RagNAR: Ray-tracing based Navigation for Autonomous Robot in Unstructured Environment

Sunday Amatare, Gaurav Singh, Michelle Samson, and Debashri Roy

Department of Computer Science and Engineering, The University of Texas at Arlington
Email: {sunday.amatare, gaurav.singh, michelle.samson, debashri.roy}@uta.edu

GLOBECOM 2024 - 2024 IEEE Global Communications Conference | 979-8-3503-5125-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/GLOBECOM52923.2024.10901133

***Abstract*—In recent years, there has been a notable surge in interest focused on refining autonomous systems, particularly in advancing robot navigation through the strategic utilization of sensing data. However, this heightened attention has also raised significant privacy concerns. To effectively address these challenges while capitalizing on the benefits of sensing-based robot navigation, this paper introduces a novel concept of Radio Frequency (RF) map creation derived from ray-tracing within a digital twin of an unstructured environment. We present a systematic pipeline for utilizing NVIDIA's Sionna Ray-Tracing tool to generate propagation models of the digital twin generated in Blender by taking inputs from the real world. Through the integration of the RF map, we propose a reinforcement learning based approach to facilitate robot navigation. This integration process enables the formulation of RF propagation models tailored for mobile robots operating within indoor environments. The validation process shows the feasibility of the proposed algorithm in an indoor lab setup with the robot navigating through the various obstacles avoiding any collision. Our work represents a significant advancement towards the practical implementation of robot navigation by harnessing RF propagation data generated through ray-tracing. Through our proposed framework, we contribute to the development of a robust and privacy-preserving approach for robot navigation in autonomous environments.**

***Index Terms*—Channel modeling, RF ray-tracing, reinforcement learning.**

I. Introduction

In previous decades, robots were predominantly designed to tackle monotonous, strenuous, or perilous tasks within industrial settings and high-risk outdoor conditions. However, due to the advancement of technology, the development of service robots is rapidly gaining momentum, leading to the emergence of various definitions aimed at encapsulating their diverse functionalities. For instance, the International Federation of Robotics [1] defines service robots as those that "perform useful tasks for humans or equipment excluding industrial automation". Also, as technology in computer vision, speech recognition, sensors, and artificial intelligence progresses, so do service robots. Breakthroughs in sensors, navigation, and machine learning are enhancing robots' intelligence, mobility, and affordability, enabling them to cater to a broader array of services. These advancements are particularly crucial for tasks in environments, where robots must navigate through populated and unstructured areas [2].

Sensor-Enabled Robot Navigation. The utilization of sensing data has become integral for autonomous systems, especially in the context of service robots tasked with mapping, localizing, and navigating through unstructured environments. In the domain of laser-based mobile robot navigation and localization, the 2D light detection and ranging (LiDAR) sensor emerges as a prevalent choice due to its widespread availability and established reliability. However, a notable drawback of the 2D LiDAR sensor lies in its reliance on a single horizontal scanning line for sensing, limiting its ability to capture detailed spatial information and navigate complex terrains effectively. Conversely, while the 3D LiDAR option offers the potential for more comprehensive environmental perception by capturing spatial information in multiple dimensions, it often comes at a higher cost, which may be a deterrent for certain applications. In contrast, depth cameras have gained popularity due to their affordability, making them an attractive alternative for robot navigation and perception tasks.

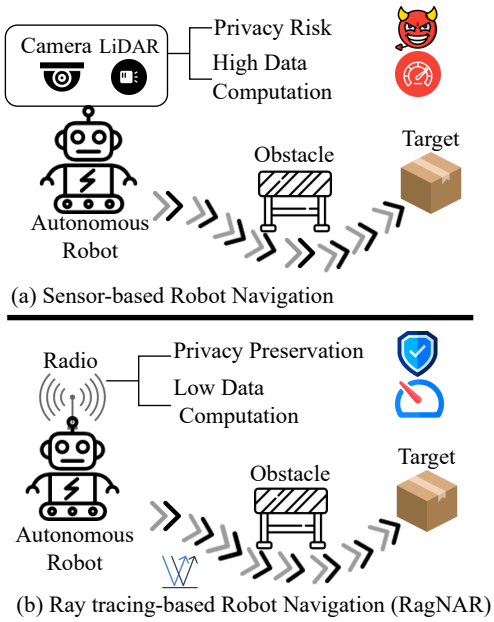


Fig. 1. Comparison of sensor based navigation and RagNAR. We show that RagNAR is privacy preserving and data efficient.

Concerns with Sensor-enabled Robot Navigation. The incorporation of such sensory data into service robots raises critical privacy considerations, with concerns such as the possibility of unauthorized surveillance and the potential for data breaches. These risks underscore the importance of

implementing a privacy-preserving system to safeguard sensitive information collected by these robots, ensuring explicit user consent acquisition, and transparent data usage policies throughout the development and deployment phases of service robots. Such measures are essential for fostering trust among users and stakeholders as highlighted by Gatesichapakorn *et al.* [3]. Furthermore, it is important to note that these sensors come with high computation requirements. This is because they need to provide fine granularity in order to capture the environmental characteristics necessary for accurate robot navigation [3].

Privacy Preservation through Ray-tracing (RT). RF ray-tracing emerges as a proficient technique in selectively capturing and analyzing visual and spatial data from the surrounding environment in the domain of privacy preservation. The concept of rays is readily grasped through our everyday encounters with sunlight. When sunlight penetrates a sizable opening in a wall and enters a room, we observe the propagation of a distinct "ray" moving along a straight path [4]. Ray-tracing has its operation based on geometric optics principles where electromagnetic waves are seen as individual rays that travels straightforwardly until they interact with environment objects producing effects such as reflection, refraction, diffraction, or scattering. Specifically, differential RT (DRT) is an extension of the traditional ray-tracing that captures small changes in the environment for precise sensitivity analysis [5]. It also extracts critical environmental features necessary for generating accurate directives for actuation in complex environment, which can be leveraged to generate robot navigation decisions. Moreover, the generated data using the RT consists of the numeric values of various properties of the rays, hence it has a potential to provide a data efficient representation of the environment.

Motivation and Contributions. State-of-the-art on service robots' navigation mostly use cameras or LiDARs for robots to be able to perceive and understand their surroundings, navigate effectively, and perform tasks in a wide range of applications which raise significant privacy concerns in terms of personal data, raising ethical issues [6], [7]. Keeping this motivation in mind, we propose an RF-based framework, RagNAR, that will ensure privacy in robots navigation and advance technology. As shown in Fig. 1, in RagNAR the robots are equipped with RF devices, which most of the times are integrated within the modern service robots but solely used for communication purposes. These RF devices are used for ray-tracing to generate maps which helps the robots to navigate through unstructured environment. The novel application of RT for robot navigation requires: (a) creation of a digital replica or 'digital twin' that simulates real-world scenarios, (b) a DRT-based propagation model generation within the digital twin of the real world, and (c) a machine learning-based approach which uses the propagation characteristics of the digital twin to generate the robot navigation path. The proposed work needs to be validated on specific type of simulation which involves digital twin creation from real world, integration of the digital twin to the ray-tracing software, generating propagation characteristic in the imported digital twin, and generating data for training the proposed reinforcement learning (RL) based robot navigation. Formally, this paper's contributions are:

C1. We propose a digital twin creation approach which uses direct extraction of real world features.

C2. We propose a methodology for generating the propagation characteristic of the environment by employing DRT on digital twin through RT based softwares.

C3. We propose an RL-based algorithm by designing: (a) a custom environment from the generated propagation data from the ray-tracing tool and (b) a custom reward function to assist the Q-learning agent to generate actuation data for robot navigation.

C4. We validate our overall framework RagNAR using our indoor lab setup as an example scene with publicly available software tools. We release our codebase and generated data for broader community use in [8].

II. Related Works and Motivation

In the domain of robot intelligence and human-robot interaction, service robots inadvertently collect, analyze, and respond to personal data, breaching both virtual and physical boundaries and raising privacy concerns in artificial intelligence environments. Song *et al.* [6] validate a contextualized model acting as a mediator for privacy challenges in consumers' virtual and physical spaces. Findings highlight a significant interplay between perceived risks and benefits, influencing decisions regarding service robot adoption. Arabo *et al.* [7] focus on privacy and security issues within smart homes, developing a system to address concerns such as identity theft, social engineering attacks, and cyber-attack vulnerabilities. However, the evaluation of the severity of those threats is still open-ended. Machine Learning and neural networks, increasingly integrated into various applications, notably enhance adaptability in unfamiliar environments. Surmann *et al.* [9] demonstrate learning-based navigation in unexplored environments using RGB-D and LiDAR sensors, improving navigation robustness. Similarly, Kastner *et al.* [10] and Xie *et al.* [11] utilize the Deep Q-Network algorithm for robot navigation due to its faster convergence in discontinuous action spaces. Lu and Huang [12] apply Deep Reinforcement Learning (DRL) to wheeled robots in unknown environments using 2D Lidar data.

Motivation for Designing RagNAR: All the mentioned works on robot navigation are burdened with data privacy risks and computation of data from high definition sensors such as RGB-D and LiDAR. In this paper, we propose a privacy-preserving, reinforcement learning-based framework, RagNAR, for robot navigation in unstructured environment leveraging RF ray-tracing.

III. System Model and Problem Formulation

In this section, we first formally state the problem. We then present the system architecture of RagNAR.

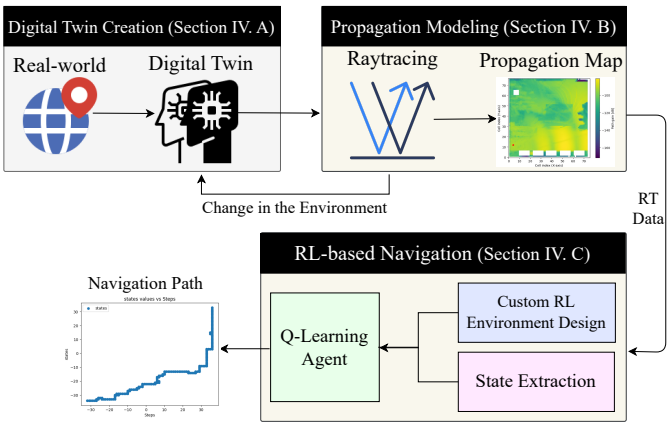


Fig. 2. Overview of the proposed system architecture. RagNAR continuously monitors the environment. In *Module 1*, we use the instantaneous sensor information to generate the digital twin. If a significant change in environment is detected (*Module 2*), we update the *Module 3* which eventually updates path planning policy by optimizing the navigation (*Module 3*).

A. Problem Formulation

We consider robot R in the environment with O obstacles. In our setting, the robot is equipped with RF receiver and edge devices. We have a transmitter in the environment which is transmitting in the 2.4GHz band. The RF receivers listen to the 2.4GHz band and perform ray-tracing. Each edge device of the robot runs a decision algorithm from the ray-tracing data and generates the next coordinate for navigating through the environment (see Fig. 1). The path planning policy targets to minimize the number of steps reaching to the target, while avoiding the obstacles. We use reinforcement learning algorithm to find the optimum policy that meets the above requirements. Thus, given a path planning policy θ, we formulate our objective as:

$$\underset{\theta}{\text{Maximize:}} \quad \underset{\theta}{\mathbb{E}}\left[f^{\theta}_{\mathcal{R}}(s_t, a_t)\right], \tag{1a}$$

$$\text{s.t} \quad a_t = \theta(s_t), \tag{1b}$$

$$\sum_{t=1}^{\mathcal{T}} \varphi_t = 0. \tag{1c}$$

Here, s_t and a_t, denote the state and action for the robot R at step t. The action space includes movement either in the X or in the Y directions. Moreover, $f^{\theta}_{\mathcal{R}}$ denotes a reward function, which is a function of state and action. Finally, φ_t is a Boolean predicate, with φ_t to be 1 if collision happens, and 0 otherwise, that ensures avoiding collisions.

B. System Architecture in RagNAR

An overall view of our framework is shown in Fig. 2 and consists of three main modules as follows:

- **Digital Twin Creation** *(Module 1)*: We create a digital twin by directly extracting the real world features to digital world via sensors. While the transmitter is placed at the roof area, the receiver (robot) is placed on the floor of the room alongside obstacles (details in Sec. IV-A).
- **Propagation Modeling** *(Module 2)*: The autonomous robot receives the instantaneous local ray-tracing data from the transmitting device in the digital twin and acts appropriately to detect the obstacles in close and distant ranges, respectively (details in Sec. IV-B).
- **RL-based Navigation** *(Module 3)*: We design a RL-based algorithm that leverages the ray-tracing data and the propagation characteristics from the digital twin for the robot navigation (details in Sec. IV-C).

IV. RagNAR Framework

In this section, we discuss different steps and components of our proposed framework.

A. Module 1: Digital Twin Creation

The digital twin creation in RagNAR consists of the sensor based scanning of the real-world and 3D map design. In the RagNAR, we consider properties with respect to map precision and radio propagation properties. While we integrate several specific key metrics in the RagNAR, our baseline can be extended in the future to different environmental setup. We initialize the created digital twin as $\mathcal{E} = f(\text{map}, \text{O}, \rho)$. Here, map and O denote the imported Blender [13] map and present structures or obstacles for the twin $\mathcal{E}$, and ρ are number of allowed reflections for the created twin. We characterize present objects in the twin with $\text{O}_u = \{(x_k, y_k, d_{0,k}, d_{1,k}, d_{2,k})\}_{k=1}^{N^{\text{obs}}}$, where N^{obs} is the number of structures or obstacles that are present in the twin $\mathcal{E}$. Moreover, $(x_k,\ y_k)$ are the obstacle coordinates on the map, whereas $d_{0,k}$, $d_{1,k}$, $d_{2,k}$ represent the height, width, and length of the obstacle-k.

It's important to note that the sensors involved in generating the 3D scan of the real world are utilized solely for the creation of the digital twin. They are not involved in generating the robot navigation decisions discussed in the subsequent sections.

B. Module 2: Propagation Modeling

We use the open source Sionna RT [14] tool to generate the propagation characteristics of the created digital twin $\mathcal{E}$ by employing RF ray-tracing. For a given transmitter TX, a propagation map is a rectangular surface with arbitrary orientation subdivided into rectangular cells of size $|\mathcal{C}|$. A parameter η controls the granularity of the map. The propagation map associates with every cell $(\mathcal{C}_i, \mathcal{C}_j)$. In RagNAR, for every ray n intersecting the propagation map cell $(\mathcal{C}_i, \mathcal{C}_j)$, the channel coefficients Θ, phase shifts ϕ, the azimuth angles of departure α_d and arrival α_a, zenith angles of departure ζ_d and arrival ζ_a are computed.

C. Module 3: RL-based Navigation

Once the propagation modeling is performed for the digital twin $\mathcal{E}$, we use reinforcement learning algorithm to solve Eq. 1 and obtain the optimum policy for navigating robot R. We use the Q-learning algorithm for performing the reinforcement learning to identify the policy for optimum path planning as shown in Fig. 3. The agent takes as input the state array s_t and outputs the action a_t following the policy θ.

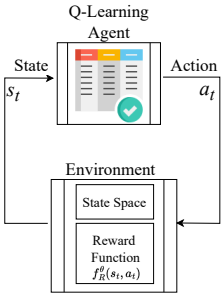


Fig. 3. Overview of the proposed Q-learning agent. We design the state and reward space to optimize navigation.

Table I: The reward function of the proposed Q-learning algorithm.

Symbol	Reward Description	Reward Value
$r^t_{d,d'}$	Relative distance to target	$-(\lvert Tg^t_x - R^t_x\rvert^2 + \lvert Tg^t_y - R^t_y\rvert^2)$
r^t_{arr}	Reaching target	100K
r^t_{obs}	Distance to obstacle	-100K (if $(\lvert Tg^t_x - R^t_x\rvert^2 + \lvert Tg^t_y - R^t_y\rvert) = 0$) 0 (otherwise)

1) Custom RL Environment Design: An RL environment represents the context in which an RL agent functions, serving as a simulated model with which the agent interacts by executing actions, receiving rewards or penalties, and transitioning to new states accordingly. This environment furnishes the agent with its initial state and updates it to the current state following each action, while also specifying the reward linked to each state-action pair. However, given the novelty of our proposed RT-based robot approach, custom RL environment must be generated to accurately capture the propagation characteristics of the digital twin $\mathcal{E}$. As depicted in Fig. 3, the custom RL environment encapsulates both the state space and reward function, is elaborated in the subsequent sections.

2) State Extraction: The state space is extracted from the Module 2. We define the state space s_t of the robot R at step t as: $s_t = \{R^t_x, R^t_y, \zeta^t_d, \alpha^t_d, \zeta^t_a, \alpha^t_a, |\Theta|^t, \phi^t\}$, where R^t_x is the X coordinate of the robot location, R^t_y is the Y coordinate of the robot location, ζ^t_d is zenith angle of departure, α^t_d is azimuth angle of departure, ζ^t_a is zenith angle of arrival, α^t_a is azimuth angle of arrival, $|\Theta|^t$ is channel coefficient magnitude, and ϕ^t is phase shift at step t. The overall state array has 8 elements, encapsulating both RF propagation characteristics and navigation elements.

3) Q-Learning Agent Design: **Action Space.** Given the state array at the step t, the Q-learning model (θ) takes the next action as: $a_t = \theta(s_t)$. The action $a_t = \{X-, X+, Y-, Y+\}$, representing traversing either in X direction ($X-$ and $X+$) or Y direction ($Y-$ and $Y+$) to reach the next state's X and Y coordinate (R^{t+1}_x, R^{t+1}_y) by the robot R at step $t+1$.

Reward Function. We define the overall reward as (details are at Table I):

$$f^\theta_R(s_t, a_t) = r^t_{d,d'} + r^t_{arr} + r^t_{obs}. \quad (2)$$

Here, s_t and a_t denote the state array and action space of the robot R respectively at timestep t. The reward is computed following the policy θ, encapsulating the Q-learning agent.

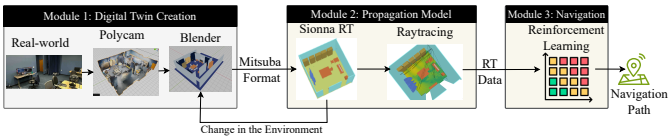


Fig. 4. Overview of various used tools for implementing RagNAR framework.

Intuitively, the relative distance reward ($r^t_{d,d'}$) is increased when the robot gets closer to the target (target coordinates: (Tg^t_x, Tg^t_y) at step t) and vice versa. On reaching the goal state the robot achieves the reward of $100K$. On the other hand (r^t_{obs}) punishes the robot when colliding with any obstacle with a reward of $-100K$.

Model Training. At the training phase, the agent interacts with the environment by observing the reward on taking an action given the current state and stores the observations in a table. The observation includes the reward received on taking an action (a_t) in the current state (s_t) along with reward (r_t) observed in the next state (s_{t+1}) discounted with a factor of γ. The q-values in the table for each state-action pairs are calculated using the Bellman equations [15].

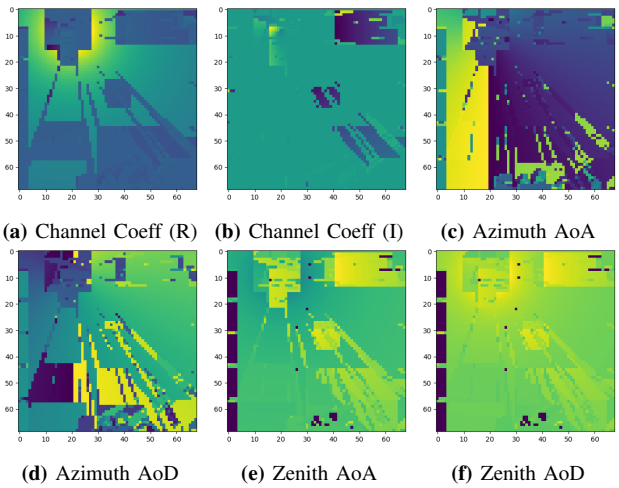
V. Experiments

Experimental Platform. We implement RagNAR using: (a) an Intel i7 system with Ubuntu integrating Polycam [16], Blender LTS v3.6.12 [13], and NVIDIA's Sionna RT [4] for *Module 1* and *2* and (b) Rayzen 9 system on Python with numpy, regex, matplotlib libraries for the *Module 3*. The overall implementation includes digital twin creation, subsequent ray-tracing and propagation characterization using Sionna RT, and reinforcement learning for robot navigation, as shown in Fig. 4.

Evaluation Metrics: To measure the performance of the training of the robot navigation, we evaluate the total number of collisions at each iteration.

A. Setup: Digital Twin Creation

We create the digital twin from our physical indoor laboratory environment, as shown in the first block of Fig. 4. By harnessing Polycam's [16] LiDAR-based 3D scanning and Blender tools, we digitally recreate an authentic indoor environment from the real world including scene objects of different radio material properties. The inclusion of obstacles in the digital twin is important for providing information about the environment, guiding path planning and decision-making as effective obstacle detection, avoidance, and understanding are essential capabilities for autonomous robots operating in diverse and dynamic environments. This digital twin is exported from Blender in `Mitsuba 3 .xml` file format which is responsible for rendering the scene and importation to Sionna RT [4] for propagation modeling. Note that the integration of Polycam and Blender is solely for the digital twin generation and is not involved in making decisions for robot navigation. The details of the digital twin creation is presented in our previous work [17].



(a) Channel Coeff (R) **(b)** Channel Coeff (I) **(c)** Azimuth AoA

(d) Azimuth AoD **(e)** Zenith AoA **(f)** Zenith AoD

Fig. 5. The heatmap of various features of the generated data. The AoA and AoD represents angle of arrival and angles of departure, respectively (details in Sec. IV-B). The "R" and "I" in (a) and (b) signify *real* and *imaginary* part of the channel coefficient data.

B. Setup: Propagation Modeling

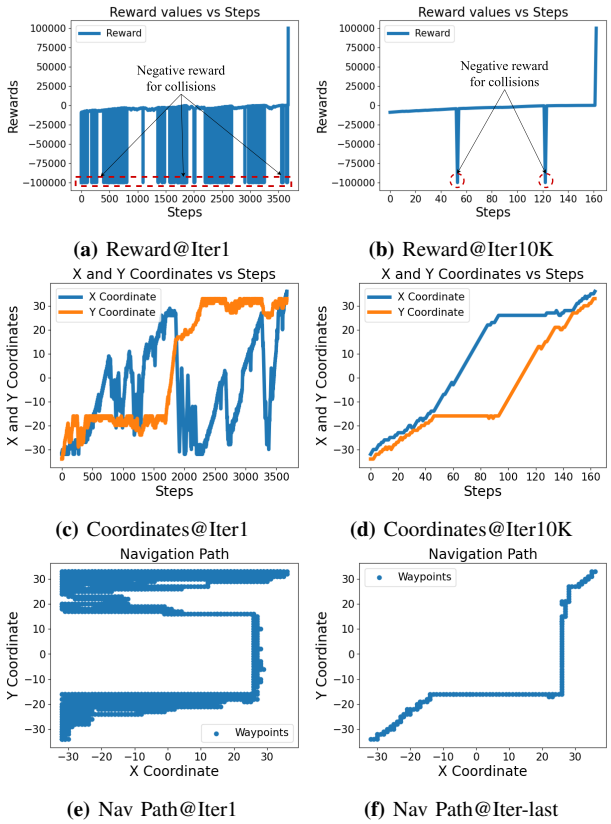
The propagation modeling process in Sionna RT commences with the integration of Sionna with the *Low Level Virtual Machine (LLVM) toolchain* to ensure smooth compatibility. This integration guarantees seamless scene loading, transmitters and receivers creation and configuration, replicating real-world characteristics within the scene. For accurately modeling RF propagation, signal behavior, and communication performance in the scene, Sionna categorizes all the scene components as *radio materials* and their *material properties*. The transmitter and receiver antennas are configured as 8×2 `tr38901` [14] with dual polarization and 1×1 *diapole* planner array, respectively. We calculate propagation paths using the `compute_paths` function in Sionna to generate ray-traced data and produce propagation map by setting `max-depth = 5` and `num-samples = 200`. Overall, our Sionna based implementation ensures improved computational efficiency by incrementally updating the ray paths based on changes in the receiver (or robot) location [14]. The generated propagation map visually depicted in Fig. 8 (d), where the obstacles in the physical world yields to lower signal strength regions in the plot.

Dataset Generation: Following propagation modeling, a dataset is extracted comprising channel coefficients Θ, phase shifts ϕ, azimuth angles of departure α_d and arrival α_a, as well as zenith angles of departure ζ_d and arrival ζ_a for each cell of the generated propagation map. Various statistics of the dataset are presented in Fig. 5. The dataset encompasses 4692 rows, equating to 4692 states, with a total size of 632KB.

Remark 1. *The overall generated ray-traced data, representing the digital twin, is $< 1MB$, which enables the Q-learning agent to have a faster training and inference.*

C. Performance: RL-Based Navigation in RagNAR

In the last module of RagNAR, we train the proposed Q-learning agent on the generated dataset.



(a) Reward@Iter1 **(b)** Reward@Iter10K

(c) Coordinates@Iter1 **(d)** Coordinates@Iter10K

(e) Nav Path@Iter1 **(f)** Nav Path@Iter-last

Fig. 6. The training performance of the proposed Q-learning agent in RagNAR framework. The first column shows various results at the iteration 1 and the second column with the last iteration, i.e., $10K$.

Q-Learning Training: We train the agent for $10K$ iterations using the Q-learning algorithm. For each iteration we took the number of steps as until the agent reaches the goal location from the starting location. We set the starting location of the robot at one corner of the grid and the ending location as the other corner of the grid. The policy θ is trained by updating the q-values within the Q table during training. We follow an ϵ-greedy policy for learning the q-values with exploration rate ϵ of 0.1. We use the Bellman optimality equations [15] for updating the q-values where each q-value is calculated at the discounted sum of rewards with discount factor of 0.9. We show the trend of reward values, X and Y coordinates, and the robot navigation path on the grid for the first and last training iteration in Fig. 6.

Additionally, Fig. 6 presents the variation in the total number of collisions across training iterations. Initially, the total number of collisions is notably high, gradually diminishing as the robot learns the optimal path towards the goal, thereby avoiding collisions and receiving negative rewards for such occurrences. However, the total number of collisions never reaches 0, as the robot continues to explore potential paths utilizing an ϵ-greedy policy with an exploration rate ϵ set at 0.1. This exploration strategy ensures that the robot maintains a level of exploration even as it learns, thereby preventing convergence to a single, potentially suboptimal path. The total training time in for the $10K$ iteration is $\sim 30s$.

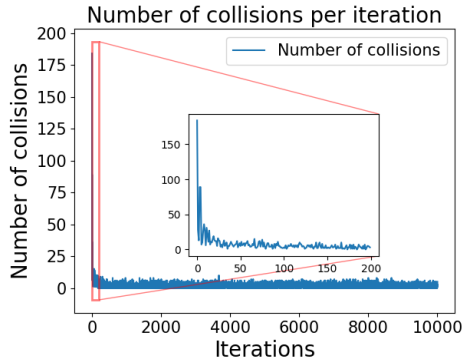
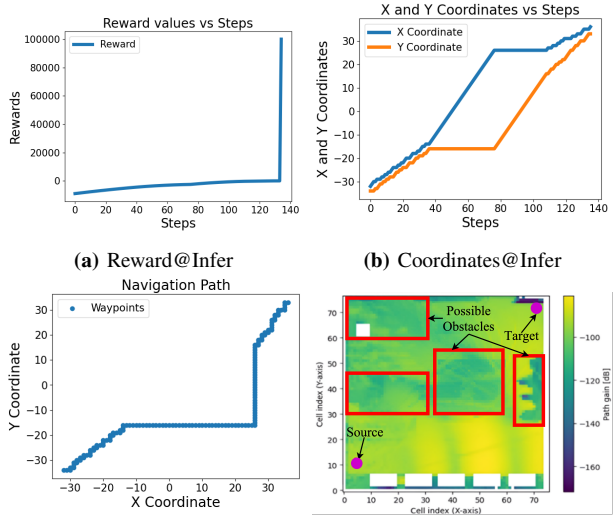


Fig. 7. The number of collision decreases over the training iteration and saturates at 200th iteration.



(a) Reward@Infer **(b)** Coordinates@Infer

(c) Nav Path@Infer **(d)** Propagation Model

Fig. 8. The inference performance of the proposed Q-learning agent in RagNAR framework. The reward plot in (a) shows no collision during inference (no negative reward). In (d) we present the propagation model of the scene where the obstacles can be perceived through the lower signal strength in the region. The source and target of the robot path is also shown in (d). It is evident that the generated robot path in (c) is avoiding the obstacle regions of (d). The scales of the both X and Y axes are same for all the plots.

Observation 1. *We observe a high number of collisions during the beginning of the training, which decreases over iterations and saturates around 200 iterations (see Fig. 6).*

Q-Learning Inference: During inference, we consistently observe no collisions across multiple experiments. Fig.8 illustrates the statistics from one such experiment. In all cases, the robot successfully reaches the target without colliding with any obstacle, as depicted in Fig. 8(a) and (c), which also can be perceived through mapping the Fig. 8 (c) to the propagation model Fig. 8 (d). Moreover, the total inference time averages at 5.8ms, indicating efficient processing and decision-making by the system.

Observation 2. *We observe the robot is reaching the target without encountering any collision following a path by avoiding obstacles (see Fig. 8 (c) and (d)).*

VI. Conclusion

This paper introduces RagNAR, an innovative system designed for robust and privacy-preserving robot navigation utilizing RF ray-tracing. Our approach entails creating a digital twin of the physical environment, conducting ray-tracing-based propagation modeling, and implementing a reinforcement learning method to derive optimal robot navigation paths from the ray-traced data. Through extensive experimental validations, we illustrate that our proposed system, RagNAR, achieves collision-free robot navigation with minimal training data ($< 1MB$). Our future research endeavors will explore advanced reinforcement learning techniques, such as deep neural networks (DQN) and deep deterministic policy gradient (DDPG), to enhance training efficiency.

References

[1] I. I. F. of Robotics, "Service robots," 2024, last accessed 6 April 2024. [Online]. Available: https://ifr.org/service-robots

[2] G. R. Collins, "Improving human–robot interactions in hospitality settings," *International Hospitality Review*, vol. 34, no. 1, pp. 61–79, 2020.

[3] S. Gatesichapakorn, J. Takamatsu, and M. Ruchanurucks, "Ros based autonomous mobile robot navigation using 2d lidar and rgb-d camera," in *2019 First international symposium on instrumentation, control, artificial intelligence, and robotics (ICA-SYMP)*. IEEE, 2019, pp. 151–154.

[4] J. Hoydis, F. A. Aoudia, S. Cammerer, M. Nimier-David, N. Binder, G. Marcus, and A. Keller, "Sionna rt: Differentiable ray tracing for radio propagation modeling," *arXiv preprint arXiv:2303.11103*, 2023.

[5] J. Hoydis, F. A. Aoudia, S. Cammerer, F. Euchner, M. Nimier-David, S. ten Brink, and A. Keller, "Learning radio environments by differentiable ray tracing," 2023.

[6] B. Song, H. Xu, W. Hu, Y. Li, and Y. Guo, "How to calculate privacy: privacy concerns and service robots' use intention in hospitality," *Current Issues in Tourism*, pp. 1–17, 2023.

[7] A. Arabo, I. Brown, and F. El-Moussa, "Privacy in the age of mobility and smart devices in smart homes," in *International Conference on Privacy, Security, Risk and Trust*. IEEE, 2012, pp. 819–826.

[8] https://github.com/TWIST-Lab/SionnaNetworkedRobotics.

[9] H. Surmann, C. Jestel, R. Marchel, F. Musberg, H. Elhadj, and M. Ardani, "Deep reinforcement learning for real autonomous mobile robot navigation in indoor environments," *arXiv preprint arXiv:2005.13857*, 2020.

[10] L. Kästner, C. Marx, and J. Lambrecht, "Deep-reinforcement-learning-based semantic navigation of mobile robots in dynamic environments," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2020, pp. 1110–1115.

[11] L. Xie, S. Wang, A. Markham, and N. Trigoni, "Towards monocular vision based obstacle avoidance through deep reinforcement learning," *arXiv preprint arXiv:1706.09829*, 2017.

[12] Z. Lu and R. Huang, "Autonomous mobile robot navigation in uncertain dynamic environments based on deep reinforcement learning," in *2021 IEEE International Conference on Real-time Computing and Robotics (RCAR)*. IEEE, 2021, pp. 423–428.

[13] B. O. Community, *Blender - a 3D modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. [Online]. Available: http://www.blender.org

[14] N. Inc., "Ray tracing," 2024, last accessed 6 April 2024. [Online]. Available: https://nvlabs.github.io/sionna/api/rt.html

[15] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[16] Polycam Inc., "Polycam (Version 3.3) [Mobile application software]," , 2024.

[17] S. Amatare, M. Samson, and D. Roy, "Testbed design for robot navigation through differential ray tracing," in *2024 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, 2024, pp. 173–174.