Swarm Robotics for Area Exploration Using Q- Learning

Kathirvelan M
Amrita School of Artificial Intelligence
Amrita Vishwa Vidyapeetham
Coimbatore, India
cb.ai.u4aid23118@cb.students.amrita.edu

Darika N. S
Amrita School of Artificial Intelligence
Amrita Vishwa Vidyapeetham
Coimbatore, India
cb.ai.u4aid23110@cb.students.amrita.edu

Jeiesh J. S
Amrita School of Artificial Intelligence
Amrita Vishwa Vidyapeetham
Coimbatore, India
cb.ai.u4aid23116@cb.students.amrita.edu

Kaveen Kawsal S. S
Amrita School of Artificial Intelligence Amrita
Vishwa Vidyapeetham
Coimbatore, India
cb.ai.u4aid23119@cb.students.amrita.edu

Akhil V. M
Amrita School of Artificial Intelligence
Amrita Vishwa Vidyapeetham
Coimbatore, India
vm akhil@cb.amrita.edu

2025 International Conference on Computing for Sustainability and Intelligent Future (COMP-SIF) | 979-8-3315-3853-8/25/$31.00 ©2025 IEEE | DOI: 10.1109/COMP-SIF65618.2025.10969952

***Abstract*—Swarm robotics is an approach inspired by natural systems, where tasks are efficiently accomplished by distributing work among multiple agents. One of the key tasks addressed in this work is area exploration. By leveraging decentralized coordination principles and a reinforcement learning technique called Q-learning, this study focuses on overcoming challenges associated with area exploration. A swarm robotics simulation is developed to map unknown environments using Simultaneous Localization and Mapping (SLAM). The TurtleBot3 Burger is initialized in a simulated environment using ROS Noetic and Gazebo. The trained robots aim to collaboratively map unknown environments while efficiently a voiding o bstacles. A c ustom Q-learning algorithm is implemented for obstacle avoidance tasks. By employing GMapping, a SLAM algorithm, the robots generate individual 2D maps of their surroundings and share this information without conflicts. T hese m aps a re t hen m erged as accurately as possible to create a comprehensive map of the environment. Real-time mapping by the robots is visualized in Rviz, demonstrating the potential of swarm systems. This work effectively provides a collaborative map of the environment, contributing to advancements in swarm intelligence for cooperative robotic systems.**

***Index Terms*—Swarm robotics, Q-learning, TurtleBot3, SLAM, ROS Noetic, collaborative mapping.**

I. Introduction

The evolution of robotics, particularly in the context of multi-agent systems (MAS), has been profoundly shaped by advancements in machine learning (ML) and artificial intelligence (AI). Efficient communication algorithms and coordination mechanisms are essential for enabling agents to learn optimal behaviors in dynamic and changing environments [1].

Machine learning algorithms, such as reinforcement learning and game theory, have been implemented to enable agents to learn complex behaviors and perform tasks beyond human capabilities. These techniques are particularly valuable in military applications, rescue operations, and other critical fields [2].

Swarm robotics, inspired by collaborative behaviors in nature, such as ants working together to locate and transport food to their colonies, offers a decentralized approach in which multiple agents collaborate to achieve common goals. This method is especially effective for managing large swarms of agents [3], [4].

Simulation tools are indispensable for studying swarm robotics. Unity, with its sophisticated physics engine, has emerged as a leading platform for simulating swarm systems, outperforming alternatives such as Webots and V-REP. It enables realistic sensor modeling and debugging, making it particularly suitable for testing algorithms such as localization and Q-learning [5], [6].

Real-world experimentation is further enabled by physical platforms such as Colias, which bridge the gap between simulation and practical testing. Colias, a low-cost microrobot equipped with advanced sensing and communication capabilities, serves as a valuable tool to study decentralized swarm behaviors [7].

Decentralized algorithms, inspired by natural systems, have been a cornerstone of swarm robotics research. Techniques such as genetic algorithms and ant colony optimization have demonstrated significant improvements in evolving coordinated behaviors. These methods are particularly effective for solving complex tasks such as task allocation and optimal pathfinding [8].

Mathematical models, including state-discrete and parameter-continuous Markov chains, have also been applied to analyze and optimize swarm dynamics. These models are particularly useful in cooperative tasks such as box-pushing, which highlight the scalability and effectiveness of decentralized control [9].

The integration of AI and the Internet of Things (IoT) is expanding the possibilities of swarm robotics. IoT enhances communication among robots, improving decision-making and adaptability. Meanwhile, AI-based techniques refine collective behaviors, including obstacle avoidance, path planning, and swarm formation maintenance [10], [11].

Decentralized control mechanisms, as demonstrated by UAV swarms, showcase the potential for real-time adaptive behaviors and effective navigation in unpredictable environments [12], [13]. These advancements are critical for applications in environmental monitoring, search and rescue, and autonomous underwater exploration, where coordination and reliability are paramount [14].

This paper introduces a Q-learning-based algorithm for obstacle avoidance in swarm robots. By leveraging Simultaneous Localization and Mapping (SLAM), each robot in the swarm constructs a map of its surroundings and shares this information with others. This process enhances spatial awareness and coverage. These techniques address challenges in exploring unknown terrains, improving the robustness and scalability of robotic swarms [15], [16].

Despite notable advancements, several challenges remain. Ethical concerns regarding the use of AI in autonomous systems such as transparency and robustness must be addressed. This is particularly important in military contexts, where misuse could lead to severe consequences [17], [18].

Further research is necessary to investigate the scalability and efficiency of swarm systems, especially under energy-constrained conditions. Interdisciplinary approaches that combine AI, IoT, and swarm intelligence are essential for unlocking the full potential of swarm robotics [19], [20].

The integration of AI, machine learning, and simulation methods is paving the way for a new era of swarm robotics. By addressing challenges related to communication, coordination, and adaptability, this research aims to develop autonomous systems capable of tackling real-world problems. In particular, this study focuses on improving exploration and mapping in hard-to-reach areas. The findings contribute to the advancement of swarm robotics, demonstrating its transformative potential in domains such as healthcare, agriculture, environmental monitoring, and disaster response.

II. Methodology

A. Setup Installation

This section describes the operating system setup used for simulating and visualizing the project and its results.

1) ROS (Robot Operating System): ROS is an open-source software that helps build robot applications. In this work, ROS is implemented to manage multiple robots, with each TurtleBot acting as an individual node. There is always a master node that communicates with other nodes by sharing data among them using publisher and subscriber nodes. ROS facilitates autonomous exploration in an environment where a swarm of Turtlebots is implemented.

2) Gazebo: Gazebo is a simulation platform that creates a virtual environment for deploying robots. The world description is given in the SDF file format, while the robot's description is provided in the URDF file format, containing information about sensors, joints, etc. Gazebo integrates the environment and multiple robots for performance analysis in area exploration using the developed custom Q-learning process.

3) Rviz: RViz (ROS Visualization) is an open-source real-time visualization tool, commonly used with Gazebo. It helps track the explored grid, view the Q-value intensity map, and monitor the robot's path using sensor data, the robot model, and its state, updated every second. The swarm's performance can be visually observed using this application.

B. Environment Setup

The TurtleBot3 Burger is a compact and lightweight robot with dimensions of 138 mm × 178 mm × 192 mm and a weight of 1.5 kg. This differential-drive robot has a wheel radius of 0.0975 meters and a wheelbase of 0.160 meters.

Fig. 1. TurtleBot3 Burger.

It is equipped with two sensors:

- **LiDAR Sensor:** Identifies collisions and enables movement by processing 360-degree data to detect obstacles.
- **IMU Sensor:** Provides coordinates and orientation of the robot at each instant.

These data are processed, and the robot's position is tracked by logging it in the Q-table. The devised Q-learning algorithm enables the robot to learn optimal navigation strategies through a reward-based system. Multiple robots coordinate by sharing a common Q-table.

C. SLAM Algorithm Implementation

Implementing SLAM (Simultaneous Localization and Mapping) along with Q-learning enables each robot to autonomously explore, map its environment, and localize itself. SLAM allows swarm robots to simultaneously map and navigate the environment using LiDAR sensor data. ROS packages such as *gmapping* and *move_base* are utilized for path planning and obstacle avoidance.

D. Q-Learning Integration for Enhanced Navigation

The project involves using robotic swarms to explore an unknown environment and obtain its SLAM using G-mapping. While obtaining SLAM is straightforward, controlling each swarm individually is inefficient. Hence, reinforcement learning is implemented to ensure efficient navigation and collision avoidance.

Q-learning is a model-free reinforcement learning algorithm that allows an agent to learn optimal policies through interactions with its environment. It involves an agent, a set of states (S), and a set of actions (A). The agent transitions between states by performing actions and receives numerical rewards or penalties. The objective is to maximize total rewards.

In this work, Q-learning enables TurtleBot3 robots to navigate unknown environments efficiently while minimizing collisions and maximizing exploration. The Q-table stores state-action pairs with rewards and penalties. The TurtleBot3 robots update the Q-table based on received rewards, improving decision-making capabilities. The goal is to maximize overall rewards, ensuring efficient exploration and successful obstacle avoidance.

1) State Representation: The TurtleBot's state space is divided into a grid representation for its x- and y-coordinates, with each grid cell corresponding to a 0.25m × 0.25m area.

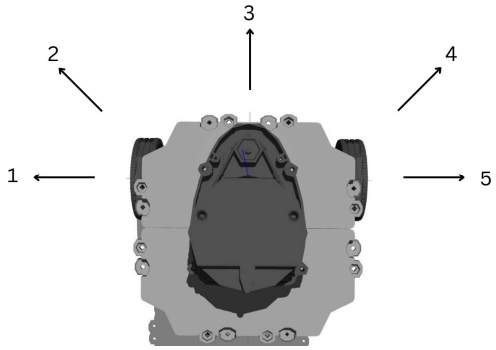

Fig. 2. State representation.

The robot's orientation is categorized into five directions: North (3), Northeast (4), East (5), West (1), and Northwest (2), as illustrated in Fig. 2. This discretization reduces state space complexity while maintaining navigation precision.

2) Action Space: As shown in Table 1, these angular velocity commands define the robot's movement in five discrete directions.

TABLE I
ACTION SPACE REPRESENTATION

Action	Angular Velocity (rad/s)	Description of Movement
1	-1.57	Sudden right turn
2	-0.785	Slight right turn
3	0	Move straight
4	0.785	Slight left turn
5	1.57	Sudden left turn

3) Reward Structure: The reward function in the system ensures that the robot optimizes its behavior by promoting exploration, avoiding collisions, and minimizing redundant movements. As shown in Fig. 2, the exploration reward encourages the agent to explore the environment effectively. A positive reward is assigned based on the proportion of unique grid cells visited.

To ensure safety, a significant penalty is applied when the robot detects a risk of collision. This occurs if the minimum distance to an obstacle, measured by the LiDAR, is less than 0.15 meters. This high penalty strongly deters the agent from entering unsafe states, ensuring that collision avoidance becomes a priority in its policy.

```python
def calculate_reward(self, robot_name):
    robot_state = self.robot_states[robot_name]

    # Calculate exploration-based reward
    exploration_ratio = len(robot_state['explored_grids']) / 324 # 18x18 grid
    exploration_reward = 1000.0 * exploration_ratio

    # Collision penalty
    collision_penalty = -200 if robot_state['min_distance'] < 0.15 else 0

    # Calculate revisit penalty
    current_state = self.discretize_state(robot_name)
    visit_count = robot_state['state_visits'].get(current_state, 0)
    revisit_penalty = 0

    if visit_count > 1:
        revisit_penalty = -5.0 * (visit_count - 1)

    total_reward = exploration_reward + collision_penalty + revisit_penalty

    rospy.loginfo(f"{robot_name} Reward breakdown - "
                  f"Exploration: {exploration_reward:.2f}, "
                  f"Collision: {collision_penalty}, "
                  f"Revisit: {revisit_penalty:.2f}")
    return total_reward
```

Fig. 3. Code snippet of the reward structure algorithm.

To promote efficient exploration and reduce repetitive visits to the same grid cell, a revisit penalty is applied. The penalty increases linearly with the frequency of visits to a specific grid cell. The combined reward function gives the total reward at each step as the sum of the individual components.

As shown in Fig. 3, a code snippet from our implementation illustrates how the reward function is structured to compute exploration rewards, collision penalties, and revisit penalties within the system.

E. Hyperparameters

Table 2 presents the hyperparameter configurations used in our approach, detailing the state representation, learning parameters, and reward structure.

TABLE II
HYPERPARAMETER CONFIGURATION

Hyperparameter	Value	Description
State space	18×18 grid	Represents the environment as a grid. Each cell corresponds to a unique state.
Action space	5	Actions, e.g., move forward, turn at different angles.
Learning rate	0.5	Balances learning speed and stability.
Discount factor	0.95	Gives importance to long-term rewards.
Epsilon	0.5	Starts with a 50% chance of exploration.
Epsilon decay	0.995	Gradually reduces exploration probability per episode.
Epsilon min	0.01	Ensures some level of exploration persists.
Stopping criteria	Full exploration	Training stops when all 18×18 = 324 grid cells are visited.
Collision threshold	≤0.15 meters	Minimum safe distance before applying a penalty.
Exploration reward	1000× exploration ratio	Encourages exploration of unvisited cells.
Collision penalty	-200	Discourages collisions with obstacles.
Revisit penalty	-5× (visit count-1)	Penalizes repeated visits to explored areas.

F. Flow Analysis

The flow of publisher and subscriber nodes is illustrated in Fig. 4, where the multi_robot_qlearning node inter- acts with multiple robots by subscribing to their sensor data and publishing velocity commands.

In a multi-robot system, publisher and subscriber nodes play a crucial role in enabling communication and coordination

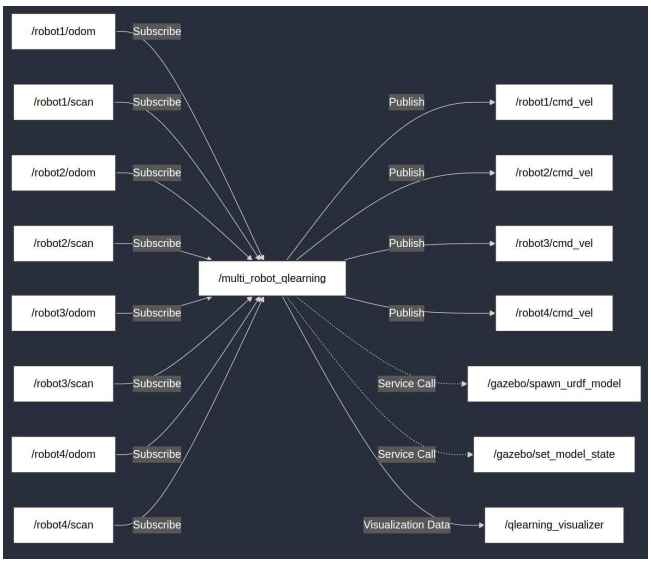

Fig. 4. Flow analysis of publisher and subscriber nodes.

	grid_x	grid_y	orientatio	action_1	action_2	action_3	action_4	action_5
2	0	0	N	-65.397	-40.04	-54.3456	-60.9487	-97.4177
3	1	0	N	-95.9491	-28.9488	-69.1288	-34.6775	-58.0731
4	1	0	NW	-9.85	-9.9	-21.1	-2.35	0
5	2	-1	NW	-9.775	-21.025	-2.35	0	-9.85
6	2	-1	W	-39.3806	-43.5804	-75.8452	-56.5994	-40.6681
7	2	-2	W	-7.2	-7.3	-4.8	0	0
8	2	-2	SW	-9.7	-17.2	-2.25	0	0
9	1	-2	SW	-34.3188	-77.1176	-75.0719	-50.6272	-41.6344
10	1	-3	SW	-32.67	-7.225	-55.2163	-20.8	-27.6413
11	0	-3	SW	-4.475	-13.225	0	0	0
12	0	-3	S	-38.8881	-89.7353	-16.8375	-76.1572	-62.2009
13	-1	-3	S	-14.3	-2.1	0	-1.25	-22.3
14	-1	-3	SE	-23	-46.4375	-22.1	-2.05	-9.6
15	-1	-2	SE	-13.075	-4.55	-2.05	0	0
16	-1	-2	E	-25.575	-27.9625	-24.325	-4.325	-24.55
17	-1	-2	NE	-4.55	-2	-7.05	0	0
18	-1	-1	NE	-16.625	-9.5	-9.25	0	0
19	-1	-1	N	-6.575	-24.25	-31.75	-24.5	-39.7688
20	0	-1	N	-64.5747	-91.6208	-26.1881	-14.5438	-45.9762
21	0	-1	NE	-12.925	-4.45	-52.4366	0	0
22	1	-1	N	-9.1	-22.7	-1.9	0	0
23	1	-1	NW	-18.95	-16.6	-35.825	-39.1	-1.9
24	1	-2	NW	-20.35	-1.9	-9.4	-6.9	-4.4
25	1	-2	W	-9.1	-38.25	-24.1	-49.5419	-1.9
26	1	-3	W	-51.1163	-45.4113	-21.6325	-50.6438	-16.9

Fig. 6. Q-table for single TurtleBot3.

among swarm robots. Within the ROS framework, each robot operates as an independent node, publishing relevant data such as sensor readings and Q-values. At the same time, robots subscribe to specific topics to receive updates and share knowledge, ensuring synchronized decision-making and efficient navigation.

III. Result and Analysis

A single TurtleBot3 robot has been successfully deployed in Gazebo, as seen in Fig. 5, and the area exploration task has begun. This is to test the robot and the world integration in Gazebo. If the robot faces an obstacle, it resets its position to the initial state and begins exploration with previous knowledge.

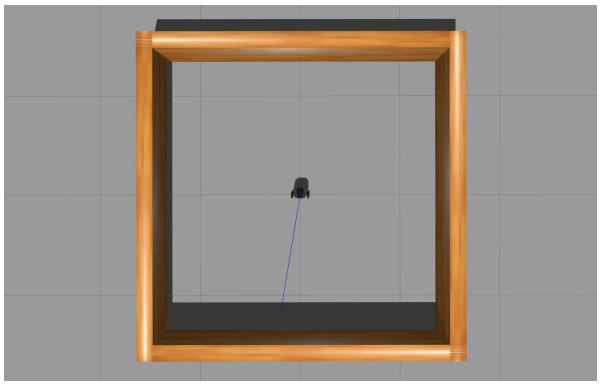

Fig. 5. Single TurtleBot3 deployed in Gazebo.

Fig. 6 illustrates the Q-table in which the Q-values collected by the single TurtleBot3 robot are being saved. The values are stored in an Excel file, and later, these values are used by the Q-learning algorithm to determine or influence the future action of the robot based on the reward and penalty points for that particular state and action. Also, the area is divided into grid cells of size 0.25 units each for simplification.

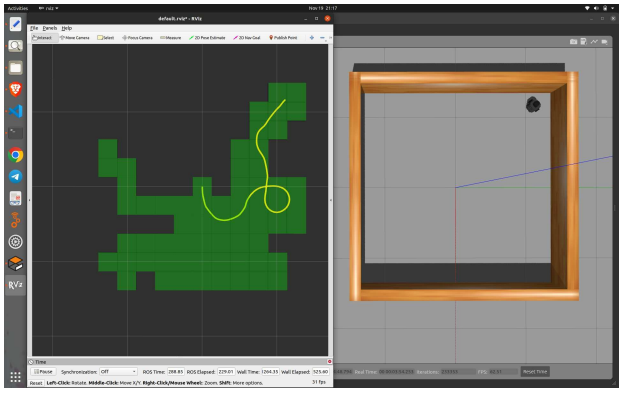

Fig. 7. Area exploration for a single TurtleBot3 using SLAM.

The graph seen in Fig. 7 shows the area exploration map visualized in RViz by the single TurtleBot3 robot. The green grids represent the area explored by the robot so far, and the yellow line shows the path trajectory of the robot in real-time. The yellow line moves in accordance with the robot's movement.

The color-coded grid map shown in Fig. 8 is the Q-value intensity for a single TurtleBot3, based on the logged values from the CSV file. Here, red (high Q-values), green (medium Q-values), and blue (low Q-values) indicate the intensity of Q-value distribution in the explored areas.

After training a single robot, the experience gained helped to deploy four TurtleBot3 Burger robots in Gazebo, as shown in Fig. 9. These robots explored an 18x18 grid (with each cell measuring 0.25 m x 0.25 m) with obstacles, which was also trained using a custom Q-learning algorithm. Here, all four robots are deployed in different states or positions in

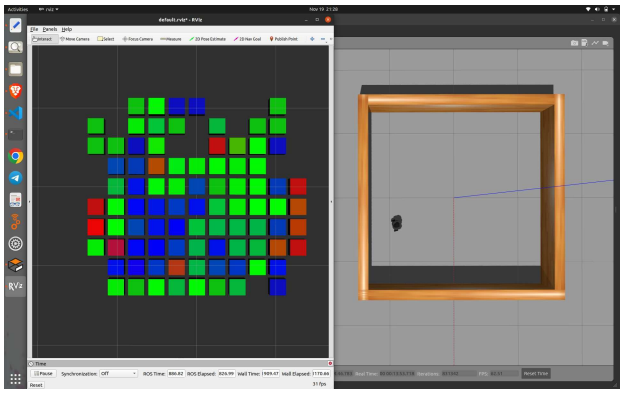

Fig. 8. Q-value intensity map for a single TurtleBot3.

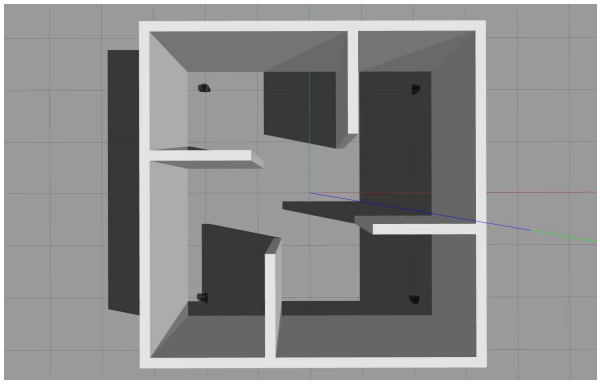

Fig. 9. Multi-robots deployed in Gazebo.

the environment, and they start exploring the unknown area while simultaneously updating the Q-table with Q-values and creating their own map. If any of the robots collide with an obstacle, they are programmed to restart from the original or initial state or position.

	robot	grid_x	grid_y	orientatio	action_1	action_2	action_3	action_4	action_5
2	turtlebot3_1	8	8	N	-14367.3	-5840.6	-32266.9	-4488.14	-13885.4
3	turtlebot3_2	-8	-8	N	-20918.5	-33360.7	-34135.6	-33185.3	-33352.7
4	turtlebot3_3	8	-8	W	-6776.02	-6628.18	-4046.31	-6613.45	-6929.64
5	turtlebot3_4	-8	8	N	-1149.15	-2853.33	-1166.12	-454.47	-6718.95
6	turtlebot3_2	-8	-8	E	-894.949	-719.967	-880.067	-998.333	-905.584
7	turtlebot3_2	-8	-7	E	-725.872	-488.776	-514.13	-638.023	-565.068
8	turtlebot3_3	8	-9	W	-1129.56	-1397.91	-1243.73	-1299.75	-1509.33
9	turtlebot3_4	-7	8	N	-667.944	-691.071	-743.38	-724.448	-463.486
10	turtlebot3_1	9	8	N	-914.178	-951.819	-1261.13	-974.487	-958.578
11	turtlebot3_2	-8	-6	E	-378.41	-338.939	-408.953	-289.78	-393.488
12	turtlebot3_4	-6	8	N	-261.151	-266.252	-280.574	-238.17	-273.222
13	turtlebot3_2	-8	-6	SE	-2.3	0	0	0	0
14	turtlebot3_4	-5	8	N	-125.856	-103.085	-31.0023	-150.242	-170.074
15	turtlebot3_2	-8	-5	SE	0	0	0	-2.25	0
16	turtlebot3_2	-9	-5	SE	-7.25	-23.375	0	0	0
17	turtlebot3_4	-5	8	NW	-12.3	-57.5963	-10.55	-110.219	-5.9
18	turtlebot3_4	-4	8	NW	-2.2	0	0	0	0
19	turtlebot3_4	-4	7	NW	-114.446	-106.584	-46.8264	-117.267	-134.794
20	turtlebot3_4	-3	7	NW	-7.15	-5.5	82.86975	-19.3475	-41.7782
21	turtlebot3_4	-3	6	NW	-28.3975	-34.0422	-47.1725	1785.462	-27.1
22	turtlebot3_4	-2	5	NW	169.4195	-46.5972	-17.05	1701.925	-38.075
23	turtlebot3_2	-8	-5	E	-86.5912	-246.529	-197.368	-221.978	-530.58
24	turtlebot3_4	-2	4	NW	156.8014	-38.5	1929.653	-44.4356	-23.43
25	turtlebot3_2	-8	-4	E	-43.3975	-40.9225	-77.4313	-52.9313	145.1829
26	turtlebot3_4	-1	4	NW	1868.236	-27.9025	-22.7075	467.1691	-16.05

Fig. 10. Q-table for multi-robots.

For swarm TurtleBots, a shared global Q-table (as illustrated in Fig. 10) is used, allowing each robot to make decisions based on the best Q-value from a state previously explored by another robot. This enables collaborative learning and efficient exploration, which is the primary goal of working as a swarm.

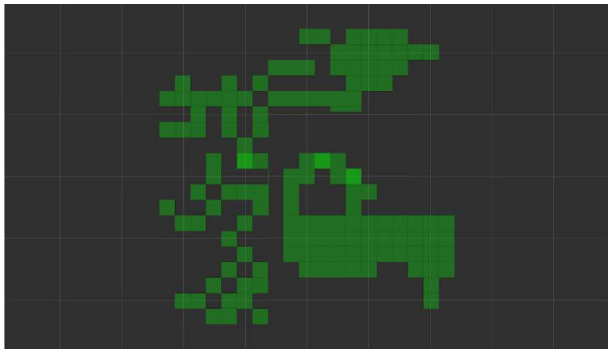

Fig. 11. Area exploration map for multi-robots.

The merged map of the grid, Fig. 11 shows the areas explored by the four TurtleBot3 Burgers working as a swarm to explore the 18x18 grid using our custom Q-learning algorithm.

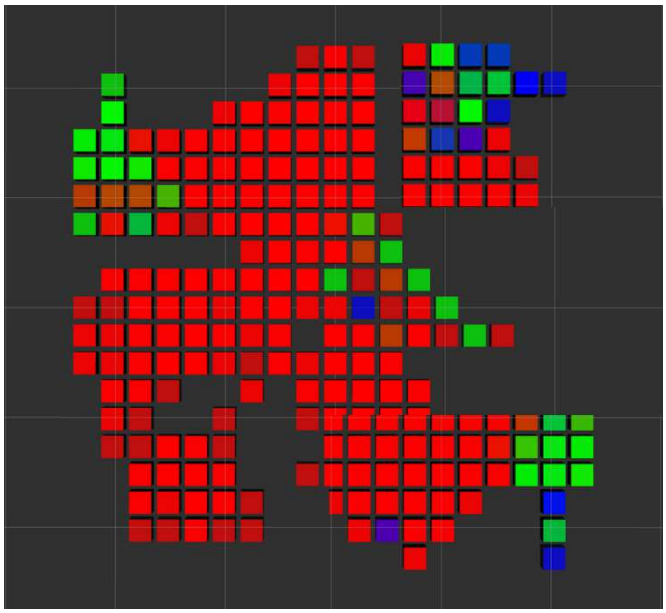

Fig. 12. Q-value intensity map for multi-robots.

Plotted in RViz, Fig. 12 shows the Q-value intensity map based on the shared Q-table data. The robots were able to autonomously navigate the unknown area using shared knowledge with Q-learning.

IV. CONCLUSION

This work follows an exploration and mapping approach for swarm robotics in unknown terrains implementing Q-learning and SLAM focusing on optimizing coverage through enhanced obstacle detection and navigation. The project successfully integrated multi-robot coordination, Q-learning, and SLAM which leads to cohesive simultaneous mapping and localization. The simulation results using Gazebo, ROS, and

Rviz illustrates that Q-learning significantly enhances obstacle avoidance and decision making through Q table and Q values in dynamic scenarios. SLAM ensures accurate localization and mapping, necessary for coordinated swarm behavior. The results demonstrate that the collective behaviour of the swarm enhances coverage and efficiency in unknown environments, with each robot contributing to the overall mapping process. The work recommends future research should explore the scalability of swarm robotics for larger, more complex environments and further develop algorithms to enhance multi-agent coordination, hazard detection, and resource allocation. Additionally, improvements in communication advanced sensors and robotic platforms, could increase the system's adaptability and robustness.

REFERENCES

[1] C. Alexandris, P. Papageorgas, and D. Piromalis, "Positioning systems for unmanned underwater vehicles: A comprehensive review," *Appl. Sci.*, vol. 14, no. 21, p. 9671, 2024.

[2] G. Jain, A. Kumar, and S. A. Bhat, "Recent developments of game theory and reinforcement learning approaches: A systematic review," *IEEE Access*, vol. 12, pp. 12345-12380, 2024.

[3] E. Sahin and A. F. T. Winfield, "Special issue on swarm robotics," *Swarm Intell.*, vol. 2, no. 2-4, pp. 69–72, 2008.

[4] R. Arnold et al., "What is a robot swarm: A definition for swarming robotics," in *Proc. 10th Annu. Ubiquitous Comput., Electron. Mobile Commun. Conf. (UEMCON)*, New York, NY, USA, 2019, pp. 0074-0081.

[5] M. S. P. De Melo et al., "Analysis and comparison of robotics 3D simulators," in *Proc. 21st Symp. Virtual Augmented Reality (SVR)*, Rio de Janeiro, Brazil, 2019, pp. 242-251.

[6] B.-S. Le, V.-L. Dang, and T.-T. Bui, "Swarm robotics simulation using Unity," Faculty Electron. Telecommun., Univ. Sci., VNU-HCM, Ho Chi Minh City, Vietnam, Tech. Rep., 2014.

[7] F. Arvin et al., "Colias: An autonomous micro robot for swarm robotic applications," *Int. J. Adv. Robot. Syst.*, vol. 11, no. 7, p. 113, 2014.

[8] N. R. Augad and V. S. Gutte, "Swarm intelligence for AI problem solving," in *Proc. Int. Conf. Adv. Comput. Commun. Technol. (ICACCTech)*, Greater Noida, India, 2023, pp. 590-596.

[9] Y. Li and X. Chen, "Modeling and simulation of a swarm of robots for box-pushing task," in *Proc. 12th Mediterr. Conf. Control Autom.*, Kusadasi, Turkey, 2004, pp. 1-6.

[10] R. Sawant et al., "A bibliometric perspective survey of IoT controlled AI based swarm robots," *Libr. Philos. Pract.*, vol. 2021, pp. 1–31, 2021.

[11] A. A. Alahmadi et al., "AI driven approaches in swarm robotics - A review," *Int. J. Comput. Inform.*, pp. 100–133, May 2024.

[12] J. Kusyk et al., "AI based flight control for autonomous UAV swarms," in *Proc. Int. Conf. Comput. Sci. Comput. Intell. (CSCI)*, Las Vegas, NV, USA, 2018, pp. 1155-1160.

[13] W. Cai et al., "Cooperative artificial intelligence for underwater robotic swarm," *Robot. Auton. Syst.*, vol. 164, p. 104410, 2023.

[14] M. Rajesh, S. R. Nagaraja, and P. Suja, "Multi-robot exploration supported by enhanced localization with reduction of localization error using particle swarm optimization," *J. Wireless Mobile Netw. Ubiquitous Comput. Dependable Appl.*, vol. 15, no. 1, pp. 202-215, 2024.

[15] C. S. Chowdary et al., "A comparative analysis of swarm algorithms for enhancing communication in drone networks," in *Proc. 15th Int. Conf. Comput. Commun. Netw. Technol. (ICCCNT)*, Kharagpur, India, 2024, pp. 1-8.

[16] C. Xiong et al., "Onboard cooperative relative positioning system for Micro-UAV swarm based on UWB/Vision/INS fusion through distributed graph optimization," *Meas. J. Int. Meas. Confed.*, vol. 234, p. 114897, 2024.

[17] A. Ilachinski, *AI, Robots, and Swarms: Issues, Questions, and Recommended Studies*. Alexandria, VA, USA: CNA Corp., 2017.

[18] S. Javed et al., "State-of-the-art and future research challenges in UAV swarms," *IEEE Internet Things J.*, early access, 2024, doi: 10.1109/JIOT.2024.123456.

[19] I. Kishor and K. Sharma, "A comprehensive review of robotics: Advances, challenges and future prospects," *J. Adv. Robot. Syst.*, vol. 15, no. 3, pp. 1-25, 2023.

[20] A. Hafid et al., "Centralized and decentralized federated learning in autonomous swarm robots: Approaches, algorithms, optimization criteria and challenges," in *Proc. 6th Int. Conf. Pattern Anal. Intell. Syst. (PAIS)*, Oujda, Morocco, 2024, pp. 1-8.