
  for $i \in \mathcal{V}$ **do**
    Compute (19) to get $\mathbf{W}_{ij}^t, \mathbf{W}_{ji}^t$
    for $b \leftarrow 0$ to B **do**
      Sample from R_i
      Gradient descent on (16) to approximate $\boldsymbol{\Theta}_i^{t+1}$
    end for
    Perform dual variable update (17) to obtain $\mathbf{p}_i^{t+1}$
    Compute (18) to get $\mathbf{u}_i^{t+1}$
  end for
end while

V. Results

We evaluate RAMEN's performance through simulation and hardware experiments. For simulation, we use two datasets:

Replica [47], containing synthetic posed RGBD images of various indoor scenes, and ScanNet [48], which offers challenging real-world posed RGBD images of large indoor spaces. We benchmark RAMEN against three state-of-the-art baselines: DSGD [30], DSGT [31], and DiNNO [13], detailed in Section II. To simulate communication disruptions, we allow only a specified percentage of messages to be successfully transferred between agents.

We reconstructed meshes for evaluation by querying the neural implicit maps within a uniform voxel grid and applying the marching cubes algorithm [49]. To assess the accuracy of the reconstructed geometry, we employ the following 3D quantitative metrics, adapted and renamed for clarity from prior neural implicit mapping works [22, 23, 50, 51]:

1) *Artifacts* (cm): The average distance between sampled points from the reconstructed mesh and the nearest ground-truth point. Higher values indicate more reconstruction artifacts. Thus, lower values represent better performance.
2) *Holes* (cm): The average distance between sampled points from the ground-truth mesh and the nearest reconstructed point. Higher values indicate larger holes in the reconstruction. Thus, lower values represent better performance.
3) *Completion Ratio* (%): The percentage of points in the reconstructed mesh with *Holes* under 5 cm, with higher values representing better performance. This metric provides a holistic measure of the captured scene geometry and is our primary evaluation criterion.

The metrics mentioned above can be combined to provide a more detailed picture of performance. For example, a high *Completion Ratio* with high *Holes* indicates small isolated objects missing from the reconstructed mesh.

A. Simulated Experiments

How does RAMEN perform in general? We first investigate RAMEN's general performance with three agents, a 50% communication success rate, and a fully-connected communication graph. We evaluated RAMEN and the baseline methods on four scenes: "office1" and "room1" from the Replica dataset, and "scene0000" and "scene0106" from the ScanNet dataset. For a comprehensive evaluation, we conducted three trials per agent per method on each scene and reported the mean and standard deviation of the performance metrics. As shown in Table I, RAMEN consistently achieves the best overall performance. DiNNO only slightly outperforms RAMEN in one metric on the simple synthetic scene "office1". While DSGD and DSGT perform adequately on the relatively simple Replica dataset, DSGT fails to converge on the more challenging ScanNet dataset under asynchronous communication. Notably, RAMEN surpasses DiNNO in completion ratio by a significant margin, demonstrating its ability to effectively combine reliable information from all agents to reconstruct a complete scene. Furthermore, RAMEN exhibits significantly smaller standard deviations compared to DiNNO, indicating a more stable and robust mapping process. Figure 3 provides a visualization example from "scene0000," where RAMEN reconstructs the highlighted region with higher accuracy than DiNNO. Since DiNNO achieves the best performance among the baseline methods, for the rest of the experiments, we only compared RAMEN to DiNNO.

We also emphasize that RAMEN employs a simple uncertainty quantifier, incurring minimal computation overhead compared to existing NeRF uncertainty methods covered in Section II. For instance, in a representative three-agent experiment conducted on an RTX 4090 GPU, RAMEN runs at 8.58 FPS, while the baseline DiNNO runs at 9.38 FPS.

How does RAMEN perform with severe communication disruptions? Next, we evaluate RAMEN under more challenging communication conditions. We conducted experiments with three agents exploring "scene0169" from the ScanNet dataset, gradually decreasing the communication success rate from 100% (synchronous updates) to 20%. Figure 4 shows that RAMEN preserves the overall structure and details of the scene even at a 20% communication success rate, while DiNNO fails to reconstruct half of the scene. As expected, the performance metrics (shown in Figure 5) deteriorate for both methods as the communication success rate decreases. However, RAMEN exhibits significantly less performance degradation compared to DiNNO. Furthermore, consistent with the previous experiment, RAMEN demonstrates much smaller standard deviations, indicating greater robustness. Interestingly, RAMEN even outperforms DiNNO under synchronous communication. These results highlight RAMEN's resilience under extreme communication conditions.

How does RAMEN scale with more agents? Finally, we investigate the scalability of RAMEN. We evaluated RAMEN and DiNNO on "scene0106" from the ScanNet dataset with 4, 5, and 6 robotic agents under a 50% communication success rate. To assess information propagation through incomplete communication graph, we restricted agents to communicate only with their immediate neighbors. For instance, with 4 agents, the possible communication among agent $0, 1, 2, 3$ are $\mathcal{E} = \{(0, 1), (1, 2), (2, 3)\}$. Figure 6 demonstrates RAMEN's effectiveness in utilizing observations from additional agents. For example, in the highlighted region (cyan box), RAMEN successfully reconstructs the computer on the desk with the extra input from the 6^{th} agent. In contrast, DiNNO struggles to represent the complete scene and produces numerous floating artifacts, leading to high artifact scores, as shown in Figure 7. Figure 7 also demonstrates that RAMEN achieves better scene completion with additional agents, further highlighting the effectiveness of our uncertainty-based weights in combining information from neighboring agents.

B. Real-World Hardware Experiments

To evaluate RAMEN's performance in real-time multi-robot mapping, we used two Turtlebots within the Robot Operating System (ROS) [52]. Each robot was manually controlled to explore half of the environment, capturing RGBD images with its onboard sensor. Figure 1 illustrates the trajectories and areas covered by each robot. In this setup, due to the limited

TABLE I: RAMEN significantly outperforms prior state-of-the-art approaches for multi-agent neural implicit mapping in challenging communication conditions.

Method	Metric	office1	room1	scene0000	scene0106	Avg.
DSGD	*Artifacts* (cm) ↓	6.38±1.76	16.39±0.45	14.59±0.57	100.40±5.02	34.34±1.95
	Holes (cm) ↓	2.70±0.20	2.55±0.02	5.03±0.16	6.35±0.44	4.20±0.21
	Completion Ratio (%) ↑	87.21±0.96	90.56±0.22	74.34±0.82	65.01±1.61	79.03±0.90
DSGT	*Artifacts* (cm) ↓	20.76±8.31	17.66±1.92	N/A	N/A	N/A
	Holes (cm) ↓	9.09±6.28	2.58±0.05	N/A	N/A	N/A
	Completion Ratio (%) ↑	56.47±1.30	90.69±0.41	N/A	N/A	N/A
DiNNO	*Artifacts* (cm) ↓	**2.78±0.34**	5.78±1.50	7.56±2.62	61.29±7.41	19.35±2.97
	Holes (cm) ↓	2.79±0.22	2.33±0.22	4.10±0.39	5.73±0.82	3.74±0.41
	Completion Ratio (%) ↑	85.90±2.55	91.45±1.99	79.75±3.71	64.73±5.94	80.46±3.55
RAMEN (ours)	*Artifacts* (cm) ↓	3.25±0.35	**5.47±0.97**	**5.55±0.35**	**37.67±1.64**	**12.98±0.83**
	Holes (cm) ↓	**1.78±0.04**	**2.01±0.03**	**3.23±0.08**	**3.28±0.11**	**2.57±0.07**
	Completion Ratio (%) ↑	**94.37±0.27**	**94.20±0.32**	**90.72±0.84**	**86.48±0.64**	**91.44±0.52**

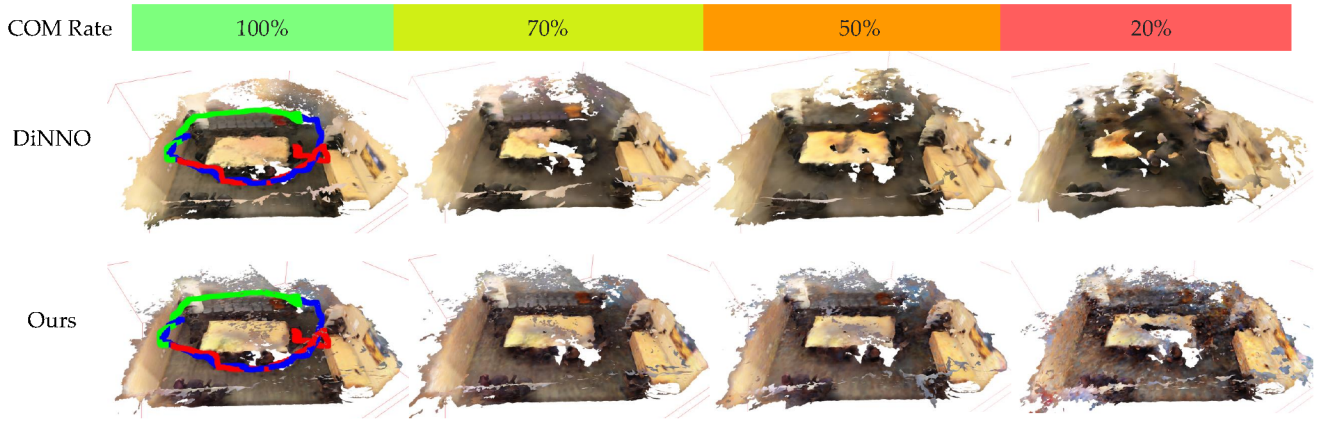

Fig. 4: Reconstruction quality of DiNNO and RAMEN (ours) on ScanNet "scene0169" under varying communication success rates (three-agent trajectories shown in different colors). Even at a 20% success rate, RAMEN preserves scene details, while DiNNO exhibits incomplete or blurry reconstructions.

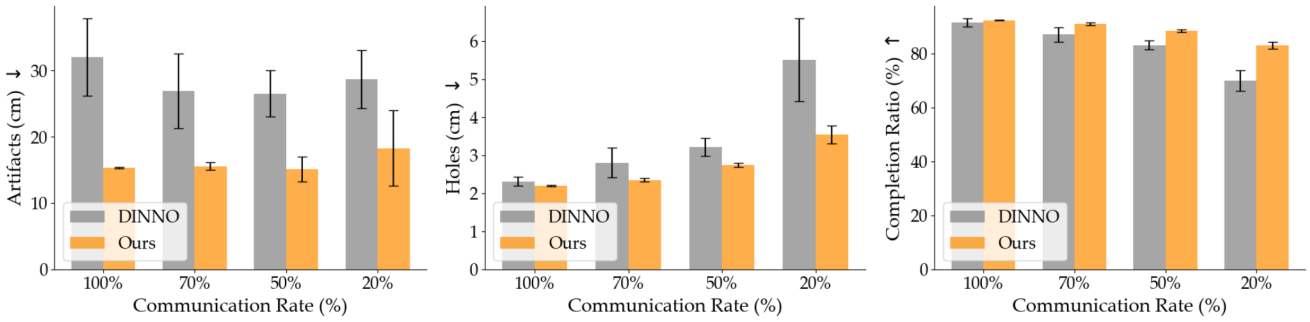

Fig. 5: Evaluation metric scores for DiNNO and RAMEN (ours) on ScanNet "scene0169" under varying communication success rates. While both methods exhibit worse scores with lower communication rate, RAMEN demonstrates greater robustness, maintaining higher accuracy and lower variance than DiNNO, especially in lower communication rates.

range of the stereo camera, successful consensus between the robots is crucial for reconstructing the complete scene. We used a VICON motion capture system to provide ground-truth robot poses. Images and pose data from each Turtlebot were streamed to a separate PC via ROS2 and WiFi for real-time processing, with mapping optimization performed at around 1Hz. Two PCs exhibited substantial differences in processing speed, and *no* synchronization mechanism was implemented

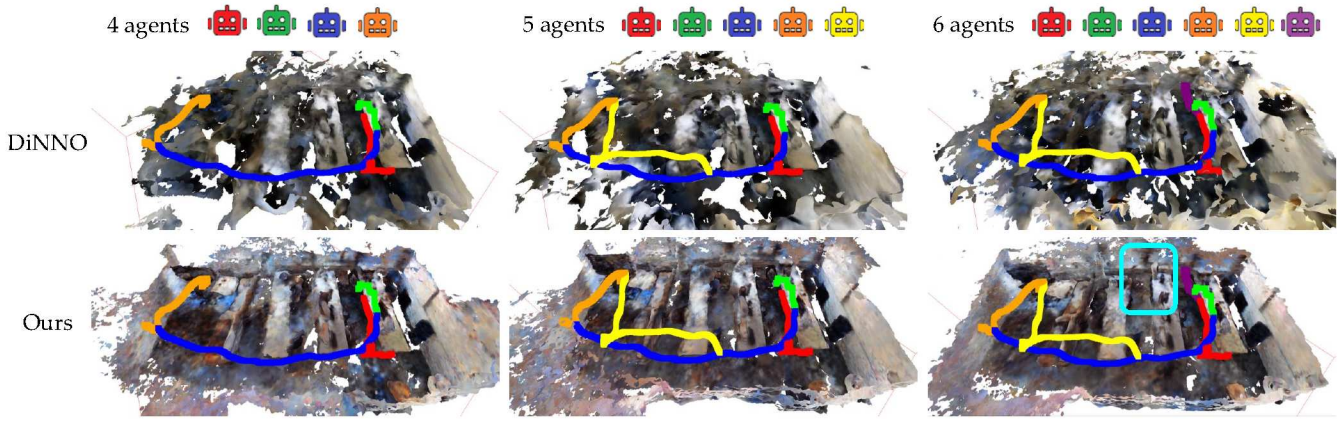

Fig. 6: Reconstruction quality of DiNNO and RAMEN (ours) on ScanNet "scene0106" with varying numbers of agents (trajectories shown in different colors). RAMEN effectively utilizes observations from additional agents to achieve a more complete scene reconstruction, particularly in the highlighted region (cyan box). In contrast, DiNNO exhibits numerous artifacts and incomplete reconstructions.

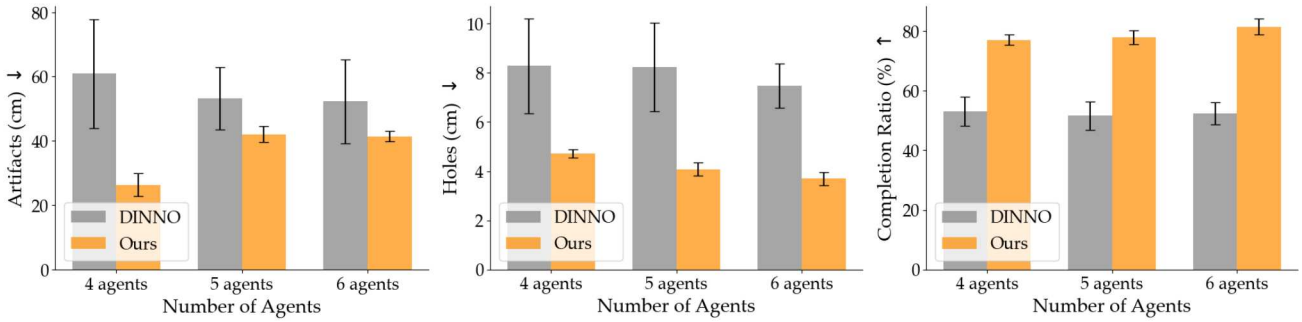

Fig. 7: Evaluation metric scores for DiNNO and RAMEN (ours) on ScanNet "scene0106" with varying numbers of agents. DiNNO exhibits worse artifact scores due to noisy reconstructions. While both methods benefit from additional agents, RAMEN more effectively leverages the increased information to achieve greater improvements in scene completion.

to align their optimization iteration counts. Due to bandwidth limitations, the two PCs could only exchange their neural implicit maps every 30 seconds, resulting in a communication success rate of 3.33%. This asynchronous communication setup, combined with noisy depth images and motion blur, presents an *extremely* challenging scenario that thoroughly tests RAMEN's robustness.

TABLE II: Hardware experiment results

Robot	Method	*Artifacts* (cm) ↓	*Holes* (cm) ↓	*Completion Ratio* (%) ↑
Turtlebot 1	DiNNO	26.43	4.70	63.07
	RAMEN	**6.50**	**4.43**	**78.47**
Turtlebot 2	DiNNO	10.76	12.77	39.83
	RAMEN	**5.87**	**5.15**	**69.66**

Figure 1 shows the overall mapping results. For comparison, we generated a ground-truth reconstruction using a single robot that captured 4x more images while exploring the entire scene. Under these challenging conditions, DiNNO fails to converge, while RAMEN enables both Turtlebots to reach consensus and reconstruct the complete scene. Figure 8 provides a closer look at the reconstructed details. Despite noisy depth images, the overall geometry is accurately captured. The absence of floating artifacts, in contrast to the simulated experiments, is attributed to human-in-the-loop active perception, which helped to minimize such errors. This also explains the higher metric scores in Table II compared to the previous, less challenging simulated experiments.

VI. Limitations

A key limitation of the proposed frequency-based epistemic uncertainty is its inability to differentiate between regions with varying geometric complexity. Regions with complex geometries, such as those containing many small objects or fine details, require more observations to reduce uncertainty and improve reconstruction quality. Another limitation, inherited from Co-SLAM, is the reliance on a keyframe database R_i. This necessitates storing a large number of images onboard and continuously replaying them to prevent catastrophic forgetting, posing challenges for large-scale mapping tasks. Our reconstruction quality also suffer from significant depth

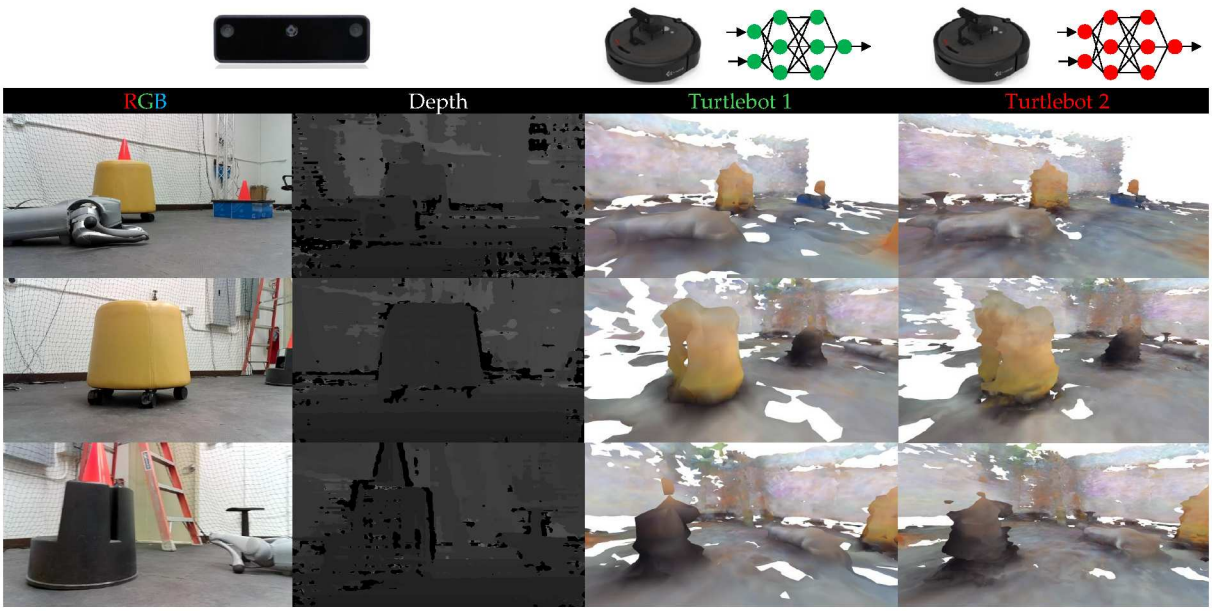

Fig. 8: Comparison of RAMEN reconstructed scene details with corresponding captured images. Each robot only physically visited half of the scene. The complete and similar reconstruction results from both robots indicate successful convergence to consensus. Despite limitations in capturing fine details due to the low quality of depth images, both robots accurately learn the overall scene structure.

estimation noise introduced by the RGBD camera on the Turtlebot. Finally, controlling the robots manually in our hardware experiments may have introduced a human bias, potentially affecting reproducibility. A promising future direction is to integrate active neural implicit mapping techniques, such as those proposed in [36], to automate data acquisition.

VII. Conclusion

This paper introduces RAMEN, a real-time asynchronous multi-agent neural implicit mapping framework. By utilizing a weighted version of C-ADMM, RAMEN handles communication disruptions and enables robots to reconstruct a complete scene collaboratively. The consensus optimization in RAMEN prioritizes information from neighboring robots with lower uncertainty, leading to more accurate and robust mapping. Extensive simulation and hardware experiments demonstrate the effectiveness of RAMEN in challenging scenarios.

Acknowledgments

This work is supported in part by the National Science Foundation, under grants ECCS-2438314 CAREER Award, CNS-2423130, and CCF-2423131.

References

[1] R. Xu, H. Xiang, Z. Tu, X. Xia, M. Yang, and J. Ma, "V2x-vit: Vehicle-to-everything cooperative perception with vision transformer," *ECCV*, 2022.

[2] T. Wang, S. Manivasagam, M. Liang, B. Yang, W. Zeng, and R. Urtasun, "V2vnet: Vehicle-to-vehicle communication for joint perception and prediction," *ECCV*, 2020.

[3] B. Wang, L. Zhang, Z. Wang, Y. Zhao, and T. Zhou, "Core: Cooperative reconstruction for multi-agent perception," *ICCV*, 2023.

[4] M. Zaccaria, M. Giorgini, R. Monica, and J. Aleotti., "Multi-robot multiple camera people detection and tracking in automated warehouses," *International Conference on Industrial Informatics*, 2021.

[5] O. Walker, F. Vanegas, and F. Gonzalez, "A framework for multi-agent uav exploration and target-finding in gps-denied and partially observable environments," *Sensors*, vol. 20, no. 17, 2020.

[6] 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," *Commun. ACM*, vol. 65, p. 99–106, dec 2021.

[7] M. Adamkiewicz, T. Chen, A. Caccavale, R. Gardner, P. Culbertson, J. Bohg, and M. Schwager, "Vision-only robot navigation in a neural radiance world," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4606–4613, 2022.

[8] T. Chen, P. Culbertson, and M. Schwager, "Catnips: Collision avoidance through neural implicit probabilistic scenes," *IEEE Transactions on Robotics*, vol. 40, pp. 2712–2728, 2024.

[9] S. Zhi, T. Laidlow, S. Leutenegger, and A. Davison, "In-place scene labelling and understanding with implicit scene representation," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[10] J. Kerr, C. M. Kim, K. Goldberg, A. Kanazawa, and M. Tancik, "Lerf: Language embedded radiance fields,"

in *International Conference on Computer Vision (ICCV)*, 2023.

[11] R. Egodagamage and M. Tuceryan, "Distributed monocular slam for indoor map building," *Journal of Sensors*, 2017.

[12] K. Ye, S. Dong, Q. Fan, H. Wang, L. Yi, F. Xia, J. Wang, and B. Chen, "Multi-robot active mapping via neural bipartite graph matching," *CVPR*, 2022.

[13] J. Yu, J. A. Vincent, and M. Schwager, "Dinno: Distributed neural network optimization for multi-robot collaborative learning," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1896–1903, 2022.

[14] Y. Deng, Y. Tang, Y. Yang, D. Wang, and Y. Yue, "Macim: Multi-agent collaborative implicit mapping," *IEEE Robotics and Automation Letters*, vol. 9, no. 5, pp. 4369–4376, 2024.

[15] M. Asadi, K. Zareinia, and S. Saeedi, "Di-nerf: Distributed nerf for collaborative learning with relative pose refinement," *IEEE Robotics and Automation Letters*, vol. 9, no. 11, pp. 10527–10534, 2024.

[16] H. Zhao, B. Ivanovic, and N. Mehr, "Distributed nerf learning for collaborative multi-robot perception," 2024.

[17] J. Zhang, "Asynchronous decentralized consensus admm for distributed machine learning," in *2019 International Conference on High Performance Big Data and Intelligent Systems (HPBD&IS)*, pp. 22–28, 2019.

[18] R. Zhang and J. Kwok, "Asynchronous distributed admm for consensus optimization," in *Proceedings of the 31st International Conference on Machine Learning*, vol. 32, (Bejing, China), pp. 1701–1709, 22–24 Jun 2014.

[19] J. Ortiz, A. Clegg, J. Dong, E. Sucar, D. Novotny, M. Zollhoefer, and M. Mukadam, "isdf: Real-time neural signed distance fields for robot perception," *Robotics: Science and Systems*, 2022.

[20] C. Jiang, H. Zhang, P. Liu, Z. Yu, H. Cheng, B. Zhou, and S. Shen, "H_2-mapping: Real-time dense mapping using hierarchical hybrid representation," *IEEE Robotics and Automation Letters*, vol. 8, no. 10, pp. 6787–6794, 2023.

[21] J. Wang, T. Bleja, and L. Agapito, "Go-surf: Neural feature grid optimization for fast, high-fidelity rgb-d surface reconstruction," *3DV*, 2022.

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

[23] 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," *CVPR*, 2022.

[24] H. Wang, J. Wang, and L. Agapito, "Co-slam: Joint coordinate and sparse parametric encodings for neural real-time slam," *CVPR*, 2023.

[25] R. Egodagamage and M. Tuceryan, "Distributed monocular slam for indoor map building," *Journal of Sensors*, 2017.

[26] J. Hu, M. Mao, H. Bao, G. Zhang, and Z. Cui, "Cp-slam: Collaborative neural point-based slam system," in *Advances in Neural Information Processing Systems*, vol. 36, pp. 39429–39442, 2023.

[27] T. Zhang, L. Zhang, Y. Chen, and Y. Zhou, "Cvids: A collaborative localization and dense mapping framework for multi-agent based visual-inertial slam," *IEEE Transactions on Image Processing*, vol. 31, pp. 6562–6576, 2022.

[28] C. Yu, X. Yang, J. Gao, H. Yang, Y. Wang, and Y. Wu, "Learning efficient multi-agent cooperative visual exploration," *ECCV*, 2022.

[29] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[30] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[31] S. Pu and A. Nedić, "Distributed stochastic gradient tracking methods," *Math. Program.*, vol. 187, p. 409–457, may 2021.

[32] Q. Ling, W. Shi, G. Wu, and A. Ribeiro, "Dlm: Decentralized linearized alternating direction method of multipliers," *IEEE Transactions on Signal Processing*, vol. 63, no. 15, pp. 4051–4064, 2015.

[33] N. Sünderhauf, J. Abou-Chakra, and D. Miller, "Density-aware nerf ensembles: Quantifying predictive uncertainty in neural radiance fields," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9370–9376, 2023.

[34] J. Shen, A. Ruiz, A. Agudo, and F. Moreno-Noguer, " Stochastic Neural Radiance Fields: Quantifying Uncertainty in Implicit 3D Representations ," in *2021 International Conference on 3D Vision (3DV)*, pp. 972–981, 2021.

[35] L. Goli, C. Reading, S. Sellán, A. Jacobson, and A. Tagliasacchi, "Bayes' Rays: Uncertainty quantification in neural radiance fields," *CVPR*, 2024.

[36] Z. Yan, H. Yang, and H. Zha, "Active neural mapping," in *Intl. Conf. on Computer Vision (ICCV)*, 2023.

[37] H. Siming, C. D. Hsu, D. Ong, Y. S. Shao, and P. Chaudhari, "Active perception using neural radiance fields," in *2024 American Control Conference (ACC)*, pp. 4353–4358, IEEE, 2024.

[38] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Trans. Graph.*, vol. 41, pp. 102:1–102:15, July 2022.

[39] T. Müller, B. McWilliams, F. Rousselle, M. Gross, and J. Novák, "Neural importance sampling," *ACM Trans. Graph.*, vol. 38, pp. 145:1–145:19, Oct. 2019.

[40] D. Azinović, R. Martin-Brualla, D. B. Goldman, M. Nießner, and J. Thies, "Neural rgb-d surface reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6290–6301, June 2022.

[41] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the linear convergence of the admm in decentralized consensus optimization," *IEEE Transactions on Signal Processing*, vol. 62, no. 7, pp. 1750–1761, 2014.

[42] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "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.

[43] S. W. Fung and L. Ruthotto, "An uncertainty-weighted asynchronous admm method for parallel pde parameter estimation," *SIAM Journal on Scientific Computing*, vol. 41, no. 5, pp. S129–S148, 2019.

[44] Q. Ling, Y. Liu, W. Shi, and Z. H. Tian, "Weighted admm for fast decentralized network optimization," *IEEE Transactions on Signal Processing*, vol. 64, pp. 5930–5942, 2016.

[45] M. Ma and G. B. Giannakis, "Graph-aware weighted hybrid admm for fast decentralized optimization," in *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, pp. 1881–1885, 2018.

[46] C. Zhang, M. Ahmad, and Y. Wang, "Admm based privacy-preserving decentralized optimization," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 3, pp. 565–580, 2019.

[47] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma, A. Clarkson, M. Yan, B. Budge, Y. Yan, X. Pan, J. Yon, Y. Zou, K. Leon, N. Carter, J. Briales, T. Gillingham, E. Mueggler, L. Pesqueira, M. Savva, D. Batra, H. M. Strasdat, R. D. Nardi, M. Goesele, S. Lovegrove, and R. Newcombe, "The Replica dataset: A digital replica of indoor spaces," *arXiv preprint arXiv:1906.05797*, 2019.

[48] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[49] P. M. Neila, "Pymcubes: Marching cubes (and related algorithms) for python," 2013.

[50] M. M. Johari, C. Carta, and F. Fleuret, "ESLAM: Efficient dense slam system based on hybrid representation of signed distance fields," in *Proceedings of the IEEE international conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[51] L. Z. Zhe Xin, Yufeng Yue and C. Wu, "Hero-slam: Hybrid enhanced robust optimization of neural slam," in *ICRA*, 2024.

[52] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022.