Mobile robot navigation: A study on various DRL and traditional path planning algorithms

Penaka Vishnu Reddy, Menta Sai Aashish, Gorantla V.N.S.L Vishnu Vardhan, Chandu Janakiram, Palmani Duraisamy
Amrita School of Artificial Intelligence,
Amrita Vishwa Vidhyapeetham, Coimbatore, India.
d_palmani@cb.amrita.edu

***Abstract*—In navigating dynamic environments, traditional methods like Dijkstra's and A* face challenges due to their reliance on explicit maps. This research explores the shift towards map-less navigation using Deep Reinforcement Learning (DRL) algorithms (Q-learning, DQN, TD3, PPO). Unlike traditional approaches, DRL allows robots to learn and adapt autonomously, proving effective in dynamic scenarios. The study compares the strengths and limitations of both traditional and DRL methods, contributing insights for developing more robust and adaptive robotic navigation systems.**

***Index Terms*—Mobile robots, path planning, reinforcement learning, simulation.**

I. Introduction

This section provides a comprehensive overview of key components in the intersection of robotics and reinforcement learning, laying the foundation for understanding the advancements and challenges in the field.

A. Robot Operating System (ROS)

ROS, an open-source software framework, is pivotal in robotics research, providing a modular environment for seamless integration of components, facilitating prototyping, testing, and deployment.

B. Rviz

Rviz, part of ROS, is a vital visualization tool for debugging and fine-tuning robotic algorithms, offering real-time insights into sensor data and robot movements.

C. Gazebo

Gazebo, a robust simulation software in ROS, enables realistic and dynamic simulations, accelerating algorithm development by providing a controlled environment for testing and validation.

D. OpenAI Gym

OpenAI Gym is a powerful toolkit for reinforcement learning, offering standardized environments for algorithm development and comparison. It fosters collaboration and accelerates progress in autonomous systems.

E. Concepts Required for Movement of Robot

This section delves into fundamental concepts crucial for understanding and implementing robot movement.

1) Frames: In the context of robotics, frames establish a coordinate system that serves as the foundation for navigation. These frames define the robot's position and orientation within its environment, forming the basis for accurate movement and spatial awareness.

2) Transformation: Transformation involves manipulating the position and orientation between frames. It allows for seamless transitions between different coordinate systems, enabling the robot to interpret and navigate diverse environments accurately.

3) Rotation Matrices: Rotation matrices are pivotal in three-dimensional transformations. They provide a mathematical representation of how the robot's orientation changes, allowing for precise control over its movements in a 3D space. General rotation matrix in 3D:

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\Theta) & -sin(\Theta) \\ 0 & sin(\Theta) & cos(\Theta) \end{bmatrix} \tag{1}$$

$$R_y = \begin{bmatrix} cos(\Theta) & 0 & sin(\Theta) \\ 0 & 1 & 0 \\ -sin(\Theta) & 0 & cos(\Theta) \end{bmatrix} \tag{2}$$

$$R_z = \begin{bmatrix} cos(\Theta) & -sin(\Theta) & 0 \\ sin(\Theta) & cos(\Theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$$R = R_z(\alpha)R_y(\beta)R_x(\gamma) \tag{4}$$

General transformation matrix in 3D:

$$\begin{bmatrix} w1_x \\ w1_y \\ w1_z \\ 1 \end{bmatrix} = \begin{bmatrix} w1_{R_{w2}} & t \\ 0_{1x3} & 1 \end{bmatrix} * \begin{bmatrix} w2_x \\ w2_y \\ w2_z \\ 1 \end{bmatrix} \tag{5}$$

F. Reinforcement Learning

Reinforcement learning is a paradigm focused on discerning optimal actions in dynamic environments through a process of trial-and-error learning [1].

1) RL Algorithm: At the core of reinforcement learning is the RL algorithm, which orchestrates the interaction between the agent and its environment. Through iterative trial-and-error, the algorithm refines the agent's decision-making processes [2].

2) Elements of RL problem: Key components in a reinforcement learning problem include the Agent, Model, Policy, Reward Signal, Value Function, and State-Action Value Function [3]. These elements collectively define the learning environment and guide the agent towards optimal decision-making.

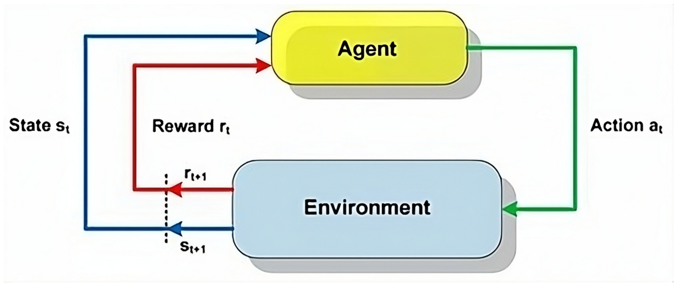


Fig. 1. RL Paradigm

3) Markov Decision Process (MDP): MDP provides a mathematical framework for solving reinforcement learning problems. It formalizes the sequential decision-making process, ensuring that the current state captures all relevant information for future actions.

4) Model-based Reinforcement Learning: Model-based reinforcement learning extends traditional Q-learning by incorporating online models [4]. These models enable the agent to simulate and predict the outcomes of different actions, facilitating more informed decision-making [5].

5) Model-free Reinforcement Learning: In contrast, model-free reinforcement learning algorithms interact with the environment directly, relying on trial and error to optimize decision-making processes. These algorithms, such as Q-learning, excel in situations where an explicit model of the environment is unavailable [6].

II. Objectives

A. Project Objective

The primary goal of "Mobile Robot Navigation: A Study on Various DRL and Traditional Path Planning Algorithms" is twofold. Firstly, the project aims to explore and compare navigation strategies by mapping a house environment using Gmap and SLAM techniques. Navigation utilizes ROS Gazebo and traditional path planning algorithms (Dijkstra's and A*). Secondly, the project extends to reinforcement learning, implementing Q-learning, DQN, TD3, and PPO in a mountain car gym environment. The objective is to assess the performance of traditional and DRL algorithms in diverse settings.

B. Key Project Tasks

1) Mapping and Navigation in House Environment [7].
2) Reinforcement Learning in Mountain Car Gym Environment.

III. Methodology

A. Navigation

The navigation phase involved a dual approach, focusing on map-based and reactive navigation strategies. This section offers an in-depth exploration of the map-based navigation strategy.

1) Map-based Navigation: Map-based navigation, a pivotal component of the project, revolves around three fundamental elements: Localization, Mapping, and Motion Planning.

a) Localization:: Accurate determination of the robot's position was achieved through the implementation of sophisticated localization techniques. This ensured precise spatial awareness during navigation.

b) Mapping:: The exploration of Simultaneous Localization and Mapping (SLAM) techniques, including Gmapping, Cartographer, and Hector SLAM, played a crucial role in dynamically generating maps as the robot traversed the environment.

c) Motion Planning:: The navigation stack, a key framework, employed move_base for dynamic obstacle-free path planning. The global path planning aspect utilized the navfn package, emphasizing a comprehensive approach to path optimization.

2) Building a Map: SLAM: In the pursuit of constructing detailed maps using range sensors, we delved into various SLAM approaches, such as Gmapping, Cartographer, and Hector SLAM.

a) Occupancy in Grid Map:: The representation of the environment as a grid, with cells indicating occupancy probabilities, involved intricate techniques like ray-tracing. This ensured a nuanced understanding of the surroundings.

b) Usage of Action Services:: To enhance the interactive capabilities of the robot, action services were utilized. These services facilitated feedback mechanisms for robot movement, goal definition, and task completion.

c) Navigation Stack:: At the core of the map-based navigation strategy was the navigation stack, comprising both global and local path planners. The selection of navfn as the global path planner underscored a commitment to efficiency and adaptability in pathfinding.

d) Marking and Clearing:: The map construction process entailed the dynamic marking of cells as obstacles or clearing them based on real-time sensor data, contributing to a dynamic and adaptive mapping system.

Cell has three possible states:

- Unknown
- Empty
- Occupied

It is based on the process of marking and clearing:

- **Marking**: A cell is marked as an obstacle.
- **Clearing**: A cell is marked as empty.
- **ray-tracing**: Used to find empty cell.

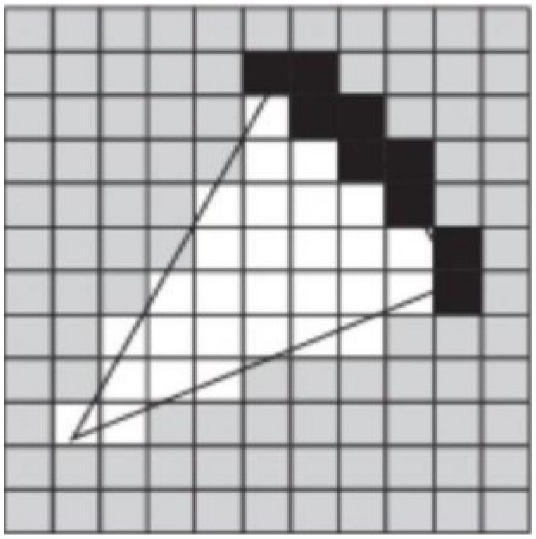


Fig. 2. Grid Mapping

B. Mountain Car Environment

Environment Specifications: The Mountain Car environment had a discrete action space of 3, observation shape (2,), and observation limits [-1.2 -0.07] to [0.6 0.07] [8]. (See Table I)

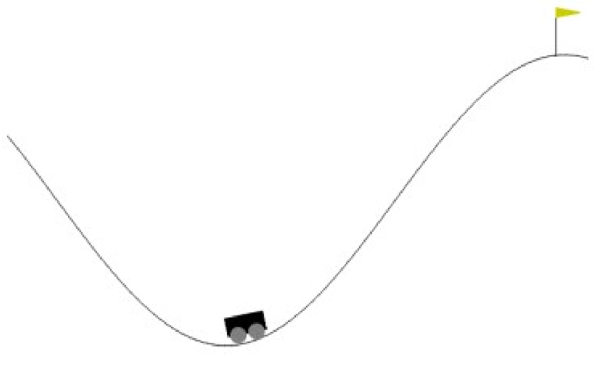


Fig. 3. Mountain Car

TABLE I
SPECIFICATIONS OF THE MOUNTAIN CAR ENVIRONMENT

Action Space	Discrete(3)
Observation Shape	(2,)
Observation High	[0.6 0.07]
Observation Low	[-1.2 -0.07]
Gym Environment	MountainCar-v0

1) Q-learning: Q-learning, a model-free reinforcement learning technique, is employed to allow the agent to learn an optimal policy without prior knowledge of the environment model. The agent constructs a Q-table, mapping state-action pairs to subsequent states and rewards. During training, an ϵ-greedy strategy balances exploration and exploitation, guiding the agent through the state space [9].

$$Q^{\pi}(s,a) = E^{\pi}((\sum_{k=0}^{k=\infty})\gamma^{k} R_{t+1+k}|S_t = s, A_t = t) \quad (6)$$

2) Deep Q Learning (DQN): Deep Q Learning leverages deep neural networks to approximate Q-values, enabling the handling of complex state-action spaces. The agent utilizes experience replay, a process where transitions are stored and randomly sampled during training. This enhances learning efficiency and stability, preventing the network from becoming biased towards recent experiences [10].

$$Y_t^{DQN} = R_{t+1} + \gamma max_a Q(S_{t+1}, a; \theta_t^{-}) \quad (7)$$

3) PPO - Proximal Policy Optimization: PPO strikes a balance between implementation simplicity and training effectiveness. The algorithm computes updates that minimize the cost function while constraining policy deviation. This is achieved by using a clipped surrogate objective function, ensuring a conservative policy update. The approach aims to provide stability in training while requiring minimal hyperparameter tuning.

$$L(\theta) = \epsilon_{\tau}[min(r(\theta)\hat{A}\_t, clip(r(\theta), 1-\epsilon, 1+\epsilon)\hat{A}\_t)] \quad (8)$$

4) TD3 - Twin-Delayed DDPG: TD3 addresses the overestimation bias common in previous algorithms, such as DDPG. It introduces twin critics, delayed updates of the actor, and action noise regularization. By employing two critics and updating the actor less frequently than the critics, TD3 mitigates the overestimation issue, leading to more stable training in continuous control problems.

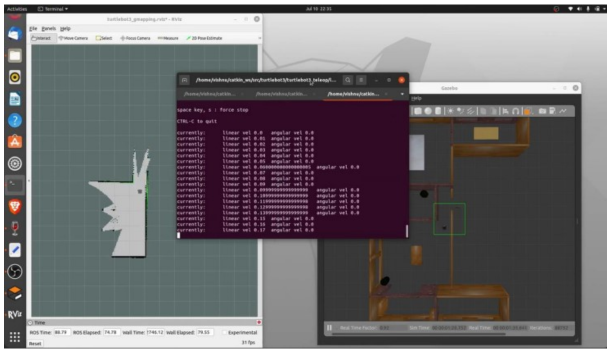
$$Q1(s,a), Q2(s,a) \leftarrow Q1(s,a) + \alpha * (r + \gamma * min(Q1(s',a'), Q2(s',a')) - Q \quad (9)$$

$$\pi\_\theta \leftarrow argmax\_aQ1(s,a) + \eta * N(updatedlessfrequently) \quad (10)$$

C. Simulation

The proposed methodologies undergo rigorous evaluation in a simulated environment facilitated by Gazebo. Gazebo serves as a virtual arena that meticulously replicates real-world scenarios, offering a controlled yet dynamic setting for comprehensive testing of the robot's design and the functionality of the implemented code. Simulations in Gazebo provide a crucial platform to assess the efficacy, reliability, and adaptability of the employed algorithms, ensuring that the robot's performance meets the desired standards under diverse conditions.

The above figures represent the simulation in ROS- Gazebo environment.



```
space key, s : force stop

CTRL-C to quit

currently:      linear vel 0.0   angular vel 0.0
currently:      linear vel 0.01  angular vel 0.0
currently:      linear vel 0.02  angular vel 0.0
currently:      linear vel 0.03  angular vel 0.0
currently:      linear vel 0.04  angular vel 0.0
currently:      linear vel 0.05  angular vel 0.0
currently:      linear vel 0.060000000000000005  angular vel 0.0
currently:      linear vel 0.07  angular vel 0.0
currently:      linear vel 0.08  angular vel 0.0
currently:      linear vel 0.09  angular vel 0.0
currently:      linear vel 0.09999999999999999   angular vel 0.0
currently:      linear vel 0.10999999999999999   angular vel 0.0
currently:      linear vel 0.11999999999999998   angular vel 0.0
currently:      linear vel 0.12999999999999998   angular vel 0.0
currently:      linear vel 0.13999999999999999   angular vel 0.0
currently:      linear vel 0.15  angular vel 0.0
currently:      linear vel 0.16  angular vel 0.0
currently:      linear vel 0.17  angular vel 0.0
```

Fig. 4. World Mapping through Rviz

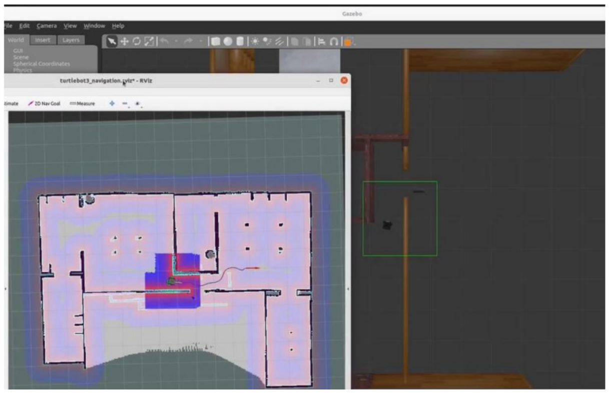
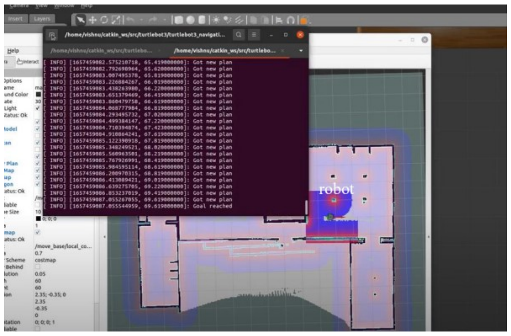


Fig. 5. Robot navigation with respect to the location and obstacles



```
[ INFO] [1657459082.575210718, 65.419000000]: Got new plan
[ INFO] [1657459082.792698964, 65.620000000]: Got new plan
[ INFO] [1657459083.007495378, 65.819000000]: Got new plan
[ INFO] [1657459083.226884267, 66.019000000]: Got new plan
[ INFO] [1657459083.438263980, 66.220000000]: Got new plan
[ INFO] [1657459083.651379469, 66.419000000]: Got new plan
[ INFO] [1657459083.860479758, 66.619000000]: Got new plan
[ INFO] [1657459084.068777984, 66.819000000]: Got new plan
[ INFO] [1657459084.293495732, 67.020000000]: Got new plan
[ INFO] [1657459084.499384147, 67.220000000]: Got new plan
[ INFO] [1657459084.710394874, 67.423000000]: Got new plan
[ INFO] [1657459084.910864241, 67.619000000]: Got new plan
[ INFO] [1657459085.122390918, 67.819000000]: Got new plan
[ INFO] [1657459085.348249521, 68.020000000]: Got new plan
[ INFO] [1657459085.560963501, 68.219000000]: Got new plan
[ INFO] [1657459085.767926991, 68.419000000]: Got new plan
[ INFO] [1657459085.984595114, 68.619000000]: Got new plan
[ INFO] [1657459086.200970315, 68.819000000]: Got new plan
[ INFO] [1657459086.413089421, 69.019000000]: Got new plan
[ INFO] [1657459086.639275705, 69.220000000]: Got new plan
[ INFO] [1657459086.853237019, 69.419000000]: Got new plan
[ INFO] [1657459087.055267055, 69.619000000]: Got new plan
[ INFO] [1657459087.055544959, 69.619000000]: Goal reached
```

Fig. 7. Robot Reached Goal Successfully

IV. Results and Discussion

A. ROS Simulation Results

The simulation results using Robot Operating System (ROS) are presented below, showcasing the achievements of the robotic system.

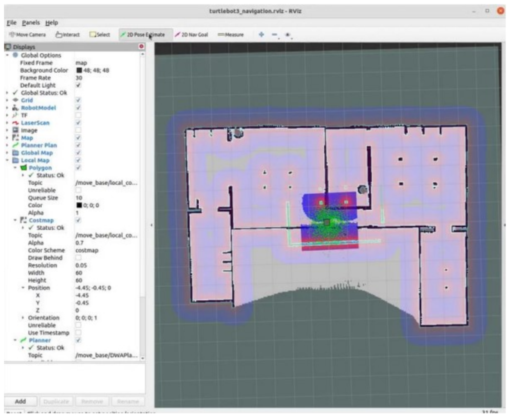


Fig. 6. Robot mapping is complete

Figure 6 illustrates the successful completion of robot mapping, highlighting the system's ability to navigate and map its environment.

Figure 7 demonstrates the robot's successful navigation, reaching its goal within the simulated environment.

B. Q-learning

The Q-learning algorithm demonstrated commendable learning capabilities in the MountainCar-v0 environment. Despite the challenge posed by gravity, the car learned to strategically leverage potential energy by navigating the valley and reaching the goal at the hill's summit. The algorithm solved the environment in 283,600 episodes within 22 minutes.

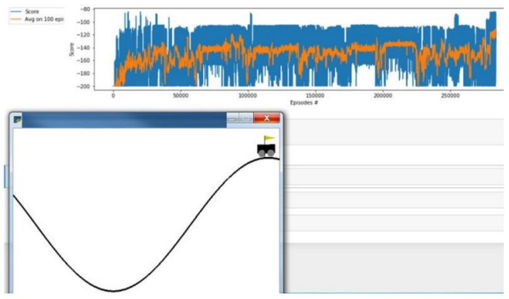


Fig. 8. Q-Learning

C. Deep Q Learning (DQN)

DQN, building upon Q-learning principles with the incorporation of deep neural networks, showcased its effectiveness in the simulated environment. The algorithm successfully solved the MountainCar-v0 environment in 1,356 episodes, taking approximately 1.75 hours.

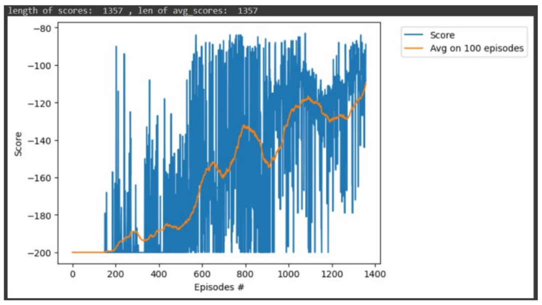
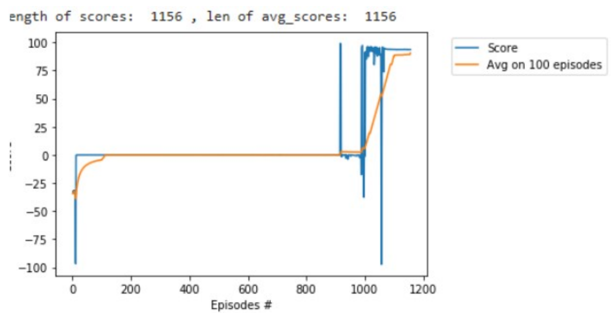


Fig. 9. DQN

D. Twin Delayed DDPG (TD3)

The Twin Delayed DDPG (TD3) algorithm addressed overestimation bias, achieving a remarkable environment-solving capability in 1,156 episodes. The algorithm's deployment spanned 10 hours, emphasizing its robustness in challenging scenarios.

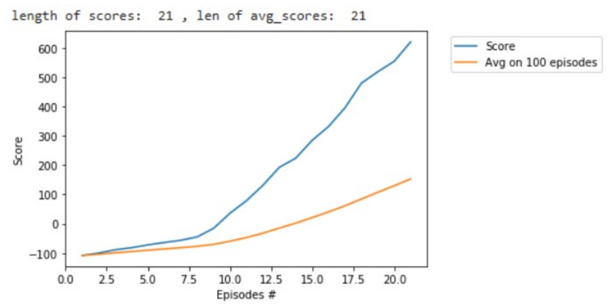


```
ength of scores:  1156 , len of avg_scores:  1156
```

Fig. 10. TD3

E. Proximal Policy Optimization (PPO)

PPO demonstrated exceptional efficiency, solving the environment within 21 episodes or approximately 4 minutes. The use of vectorized environments with 16 processes contributed to this rapid convergence. The average score reached an impressive 152.74, highlighting the algorithm's effectiveness in achieving the defined threshold.



```
length of scores:  21 , len of avg_scores:  21
```

Fig. 11. PPO

V. Future Scope

A. Unified Metric Development

In the realm of robotic navigation, the development of a unified metric stands as a crucial avenue for future research. This metric should transcend the boundaries between map-based and reactive navigation systems, offering a standardized framework for comprehensive evaluation. Parameters to be considered include efficiency, measuring the system's ability to reach goals with minimal resources; adaptability, assessing how well the system responds to dynamic and changing environments; and computational overhead, understanding the computational resources required for optimal performance. Establishing a unified metric will not only facilitate fair comparisons between different navigation approaches but also guide the design and optimization of robotic systems for a wide range of applications.

B. Integration of SLAM with DRL

The integration of Simultaneous Localization and Mapping (SLAM) techniques with Deep Reinforcement Learning (DRL) represents a promising direction for advancing robotic navigation capabilities. Research in this area should delve into the synergy between SLAM, which enhances spatial awareness by mapping the environment, and DRL, which empowers the system with adaptive learning. Investigating how SLAM-derived spatial information can inform DRL decisions and how DRL can contribute to real-time SLAM updates will be crucial. This integration has the potential to create more robust and adaptable robotic navigation systems, particularly in complex and unknown environments where both mapping and learning are imperative.

C. Exploration of Self-Adaptive Critic (SAC) in DRL

The application of Self-Adaptive Critic (SAC) mechanisms in Deep Reinforcement Learning (DRL) for robotic navigation introduces a novel dimension to the field. SAC's dynamic adjustments to the critic network during training open avenues for improving learning efficiency and adaptability. Future research should delve into the nuanced aspects of SAC implementation, exploring how these self-adaptive mechanisms can be fine-tuned for optimal performance in various navigation scenarios. Understanding the impact of SAC on learning convergence, exploration-exploitation trade-offs, and adaptability to different environmental dynamics will be pivotal for harnessing the full potential of this approach.

D. Advanced Learning Techniques

The future of robotic navigation lies in the integration of advanced learning techniques to augment adaptability and intelligence. Building upon the foundation laid by approaches like Self-Adaptive Critic (SAC), the focus should be on incorporating a spectrum of cutting-edge learning techniques. This may include advanced reinforcement learning algorithms, neural network architectures, and ensemble methods. The objective is to elevate the adaptability of robotic systems, enabling them to learn more efficiently and perform optimally in diverse and dynamic environments. As technology evolves, exploring and incorporating these advanced learning techniques will be instrumental in pushing the boundaries of what robotic navigation systems can achieve.

VI. Conclusion

This project extensively explored mobile robotics, specifically focusing on the TurtleBot 3 in ROS Gazebo. The integration of Simultaneous Localization and Mapping (SLAM) techniques, including Dijkstra's algorithm, A*, and the Dynamic Window Approach (DWA) planner, allowed the TurtleBot 3 to navigate autonomously within a simulated environment.

The mapping and navigation phases showcased the strengths and trade-offs between map-based and reactive navigation strategies. Map-based navigation, utilizing algorithms like Dijkstra's and A*, demonstrated excellent performance in environments where the entire layout is known beforehand, such as structured warehouses.

On the other hand, reactive navigation, implemented through the DWA planner, proved advantageous in dynamic and changing environments, allowing the TurtleBot 3 to adapt quickly to obstacles or unforeseen changes.

In the reinforcement learning domain, various algorithms, including Q-learning, Deep Q Networks (DQN), Twin Delayed DDPG (TD3), and Proximal Policy Optimization (PPO), were applied to navigate the challenging MountainCar-v0 environment.

A. Key Findings

1) Q-learning: The traditional Q-learning algorithm, despite its simplicity, provided a baseline for comparison. It demonstrated satisfactory performance by solving the environment in 283,600 episodes within 22 minutes. The discretization of the state space into 12x12 buckets played a crucial role in achieving the desired results.

2) DQN (Deep Q Networks): Employing a neural network to approximate the optimal Q-function, DQN showcased notable efficiency, solving the environment in 1,356 episodes over 1.75 hours. The utilization of deep learning principles contributed to the algorithm's ability to generalize and learn complex patterns.

3) TD3 (Twin Delayed DDPG): TD3, a state-of-the-art deep reinforcement learning algorithm, exhibited a longer training time, solving the environment in 1,156 episodes over 10 hours. The Twin Delayed DDPG algorithm is known for its stability and robustness.

4) PPO (Proximal Policy Optimization): PPO, leveraging vectorized environments with 16 processes, demonstrated exceptional speed in solving the environment. Achieving the solution in 21 episodes within 4 minutes, PPO's efficiency can be attributed to the parallelization of training processes.

B. Comparative Analysis

In summary, each algorithm presented unique strengths and trade-offs. Q-learning, despite its simplicity, played a vital role as a baseline for comparison. DQN showcased the benefits of deep learning in reinforcement learning tasks, emphasizing efficiency and generalization. TD3, while exhibiting stability, required a longer training time. PPO, with its parallelization strategy, delivered remarkable efficiency.

C. Choosing the Right Algorithm

The choice of the most suitable algorithm depends on specific project requirements. For scenarios demanding quick learning and adaptability, DQN and PPO could be preferable due to their efficiency and speed. In contrast, for applications where stability and robustness are critical, TD3 might be the algorithm of choice.

This project's contribution extends beyond successful navigation in specific environments; it provides valuable insights into the comparative performance of reinforcement learning algorithms. The considerations for choosing an algorithm go beyond mere computational efficiency, encompassing factors such as training time, adaptability, and task complexity.

The combination of map-based and reactive navigation strategies, coupled with the exploration of various reinforcement learning algorithms, offers a holistic understanding of robotic systems' adaptability and intelligence across different applications. Future work can build upon these insights to further enhance the capabilities of robotic navigation systems.

Acknowledgment

The preferred spelling of the word "acknowledgment" in America is without an "e" after the "g". Avoid the stilted expression "one of us (R. B. G.) thanks ...". Instead, try "R. B. G. thanks...". Put sponsor acknowledgments in the unnumbered footnote on the first page.

References

[1] H. R. Beom and H. S. Cho, "A sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 25, no. 3, pp. 464–477, 1995.

[2] K. Macek, I. PetroviC, and N. Peric, "A reinforcement learning approach to obstacle avoidance of mobile robots," in *7th International Workshop on Advanced Motion Control. Proceedings (Cat. No. 02TH8623)*. IEEE, 2002, pp. 462–466.

[3] M. Lapan, *Deep Reinforcement Learning Hands-On: Apply modern RL methods, with deep Q-networks, value iteration, policy gradients, TRPO, AlphaGo Zero and more*. Packt Publishing Ltd, 2018.

[4] E. Marchesini and A. Farinelli, "Discrete deep reinforcement learning for mapless navigation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 10 688–10 694.

[5] M. A. K. Jaradat, M. Al-Rousan, and L. Quadan, "Reinforcement based mobile robot navigation in dynamic environment," *Robotics and Computer-Integrated Manufacturing*, vol. 27, no. 1, pp. 135–149, 2011.

[6] C. Xia and A. El Kamel, "A reinforcement learning method of obstacle avoidance for industrial mobile vehicles in unknown environments using neural network," in *Proceedings of the 21st International Conference on Industrial Engineering and Engineering Management 2014*. Springer, 2015, pp. 671–675.

[7] M. Alhawary, "Reinforcement-learning-based navigation for autonomous mobile robots in unknown environments," August 2018.

[8] K. Kozłowski and D. Pazderski, "Modeling and control of a 4-wheel skid-steering mobile robot," *International journal of applied mathematics and computer science*, vol. 14, no. 4, pp. 477–496, 2004.

[9] L. Khriji, F. Touati, K. Benhmed, and A. Al-Yahmedi, "Mobile robot navigation based on q-learning technique," *International Journal of Advanced Robotic Systems*, vol. 8, no. 1, p. 4, 2011.

[10] M. R. Lemos, A. V. R. de Souza, R. S. de Lira, C. A. O. de Freitas, V. J. da Silva, and V. F. de Lucena, "Robot training and navigation through the deep q-learning algorithm," in *2021 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE, 2021, pp. 1–6.