RaGNNarok: A Light-Weight Graph Neural Network for Enhancing Radar Point Clouds on Unmanned Ground Vehicles

David Hunt*, Shaocheng Luo*, Spencer Hallyburton, Shafii Nillongo, Yi Li,
Tingjun Chen, and Miroslav Pajic

***Abstract*— Low-cost indoor mobile robots have gained popularity with the increasing adoption of automation in homes and commercial spaces. However, existing lidar and camera-based solutions have limitations such as poor performance in visually obscured environments, high computational overhead for data processing, and high costs for lidars. In contrast, mmWave radar sensors offer a cost-effective and lightweight alternative, providing accurate ranging regardless of visibility. However, existing radar-based localization suffers from sparse point cloud generation, noise, and false detections. Thus, in this work, we introduce RaGNNarok, a real-time, lightweight, and generalizable graph neural network (GNN)-based framework to enhance radar point clouds, even in complex and dynamic environments. With an inference time of just 7.3 ms on the low-cost Raspberry Pi 5, RaGNNarok runs efficiently even on such resource-constrained devices, requiring no additional computational resources. We evaluate its performance across key tasks, including localization, SLAM, and autonomous navigation, in three different environments. Our results demonstrate strong reliability and generalizability, making RaGNNarok a robust solution for low-cost indoor mobile robots.**

I. Introduction

Indoor mobile robots, such as unmanned ground vehicles (UGVs), are increasingly used in homes and commercial spaces, requiring accurate sensing for GNSS-free mapping and navigation in potentially complex environments. Traditionally, these robots rely on lidar and cameras, but both technologies have limitations that hinder their wide adoption on low-cost platforms.

Lidar offers high precision but is expensive (e.g., $800+ for the Livox MID-360), power-intensive, and struggles in visually occluded environments (e.g., smoke, dust). Cameras, while affordable, face challenges in low-light or visually uniform environments and lack depth perception unless combined with additional sensors. Both lidar and camera-based systems require deep learning (DL) models for reliable scene understanding, which may exceed the computational capacity of low-cost robots and fail in tasks that have real-time requirements.

In contrast, mmWave radar provides a low-cost, lightweight alternative with accurate ranging, even in poor visibility. Modern 77 GHz mmWave radars achieve 4 cm range resolution in a compact, low-power form factor, making them ideal for indoor mobile robots. Unlike lidar and cameras, radar can also detect velocity, enabling real-time differentiation between static and dynamic objects, and enhancing situational awareness in complex environments.

While mmWave radar may offer a low-cost alternative for UGV-based mapping, localization, and navigation, several critical challenges have hindered its real-time adoption. First, the angular resolution of typical mmWave radar sensors used on UGVs is limited to 14.3° [1], significantly coarser than lidar's 0.1° resolution [2]. This results in 90% fewer points than even 2D lidar slices, leading to sparse and incomplete environmental representations. Second, radar point clouds are prone to high false detection rates due to multipath interference, where radio waves reflect off multiple objects before returning to the sensor. In indoor environments, we observed that up to *60% of detected points were false*, further complicating localization and mapping.

Existing approaches for mmWave radar-based sensing on UGVs struggle with real-time feasibility and degrade in dynamic environments. Some methods encode radar inputs as images[3]–[5], while others rely on generative adversarial networks (GANs) [6], but these approaches demand high-compute, segmentation-based deep learning models that are impractical for UGVs with limited computational resources. Additionally, no prior work effectively utilizes radar velocity measurements, making it challenging to distinguish static from moving objects—a crucial limitation that hampers mmWave radar-only mapping and reliable navigation.

To overcome these challenges, we introduce RaGNNarok, a lightweight graph neural network (GNN)-based framework that enhances 2D mmWave radar point clouds for real-time UGV navigation. Unlike CNN-based approaches that struggle with irregular, sparse radar data, RaGNNarok directly models point relationships as a graph, enabling spatial feature aggregation while filtering multipath artifacts. By incorporating velocity measurements, our method uniquely discriminates between static and dynamic objects, improving navigation robustness. Additionally, its efficient architecture significantly boosts frame rate, achieving 7.3 ms inference time on a Raspberry Pi 5, making RaGNNarok the first practical GNN-based solution for real-time mmWave radar-based UGV autonomy.

To demonstrate both performance and real-time feasibility, we integrate RaGNNarok into the navigation pipeline on a resource-constrained UGV. Combined with industry—standard ROS2 packages—Nav2 and slam-toolbox ([7]–

*These authors contributed equally to this work.

This work is sponsored in part by the ONR under the agreement N00014-23-1-2206, AFOSR FA9550-19-1-0169 Award, NSF CNS-1652544 and CNS-2211944 awards, and the National AI Institute for Edge Computing Leveraging Next Generation Wireless Networks, Grant CNS-2112562.

The authors are with the Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708 USA (e-mail: {david.hunt, shaocheng.luo, spencer.hallyburton, shafii.nillongo, yi.li, tingjun.chen, miroslav.pajic}@duke.edu).

[9])—we show that our GNN-enhanced mmWave radar framework enables accurate mapping, real-time localization, and robust navigation in complex, dynamic environments. Unlike prior methods that struggle with sparse, noisy radar data, RaGNNarok enhances point clouds in real-time, ensuring precise localization without reliance on high-resolution lidar or vision-based sensors. To the best of our knowledge, this is the first work to achieve real-time localization and navigation on a UGV using enhanced mmWave radar point clouds, demonstrating the practical viability of GNNs for mmWave radar-based autonomy.

This paper is organized as follows. Section Sec. III introduces the framework used to implement the RaGNNarok model. Following, section Sec. IV describes the robust evaluations used to validate RaGNNarok, including real-world case studies on a UGV and comparisons to existing traditional and learning-based methods. Finally, section Sec. V presents the results of our evaluations where we demonstrate the accuracy, computational efficiency, and feasibility of the RaGNNarok model.

II. Related Works

Deep learning (DL) models for mmWave radar sensing. Previous works [4]–[6], [10]–[12] have introduced methods of converting low resolution mmWave radar data into high-resolution 2D and/or 3D lidar-like point clouds through various DL models. However, these models only focused on static environments and either do not generalize well in complex, previously unseen, environments and/or cannot be executed in real-time on computationally-constrained platforms. Furthermore, [13] generated high-resolution 3D reconstructions on indoor environments by using a rotation mmWave radar, but also only focused on static environments and requires significant computational resources to run in real time. By contrast, RaGNNarok allows for operating in complex and dynamic environments while being generalizable and executing in real-time on resource-constrained platforms.

Graph Neural Networks (GNNs). Previously, [14]–[16] have each applied GNNs to mmWave radar point clouds; yet, they focused only on detecting specific objects (e.g., vehicles and people) in radar point clouds. By contrast, RaGNNarok is designed to enhance the point cloud corresponding *only* to the static environment by identifying and filtering out multi-path detections.

Automotive radars. In the automotive domain, previous works have implemented mmWaveradar-based localization [17]–[25], using methods like the cross-correlation of occupancy grid maps [18] and the Fourier-Mellin Transform method [22]. Further, [17], [23], [24] introduced simultaneous localization and mapping (SLAM) pipelines. Yet, all focused on outdoor environments and used *high-resolution* radars such as the CTS350-X [26] with angular resolutions ranging from 1° to 4°. These radars are expensive, large, relatively heavy (6 kg), consume a high amount of power (24 W), and require high data rates (1 Gbps ethernet), making them infeasible for low-cost, resource-constrained UGVs [26].

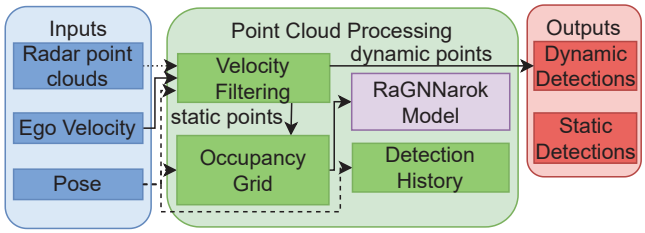

Fig. 1: RaGNNarok block diagram.

Radar odometry. Regarding UGV odometry and collision avoidance, [27]–[33] proposed techniques for mmWave radar-based *odometry* that determines a UGV's relative location with regards to its starting point in an unknown environment; e.g., [27], [30]–[33] used a mmWave radar to estimate the velocity and an IMU sensor to determine the yaw angle and rate of the vehicle. Still, these methods are susceptible to large odometry drifts over time making them infeasible for localization over longer trajectories.

Collision avoidance. Recently, [34], [35] developed mmWave radar-based collision avoidance for indoor UGVs. Yet, both were restricted to static environments; [34] only operates in simple small-scale rectangular spaces, and [35] only plans continuous paths around an environment that avoids collisions.

III. System Design

RaGNNarok utilizes a UGV's estimated position and velocity (e.g., obtained from wheel encoders) to efficiently enhance noisy radar point clouds and differentiate between static and dynamic detections (see Fig. 1). The enhanced point clouds can then be used for downstream tracking, mapping, localization, and navigation tasks in real-time. We now introduce the three main RaGNNarok components.

A. Radar Point Cloud Pre-processing

To optimize performance of the RaGNNarok framework, we perform the following pre-processing steps.

Step ①: Multi-radar point cloud. First, we utilize a "front" and "rear" 77 GHz TI-IWR1843 radar sensor [36], [37] operating at 20 Hz to achieve a 360° field-of-view (FOV). This provides greater situational awareness and significantly improves down-stream mapping, localization, and navigation performance when one radar is obscured (e.g., by objects or people close to the vehicle). Here, each radar detection ($\mathbf{d} = [d_x; d_y; d_z; d_v]$) contains the 3D Cartesian coordinates and relative velocity (i.e., velocity towards or away from the radar) of an object.

While the TI-IWR1843 can perform 3D sensing, we opt to use a 2D configuration (i.e., d_z term is set to 0) as we found the elevation estimates to be inaccurate for UGV scenarios. Thus, both radars are configured to have a range resolution (d_{res}) of 7 cm, maximum sensing range (d_{max}) of 8.56 m, azimuth angular resolution (θ_{res}) of 14.3°, and velocity resolutions (v_{res}) 0.01 m/s. Here, we minimize the processing load by using the TI mmWave SDK [38] to implement a standard radar processing pipeline directly on the radars.

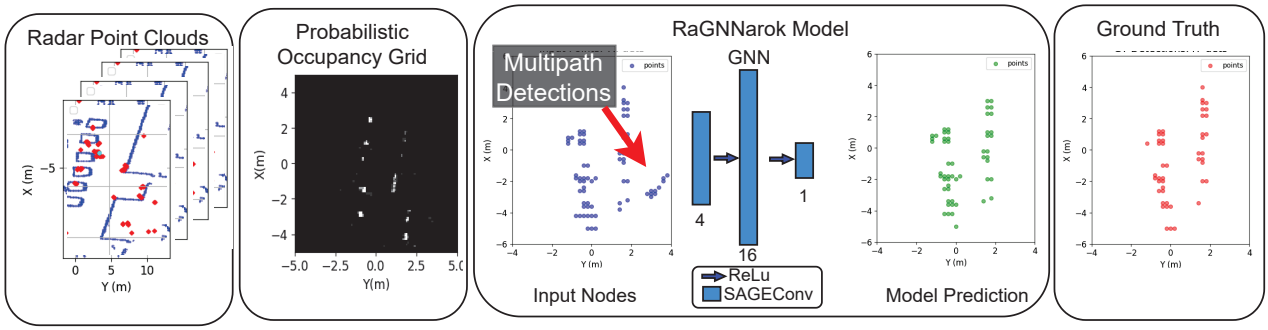

Fig. 2: Overview of the RaGNNarok model architecture

Additionally, we filter ground detections by removing all detections within a range of 1.5 m. The final combined point cloud typically features ∼50 points.

Step ②: Dynamic object detection. In dynamic environments, a significant number of radar detections can correspond to moving objects (e.g., people). Thus, for each new frame of radar data, we identify detections corresponding to dynamic objects by utilizing the UGV's translational velocity ($\mathbf{v}$) and the velocity component of each detection (d_v). For a detection with 2D coordinates $\mathbf{q} = [d_\mathrm{x}; d_\mathrm{y}]$, the relative velocity measured by the radar for a static point can be expressed as $\hat{d}_\mathrm{v} = \langle -\frac{\mathbf{q}}{\|\mathbf{q}\|}, \mathbf{v} \rangle$ Thus, we define dynamic detections as any detection where $|\hat{d}_\mathrm{v} - d_\mathrm{v}| > 0.05$ m/s. Once separated, static detections are then used to generate a probabilistic occupancy grid map while dynamic detections can then be used by other downstream tasks (e.g.; tracking)

Step ③: Probabilistic occupancy grid Given the sparsity of the radar point clouds, we additionally employ a probabilistic occupancy grid to provide recent temporal history to the model. Empirically, we found that an occupancy grid with a cell resolution of 20 cm, range of [-5 m,5 m], and temporal history leveraging the previous 20 radar frames (i.e. 1 s of previous sensing) best balanced between increasing point cloud accuracy while minimizing the required sensing duration. Here, we utilize a UGV's pose (i.e.; position and orientation) estimate to continuously align the occupancy grid with the most recently recorded radar frame.

B. RaGNNarok Model

Graph neural networks (GNNs) encode data as a set of nodes and edges where each node is defined by a set of properties and each edge defines how the nodes are connected. Compared to other methods, graph neural networks are particularly well suited for enhancing radar point clouds because their graph structure allows them to work well with sparse data and spatial representations while also being more robust to noise and false detections [39]. RaGNNarok takes in a radar point cloud and detection probability information to classify each detection as valid or invalid (e.g.; a multipath detection).

Input graph nodes and edges For each radar detection captured by the probabilistic occupancy grid, we define a node ($\mathbf{n} = [d_\mathrm{x}; d_\mathrm{y}; d_\mathrm{z}; p_\mathrm{det}]$) which contains the Cartesian coordinates and current probability of the detection. For the edges between nodes, we use the Pytorch Geometric `radius_graph` module to define the edges between all nodes within a 10 m radius of each other. Here, each edge's value is the euclidean distance between the two corresponding nodes. Compared to previous segmentation approaches, we highlight how this input format significantly reduces the size of the model input data while allowing it to dynamically adapt to point clouds with varying numbers of points.

Model Output. For each radar node in the graph, RaGNNarok classifies each node as valid or invalid. When labeling each node, we define a node as valid if its corresponding radar detection was within 20 cm of a lidar ground-truth detection.

Model Architecture. As shown in Fig. 2, we implemented a light-weight GNN architecture by using a series of three GraphSAGE convolution blocks using the Pytorch `SAGEConv` modules [40]. Compared to other graphical convolution methods, we selected GraphSAGE convolutions because they learn a flexible aggregation function and generalize particularly well beyond the training data [40]. In total, our model only utilizes 705 parameters, compared to recent state of the art (SOTA) works [5] and [3] which utilized ∼7.7 M and ∼17.5 M parameters, respectively. As we show in the following sections, this simpler model significantly reduces the computational time required for each inference.

Loss Function We utilize Binary Cross Entropy (BCE) loss during training as this is a commonly applied loss function in graph neural network training pipelines and is particularly well suited for classification tasks.

Data Augmentations To further improve the robustness of our model as well as to prevent over-fitting during training, we additionally employed the following three data augmentations during the training process. First, a random yaw (i.e., around the z axis) rotation in the range [0°,360°]. Secondly, we applied a random perturbations to the occupancy probability (i.e., p_det) of each node by sampling from a normal distribution with a standard deviation of 0.05. Finally, we randomly perturbed $(d_\mathrm{x}, d_\mathrm{y})$ for each node by independently sampling from a normal distribution with standard deviation of 16 cm.

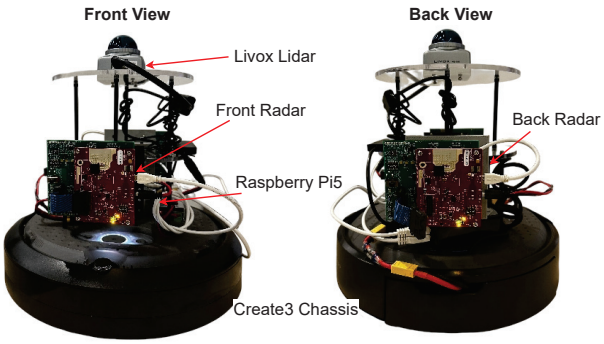

Fig. 3: Experimental platform.

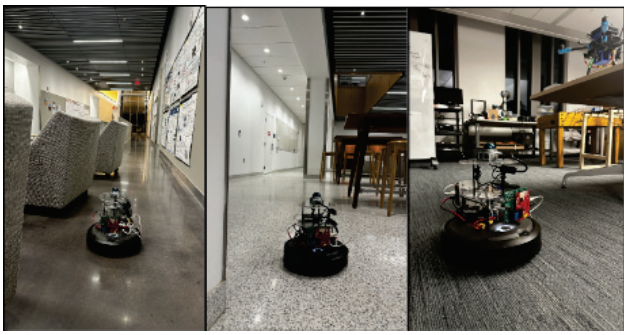

Fig. 4: Indoor test environments 1, 2, and 3 (left to right).

C. Detection History

After enhancing the radar point clouds using the RaGNNarok model, we further increase the density of the final output point cloud by maintaining a short history of detected points. To achieve this, we keep a list of valid detections, and each detection is maintained for a duration of 10 radar frames (i.e., 0.5 s worth of radar frames). As with the probabilistic occupancy grid, we utilize the UGVs pose information to continuously align the point cloud history with the most recently received radar frame.

IV. Evaluation

We rigorously validated the performance of RaGNNarok using a real-world ROS2 prototype UGV equipped compute-limited single board computer. In addition to assessing the quality of the generated point clouds, we also evaluated the performance of downstream localization, navigation, and mapping tasks through a series of offline and real-time experiments. Furthermore, we benchmark our performance against traditional and state of the art DL models. Notably, evaluations were performed across a diverse set of complex environments where people were regularly moving throughout the scene.

A. Experimental Setup

Platform. We used an iRobot Create3 unmanned ground vehicle (Fig. 3) equipped with a Raspberry Pi5 single board computer, TI-IWR1843front and back radars, a Livox MID-360 lidar (used only for collecting ground truth), and Xsens MTi-10 IMU sensor. Radar and lidar data were sampled at a rate of 20 Hz while IMU measurements were sampled at 200 Hz. Also, we align the coordinate frames of the sensors by applying coordinate transformations to the radar data such that both radars have the same coordinate frame as the lidar. The entire design is implemented in a real-time ROS2 framework.

Test environments. We used the three complex environments shown in Fig. 4 to evaluate the performance of RaGNNarok. Each environment featured people moving freely throughout the scene, objects of different shapes, sizes, and materials, and a combination of enclosed and open spaces. Finally, we also ensured that RaGNNarok is robust to changing environments by moving various objects (e.g., chairs, tables, and whiteboards) in all environments between trials.

B. Baseline Methods

RaGNNarok's performance is compared against the following baseline methods.

Naive Radar: Utilizing the combined point cloud from the radar sensors (i.e., no additional filtering or point cloud stacking) to localize the UGV at each frame.

RadarHD [3]: A DL segmentation model that converts raw data from a single radar into 2D lidar-like point clouds. Here, we utilized a TI-DCA1000 to capture raw data from the front radar and RadarHD's pre-trained model to obtain a predicted 2D point cloud to localize the UGV at each frame.

RadCloud [5]: A more recent DL model, optimized for resource-constrained platforms, that converts raw data from a single radar into 2D lidar-like point clouds. Again, we used the TI-DCA1000 to capture raw data from the front radar and RadCloud's pre-trained model to obtain a predicted 2D point cloud for localizing the UGV in each frame.

C. Datasets

For model training, point cloud quality evaluation, and localization accuracy experiments, we recorded the following large-scale datasets.

RaGNNarok Training, Validation, and Test Datasets. For training, validation, and testing of the RaGNNarok model, we captured a total of 57,641 time-synchronized frames by performing 33 independent trials across the three testing environments, covering a total trajectory distance of 1.3 km. For each trial, the UGV was driven along a unique trajectory consisting of irregular turns at varying speeds. All trials contained people moving freely throughout the scene and objects that changed position from trial to trial. For model training and validation, we used 5,400 samples for training and 1,801 samples for validation. Notably, only environment 2 was used for training the model, allowing environments 1 and 3 to be used for assessing performance in new environments. The remaining 50, 440 samples were then used for assessing the quality of the point cloud and for offline localization accuracy experiments.

Baseline datasets As RadarHD and RadCloud both used unique radar configurations, we recorded additional datasets

containing 10,360 (for RadarHD) and 11,244 (for RadCloud) time-synchronized frames following similar trajectories and featuring similar environmental factors as the RaGNNarok dataset. For the Naive radar dataset, we used the existing 57,641 samples recorded for evaluating RaGNNarok's performance.

D. Offline Evaluations

Point cloud quality To evaluate the quality of the generated point clouds versus ground truth lidar scans, we use the commonly used Chamfer and Hausdorff metrics [41]–[43]. However, given that the point clouds generated by each baseline method have varying resolutions and densities, we use the one-way version of these metrics to see how close the radar point cloud is to the ground truth lidar point cloud. Thus, we define the Chamfer distance (CD) as

$$\text{CD}(S_{\text{radar}}, S_{\text{lidar}}) = \frac{1}{2|S_{\text{radar}}|} \sum_{x \in S_{\text{radar}}} \min_{y \in S_{\text{lidar}}} d(x, y) \tag{1}$$

and Hausdorff distance (HD) as

$$\text{HD}(S_{\text{radar}}, S_{\text{lidar}}) = \max_{x \in S_{\text{radar}}} \min_{y \in S_{\text{lidar}}} d(x, y),$$

where $d(x, y)$ denotes the Euclidean distance i.e., $||x - y||_2^2$. Finally, we record the number of points generated by each method to better understand the density of each point cloud.

Localization accuracy Next, we evaluated how well the generated point clouds could be used for downstream localization tasks. To accomplish this, we implemented a radar-inertial localization stack which used an extended kalman filter (EKF) to continuously estimate the global pose of the UGV in a pre-mapped environment [44]. Here, we define the pose estimate as $\hat{\mathbf{P}} = [\hat{\mathbf{p}}, \hat{\Psi}]$, where $\hat{\mathbf{p}} = (x, y)$ and $\hat{\Psi} = \Psi$ denote the UGV's global position and heading, respectively. For the EKF 'predict' step, we employ a non-linear first-order motion model to fuse vehicle velocity and IMU measurements. Then we used the popular iterative closest point (ICP) scan matching to "update" the kalman filter with new measurements. Additionally, erroneous measurements were excluded using the χ^2 anomaly detector (e.g., [45]), with an empirically determined probability of valid data of 0.95.

To construct maps of each test environment, we utilized the popular hector mapping [46] algorithm to generate a 2D map using the lidar sensor. Additionally, we measured the ground truth pose $\mathbf{P}$ for each experiment using a lidar-inertial localization stack which utilized an EKF to fuse IMU, wheel encoders, and lidar ICP measurements. This was benchmarked against a VICON motion capture system showing that the lidar ground truth was always within 20 cm of the VICON ground truth.

Finally, we use the standard absolute trajectory error (ATE) and relative trajectory error (RTE) metrics [47] to assess the localization performance of each method. Here, for the i-th frame, the error metrics for translation (tr) and heading (hd) trajectory errors are defined as

$$\begin{aligned} \text{ATE}_{\text{tr}}(i) &= ||\hat{\mathbf{p}}_i - \mathbf{p}_i||_2 \\ \text{RTE}_{\text{tr}}(i) &= ||(\hat{\mathbf{p}}_i - \hat{\mathbf{p}}_{i-1}) - (\mathbf{p}_i - \mathbf{p}_{i-1})||_2, \end{aligned} \tag{2}$$

TABLE I: Comparison of average inference time where RaGNNarok achieves significant reductions compared to existing works.

Platform	RaGNNarok	RadCloud [5]	RadarHD [3]
Raspberry Pi 5	7.3 ms	178.5 ms	290.7 ms
Desktop (GPU)	1.3 ms	8.2 ms	33.3 ms

$$\begin{aligned} \text{ATE}_{\text{hd}}(i) &= |\hat{\Psi}_i - \Psi_i| \\ \text{RTE}_{\text{hd}}(i) &= |(\hat{\Psi}_i - \hat{\Psi}_{i-1}) - (\Psi_i - \Psi_{i-1})|. \end{aligned} \tag{3}$$

E. Real-time Full Stack Case Studies

To round out our evaluation, we demonstrated the real-time feasibility of the RaGNNarok framework, by performing real-time simultaneous localization and mapping (SLAM) and navigation case studies. We highlight that the RaGNNarok framework was run alongside industry standard SLAM, localization, and navigation stacks in real-time on the Raspberry Pi 5.

Simultaneous Localization and Mapping We used the commonly used `slam-toolbox` ROS2 package to perform SLAM using the point clouds generated by the RaGNNarok model[9]. Due to the lower resolution nature of the radar sensing, we set the map resolution to 10 cm.

Navigation Finally, we used the popular ROS2 `Nav2` package to demonstrate real-time localization and navigation through a mapped environment[7], [8]. Here, we adapted the standard `Nav2` adaptive monte carlo localization (AMCL) and default navigation configuration to utilize the enhanced point clouds generated from the RaGNNarok framework. Noteably, we also used the maps generated in the previous step to successfully demonstrate how our framework can be used to successfully map and then autonomously navigation through an environment.

V. Results

We start with an analysis of the computational time required to run each model, followed by the offline evaluations of point cloud quality and localization accuracy. Then, we conclude with a discussion on our real-time full stack case studies for SLAM and navigation.

A. Computational Time Analysis

To compare the computational time required to make each inference on RaGNNarok versus the other baseline methods, we conducted run-time timing tests on a Raspberry Pi 5 and a Lenovo P360 equipped with a Nvidia T1000 GPU and an Intel i9 CPU. As seen in Table I, RaGNNarok significantly reduces the computational time required to enhance input point clouds. Even on a GPU equipped machine, we achieve an over 6× and 25× reduction in inference time compared to RadCloud and RadarHD, respectively. Additionally, we highlight that RaGNNarok still maintains low inference times on the compute constrained Raspberry Pi 5, enabling real-time operations for our UGV platform.

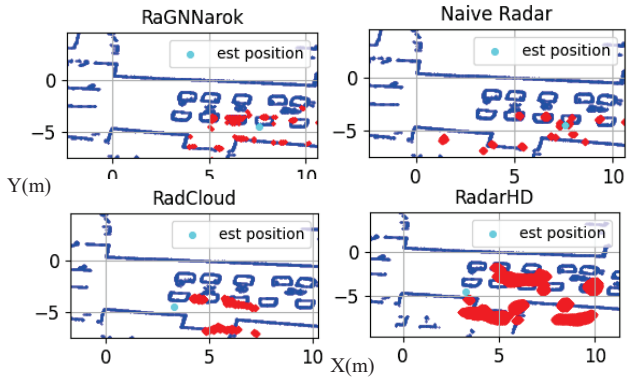

Fig. 5: RaGNNarok generates more accurate point clouds that remove false detections and generalize well to new environments.

TABLE II: Comparison of point cloud quality (* and † indicate results from previously seen and new environments, respectively).

Method	avg points	Chamfer (CD) mean	Chamfer (CD) tail (90%)	Hausdorf (HD) mean	Hausdorf (HD) tail (90%)
RadarHD [3]	5,651	0.71 m	1.17 m	3.61 m	6.31 m
RadCloud [5]	65	0.25 m	0.42 m	0.85 m	1.54 m
Naive radar	65	0.47 m	0.71 m	2.30 m	4.02 m
RaGNNarok*	109	0.28 m	0.41 m	1.41 m	2.13 m
RaGNNarok †	108	0.30 m	0.44 m	1.11 m	2.40 m

TABLE III: Comparison of average trajectory errors. (* and † indicate results from previously seen and new environments, respectively)

Localization Method	Absolute (ATE) Trans(m)	Absolute (ATE) Rot(deg)	Relative (RTE) Trans(m)	Relative (RTE) Rot(deg)
Naive Radar	2.45 m	4.22°	0.013 m	0.056°
RadCloud [5]	0.32 m	2.71°	0.016 m	0.080°
RadarHD [3]	1.31 m	6.21°	0.027 m	0.111°
RaGNNarok*	**0.16 m**	**1.43°**	**0.004 m**	**0.038°**
RaGNNarok†	0.20 m	2.07°	0.004 m	0.036°

B. *Offline analysis*

Point cloud quality. Table II summarizes the point cloud quality assessment, and Fig. 5 presents examples of point clouds generated using each method. As seen in Fig. 5, RaGNNarok produces accurate and dense point clouds that feature minimal false detections. Compared to RadarHD and the Naive radar methods, RaGNNarok significantly improved the chamfer and hausdorf point cloud error metrics, indicating that its point clouds more accurately resembled the environment. Additionally, compared to the RadCloud model, RaGNNarok produced 1.6× more points on average while still generating sufficiently accurate point clouds. Finally, we highlight that the RaGNNarok model generalized well as it showed almost no performance drop when operating in new environments.

Localization. Table III summarizes the average ATE and RTE for each sensing method. Fig. 6 presents an example

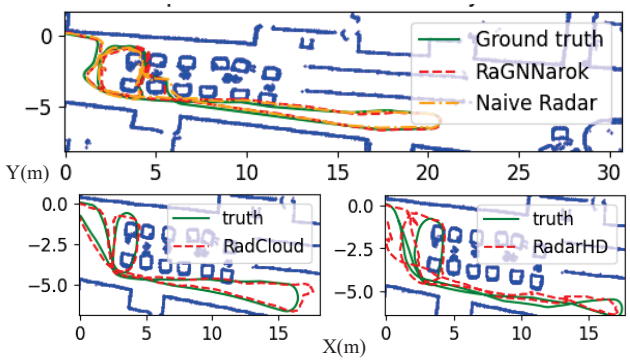

Fig. 6: Comparison of RaGNNarokand Naive Radar trajectory estimates (top), RadCloud estimates (bottom left), and RadarHD estimates (bottom right).

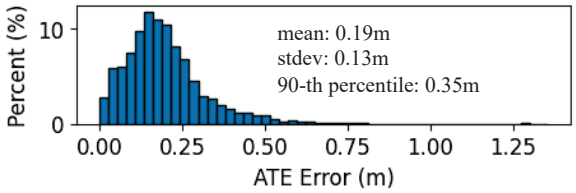

Fig. 7: Histogram of the RaGNNarok ATE errors illustrates how RaGNNarok enables accurate localization performance with minimal errors

of each method's localization performance in environment 1. As shown in Table III, RaGNNarok enables accurate location estimates of the UGV's location with an average ATE of 19 cm. Moreover, 90% of RaGNNarok errors are less than 35 cm (Fig. 7). These results indicate that RaGNNarok maintains consistently accurate trajectory estimates, even in dynamic and complex environments.

Compared to the baseline methods, when considering the average ATE, RaGNNarok achieves 1.6× improvement compared to RadCloud (i.e., the best performing SOTA DL-based method), and 12.8× improvement compared to the naive radar sensing method. Moreover, the naive radar baseline enabled successful localization in most trials, but there were several trials where a large number of false detections led to a complete loss of localization; thus, resulting in the very high average ATE error. Finally, the SOTA DL models somewhat provided accurate localization (especially RadCloud), but with sporadic large localization errors due to inaccurate predictions from the models in more complicated environments. Ultimately, RaGNNarok provided the most accurate and consistent localization of all the considered methods.

C. *Real-time Full Stack Case Studies*

We now present results from our real-time SLAM and navigation case studies. Note video examples of the case studies can be found in our accompanying video submission.

SLAM. Fig. 8 presents an example of a map generated using the RaGNNarok framework. As seen in the figure, the generated map accurately captures the key features of the environment and largely resembles the map obtained

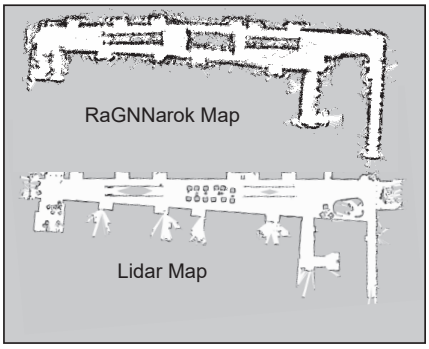

Fig. 8: Comparison of map generated using RaGNNarok point clouds versus lidar point clouds.

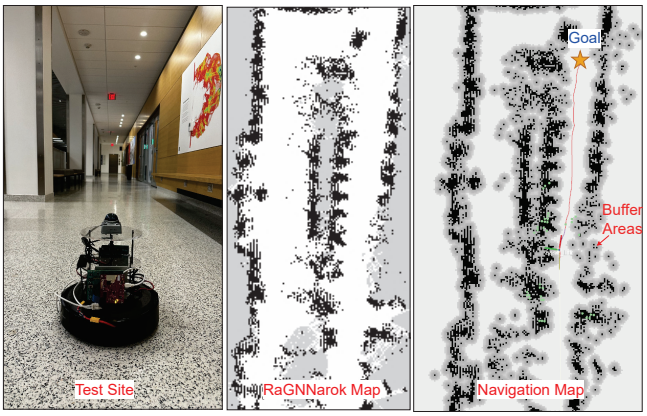

Fig. 9: Navigation trials in test environment 2. The green dots in the right figure denote RaGNNarok-generated radar point clouds, while the red line denotes the planned trajectory.

using the lidar sensors. Additionally, the generated map features relatively few false detections, demonstrating how RaGNNarok enables real-time SLAM, even on compute constrained UGVs.

Navigation. Finally, we demonstrate how the maps generated using RaGNNarok's enhanced point clouds can be used to enable real-time radar navigation. Here, Fig. 9 presents an example of a navigation case study, the map of the environment, and the real-time planned trajectory of the UGV. As seen in our corresponding video submission, the UGV is successfully able to estimate its location and autonomously navigate through a complex environment by utilizing the enhanced point cloud from RaGNNarok.

VI. Conclusion

In this work, we introduced RaGNNarok, a lightweight graph neural network designed to enhance radar point clouds for real-time sensing, localization, and navigation. We evaluated its effectiveness in terms of computational efficiency and point cloud quality, as well as its ability to enable accurate localization, mapping, and collision-free navigation in complex and dynamic environments. Furthermore, we validated its real-world feasibility by deploying RaGNNarokon a ROS2-based unmanned ground vehicle (UGV) running on a Raspberry Pi 5, proving that high-performance radar-based autonomy is achievable on resource-constrained platforms.Looking ahead, we plan to extend RaGNNarok from 2D to 3D point cloud enhancement, further improving spatial awareness and precision. Additionally, we aim to adapt our framework for aerial vehicles, expanding its applicability to a broader range of autonomous systems.

References

[1] S. Rao, "Introduction to mmwave Sensing: FMCW Radars."
[2] "VLP-16 User Manual," Oct. 2018.
[3] A. Prabhakara, T. Jin, A. Das, G. Bhatt, L. Kumari, E. Soltanaghai, J. Bilmes, S. Kumar, and A. Rowe, "High Resolution Point Clouds from mmWave Radar," in *2023 IEEE Int. Conf. on Robot. and Automat. (ICRA'23)*, London, United Kingdom, May 2023, pp. 4135–4142.
[4] E. Sie, X. Wu, H. Guo, and D. Vasisht, "Radarize: Enhancing radar slam with generalizable doppler-based odometry," in *Proceedings of the 22nd Annual International Conference on Mobile Systems, Applications and Services*, 2024, pp. 331–344.
[5] D. Hunt, S. Luo, A. Khazraei, X. Zhang, S. Hallyburton, T. Chen, and M. Pajic, "RadCloud: Real-Time High-Resolution Point Cloud Generation Using Low-Cost Radars for Aerial and Ground Vehicles," in *2024 IEEE Int. Conf. on Robot. and Automat. (ICRA'24)*.
[6] C. X. Lu, S. Rosa, P. Zhao, B. Wang, C. Chen, J. A. Stankovic, N. Trigoni, and A. Markham, "See Through Smoke: Robust Indoor Mapping with Low-cost mmWave Radar," May 2020, arXiv:1911.00398 [eess]. [Online]. Available: http://arxiv.org
[7] S. Macenski, T. Moore, D. V. Lu, A. Merzlyakov, and M. Ferguson, "From the desks of ros maintainers: A survey of modern and capable mobile robotics algorithms in the robot operating system 2," *Robotics and Autonomous Systems*, vol. 168, p. 104493, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092188902300132X
[8] S. Macenski, F. Martin, R. White, and J. Ginés Clavero, "The marathon 2: A navigation system," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.
[9] S. Macenski and I. Jambrecic, "Slam toolbox: Slam for the dynamic world," *Journal of Open Source Software*, vol. 6, no. 61, p. 2783, 2021. [Online]. Available: https://doi.org/10.21105/joss.02783
[10] A. Prabhakara, D. Zhang, C. Li, S. Munir, A. C. Sankaranarayanan, A. Rowe, and S. Kumar, "Exploring mmWave Radar and Camera Fusion for High-Resolution and Long-Range Depth Imaging," in *2022 IEEE/RSJ Int. Conf. on Intell. Robots and Syst. (IROS'22)*, Kyoto, Japan, Oct. 2022, pp. 3995–4002.
[11] P. Cai and S. Sur, "MilliPCD: Beyond Traditional Vision Indoor Point Cloud Generation via Handheld Millimeter-Wave Devices," *Proc. of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 6, no. 4, pp. 1–24, Dec. 2022.
[12] R. Geng, Y. Li, D. Zhang, J. Wu, Y. Gao, Y. Hu, and Y. Chen, "Dream-pcd: Deep reconstruction and enhancement of mmwave radar pointclouds," *IEEE Transactions on Image Processing*, 2024.
[13] H. Lai, G. Luo, Y. Liu, and M. Zhao, "Enabling visual recognition at radio frequency," in *ACM International Conference on Mobile Computing (MobiCom)*, 2024.
[14] P. Svenningsson, F. Fioranelli, and A. Yarovoy, "Radar-pointgnn: Graph based object recognition for unstructured radar point-cloud data," in *2021 IEEE Radar Conference (RadarConf21)*. IEEE, 2021, pp. 1–6.
[15] R. A. Sevimli, M. Üçüncü, and A. Koç, "Graph signal processing based object classification for automotive radar point clouds," *Digital Signal Processing*, vol. 137, p. 104045, 2023.
[16] F. Fent, P. Bauerschmidt, and M. Lienkamp, "Radargnn: Transformation invariant graph neural network for radar-based perception," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 182–191.
[17] M. Adams, J. Mullane, E. Jose, and B.-N. Vo, *Robotic Navigation and Mapping with Radar*. Boston, MA, USA: Artech House, 2012.
[18] L. Narula, P. A. Iannucci, and T. E. Humphreys, "Automotive-radar-based 50-cm urban positioning," in *2020 IEEE/ION Position, Location and Navigation Symp. (PLANS)*, 2020, pp. 856–867.

[19] A. Abosekeen, A. Noureldin, and M. J. Korenberg, "Utilizing the acc-fmcw radar for land vehicles navigation," in *2018 IEEE/ION Position, Location and Navigation Symp. (PLANS'18)*, 2018, pp. 124–132.

[20] T. Ort, I. Gilitschenski, and D. Rus, "Autonomous navigation in inclement weather based on a localizing ground penetrating radar," *IEEE Robot. and Automat. Letters*, vol. 5, no. 2, pp. 3267–3274, 2020.

[21] S. H. Cen and P. Newman, "Precise ego-motion estimation with millimeter-wave radar under diverse and challenging conditions," in *IEEE Int. Conf. on Robot. and Automat. (ICRA)*, 2018, pp. 6045–6052.

[22] A. Pishehvari, U. Iurgel, S. Lessmann, L. Roese-Koerner, and B. Tibken, "Radar scan matching using navigation maps," in *Third IEEE Int. Conf. on Robotic Computing (IRC)*, 2019, pp. 204–211.

[23] R. Rouveure, C. Debain, R. Peuchot, and J. Laneurit, "Robot localization and navigation with a ground-based microwave radar," in *2019 Int. Radar Conf. (RADARConf)*. IEEE, 2019, pp. 1–4.

[24] Z. Hong, Y. Petillot, A. Wallace, and S. Wang, "Radarslam: A robust simultaneous localization and mapping system for all weather conditions," *The Int. J. of Robot. Research*, vol. 41, no. 5, pp. 519–542, 2022.

[25] D. Adolfsson, M. Magnusson, A. Alhashimi, A. J. Lilienthal, and H. Andreasson, "Lidar-level localization with radar? the cfear approach to accurate, fast, and robust large-scale radar odometry in diverse environments," *IEEE Trans. on Robot.*, vol. 39, no. 2, pp. 1476–1495, 2022.

[26] "Clearway technical specifications," march 2023. [Online]. Available: www.navtechradar.com

[27] H. Chen, Y. Liu, and Y. Cheng, "Drio: Robust radar-inertial odometry in dynamic environments," *IEEE Robot. and Automat. Letters*, 2023.

[28] C. X. Lu, M. R. U. Saputra, P. Zhao, Y. Almalioglu, P. P. De Gusmao, C. Chen, K. Sun, N. Trigoni, and A. Markham, "milliego: single-chip mmwave radar aided egomotion estimation via deep sensor fusion," in *18th Conf. on Embedded Networked Sensor Syst.*, 2020, pp. 109–122.

[29] Y. Almalioglu, M. Turan, C. X. Lu, N. Trigoni, and A. Markham, "Milli-rio: Ego-motion estimation with low-cost millimetre-wave radar," *IEEE Sensors J.*, vol. 21, no. 3, pp. 3314–3323, 2020.

[30] Y.-E. Lu, S. Tsai, M.-L. Tsai, and K.-W. Chiang, "A low-cost visual radar-based odometry framework with mmwave radar and monocular camera," *The Int. Archives of the Photogrammetry, Remote Sens. and Spatial Inf. Sciences*, vol. 43, pp. 235–240, 2022.

[31] C. Doer and G. Trommer, "x-rio: Radar inertial odometry with multiple radar sensors and yaw aiding," *Gyroscopy and Navigation*, vol. 12, no. 4, pp. 329–339, 2021.

[32] C. Doer and G. F. Trommer, "An ekf based approach to radar inertial odometry," in *2020 IEEE Int. Conf. on Multisensor Fusion and Integration for Intell. Syst. (MFI'20)*, 2020, pp. 152–159.

[33] Y. S. Park, Y.-S. Shin, J. Kim, and A. Kim, "3d ego-motion estimation using low-cost mmwave radars via radar velocity factor for pose-graph slam," *IEEE Robot. and Automat. Letters*, no. 4, pp. 7691–7698, 2021.

[34] G. M. Reich, M. Antoniou, and C. J. Baker, "Memory-enhanced cognitive radar for autonomous navigation," *IET Radar, Sonar & Navigation*, vol. 14, no. 9, pp. 1287–1296, 2020.

[35] G. Schouten and J. Steckel, "A biomimetic radar system for autonomous navigation," *IEEE Trans. on Robot.*, vol. 35, no. 3, pp. 539–548, 2019.

[36] "IWR1843 Single-Chip 76- to 81GHz mmWave Sensor," Oct. 2018. [Online]. Available: www.ti.com

[37] "IWR1843 Evaluation Module (IWR1843BOOST) Single Chirp mmWave Sensing Solution User's Guide.pdf," May 2020.

[38] "Mmwave-sdk software development kit (sdk)," 2024. [Online]. Available: www.ti.com

[39] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI open*, vol. 1, pp. 57–81, 2020.

[40] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[41] A. Bell, B. Chambers, and H. Butler, "Chamfer," Sept. 2023. [Online]. Available: https://pdal.io/en/latest/apps/chamfer.html

[42] D. Watkins, "Chamfer Distance API," Sept. 2023. [Online]. Available: https://github.com/DavidWatkins/chamfer_distance

[43] M.-P. Dubuisson and A. Jain, "A modified Hausdorff distance for object matching," in *Proc. of 12th Int. Conf. on Pattern Recognition*, vol. 1. Jerusalem, Israel: IEEE Comput. Soc. Press, 1994, pp. 566–568. [Online]. Available: http://ieeexplore.ieee.org/document/576361/

[44] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Trans. of the ASME–J. of Basic Eng.*, vol. 82, no. Series D, pp. 35–45, 1960.

[45] R. S. Hallyburton, A. Khazraei, and M. Pajic, "Optimal myopic attacks on nonlinear estimation," in *IEEE 61st Conf. on Decision and Control (CDC)*, 2022, pp. 5480–5485.

[46] S. Kohlbrecher, O. Von Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *IEEE Int. Symp. on Saf., Secur., and Rescue Robotics*, 2011, pp. 155–160.

[47] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 7244–7251.