World Academy of Science, Engineering and Technology
International Journal of Computer and Systems Engineering
Vol:18, No:9, 2024

Robot Exploration and Navigation in Unseen Environments Using Deep Reinforcement Learning

Romisaa Ali

Abstract—This paper presents a comparison between twin-delayed Deep Deterministic Policy Gradient (TD3) and Soft Actor-Critic (SAC) reinforcement learning algorithms in the context of training robust navigation policies for Jackal robots. By leveraging an open-source framework and custom motion control environments, the study evaluates the performance, robustness, and transferability of the trained policies across a range of scenarios. The primary focus of the experiments is to assess the training process, the adaptability of the algorithms, and the robot's ability to navigate in previously unseen environments. Moreover, the paper examines the influence of varying environment complexities on the learning process and the generalization capabilities of the resulting policies. The results of this study aim to inform and guide the development of more efficient and practical reinforcement learning-based navigation policies for Jackal robots in real-world scenarios.

Keywords—Jackal robot environments, reinforcement learning, TD3, SAC, robust navigation, transferability, Custom Environment.

I. Introduction

In recent years, reinforcement learning (RL) has emerged as a promising approach for developing intelligent and adaptive control policies for robotics and autonomous systems [1]. A crucial aspect of these systems is their ability to navigate and interact with complex and dynamic environments. Jackal robots, in particular, have been widely employed for various applications in robotics due to their versatility, robustness, and maneuverability [8]. Developing robust and efficient navigation policies for autonomous robots, such as Jackal robots, is a challenging task due to the complexity and variability of real-world environments. The choice of reinforcement learning algorithm used to train these policies can significantly impact their performance, robustness, and adaptability. In this context, two state-of-the-art algorithms, Twin Delayed Deep Deterministic Policy Gradient (TD3) and Soft Actor-Critic (SAC), have shown promising results for various applications. However, their comparative performance in the specific domain of Jackal robot navigation remains underexplored. The objective of this study is to investigate and compare the performance of TD3 and SAC in training navigation policies for Jackal robots using an open-source framework and custom motion control environments. The scope of the research includes evaluating the robustness of the trained policies, their transferability across different scenarios, and the ability of the robots to navigate in previously unseen environments.

This paper is organized as follows: Section I explores how RL enhances Jackal robot controls, while contrasting Twin-Delayed Deep Deterministic Policy Gradient (TD3) and Soft Actor-Critic (SAC) algorithms in robot navigation tasks; Sections II and III detail the integration of the ROS (Robot Operating System) framework with the OpenAI Gym library, utilizing a DRL (deep reinforcement learning) structure based on the PyTorch library. The TD3 and SAC algorithms were employed due to their robustness and strong exploration capabilities, which are essential for handling complex and dynamic environments. Both algorithms are widely implemented in ROS and OpenAI Gym, making them well-suited for integration into our framework for real-time motion control, the environments used for testing include static, dynamic box, and dynamic wall motion control setups, all simulated in real-time. Each environment was designed with varying levels of difficulty to ensure that the robot can adapt to and navigate through unseen situations. The DRL algorithms act as decision-making agents, determining and sending the appropriate control signals to the robot for navigation, The robot's performance was evaluated based on three key metrics: success rate, collision rate, and the average time required to complete one path. Section IV details the experiment design, covering the implementation of TD3 and SAC algorithms, training and evaluation metrics, and training procedures and hyper-parameters. Section V presents the experiment results, including the comparison of training performance, transferability and robustness analysis, and navigation performance in unseen environments. Section VI discusses the key findings and insights, limitations, and challenges, and implications for future research.

Romisaa Ali is with the Department of Computer and Control Engineering (DAUIN), Politecnico di Torino University, Turin, Italy (e-mail: romisaa.ali@polito.it).

This work was supported by REPLY concept company, Torino, Italy.

II. Motion Control Environment

In this study, we employed the Motion Control Continuous Laser environment, designed for various control algorithms in robotic navigation, including classical and motion control approaches. This environment serves as a challenging and realistic setting for testing and evaluating reinforcement learning algorithms for continuous control tasks in robotics. The environment focuses on integrating motion control, laser scan data processing, and continuous action spaces to enable a smooth and practical simulation experience [15].

A. Custom Motion Control Environment Design

The MotionControlContinuousLaser environment is implemented for the Jackal robot in a Gazebo simulation [7], [14]. It provides a continuous action space that consists of linear and angular velocities, allowing the robot to move smoothly within the simulation. In addition, the environment processes the reduced 249-dimensional laser

scan data and incorporates it into the observation space, ensuring that the agent has access to essential sensory information for decision-making. By integrating motion control and laser scan data processing capabilities, the MotionControlContinuousLaser environment presents a complex scenario for testing and evaluating reinforcement learning algorithms in a practical and real-world context, paving the way for future advancements in robotic control and navigation.

B. Jackal Robot Dynamics and Sensors

Sensors play a vital role in the training process of the Jackal robot [8], and two primary sensors are used during the training process.

1) Laser sensor: The laser sensor is integral to the training process as it provides essential data for obstacle detection and avoidance, as well as navigation in the environment. During training, the robot collects laser scan data consisting of 249 beams, which is directly incorporated into the reinforcement learning algorithm. These data enable the agent to learn how to effectively navigate through the environment while avoiding obstacles.

2) IMU sensor: While the IMU sensor data are not directly fed into the reinforcement learning algorithm, it still plays a crucial role in the training process. The IMU sensor measures linear acceleration, angular velocity, and sometimes orientation. These measurements are used by the robot's low-level control system, ensuring stability and proper motion during navigation. The Gazebo simulator replicates the Jackal robot's dynamics by incorporating the IMU sensor data into the simulation, providing a realistic environment for training the agent.

III. Training and Evaluation Scenarios

Our environment is based on the Motion Control Continuous Environment and consists of three types of navigation environments: static environments, dynamic box environments, and dynamic wall environments. These environments were used for both training and evaluation to assess the environments. Our aim is to carry out a comprehensive comparison between two distinct reinforcement learning algorithms: the TD3 and the SAC. The primary objective of this investigation is to delve into these algorithms' abilities to generalize and successfully operate within unseen static environments.

A. Static Environments

These environments feature red cylindrical obstacles representing obstacle-occupied spaces and blue cylindrical barriers forming the borders, with an open side at the goal place. The static environments challenge the agent to navigate around fixed obstacles while learning the dynamics of the robot. As shown in Fig. 1, the static environment provides a controlled setup for evaluating the robot's navigation capabilities.

B. Dynamic Box Environments

These environments feature blue box-shaped obstacles representing moving obstacles and blue barriers forming three sides of the environment, leaving one side open. The dynamic box environments challenge the agent to navigate around these moving obstacles, which are randomly placed and move within the environment. As shown in Fig. 2, the dynamic box environment provides a complex setup for testing the robot's ability to adapt to dynamic changes and navigate efficiently.

C. Dynamic Wall Environments

These environments feature blue cylindrical barriers forming three sides of the environment, leaving one side open, and two long blue walls moving in opposite directions with slight angles. The walls' velocities are perpendicular to the start-goal direction, creating a challenging navigation scenario. The robot can only pass when the two walls are moving apart. To challenge the agent, small variances are added to each wall's length, tilting angle, and magnitude of the velocity. As depicted in Fig. 3, the dynamic wall environment presents a unique challenge by introducing continuously moving obstacles. Additional resources can be found at: [5], [17], [18].

IV. Experiment Design

In the original design of the Motion Control Continuous Environment, the author employed three distinct types of navigation environments: static, dynamic box, and dynamic wall environments. These environments were developed to offer varying levels of complexity, thereby facilitating the reinforcement learning agent's capacity to learn, adapt, and navigate with efficiency. However, in the context of our research, we have chosen to exclusively focus on the static environments. Our aim is to carry out a comprehensive comparison between two distinct reinforcement learning algorithms: the TD3 and the SAC. The primary objective of this investigation is to delve into these algorithms' abilities to generalize and successfully operate within unseen static environments.

A. TD3 and SAC Robustness Overview

TD3, introduced by Fujimoto et al. at NeurIPS [16], significantly advances actor-critic methodologies by addressing overestimation bias and improving learning speed in continuous control tasks. It features dual Critic networks and delayed policy updates to enhance policy quality and reduce overfitting risks. Moreover, TD3 utilizes noise, specifically through the Ornstein-Uhlenbeck process, to encourage action space exploration during training. Empirical evaluations on OpenAI gym environments show that it outperforms several state-of-the-art methods, including DDPG. They propose a novel approach that combines the Twin Delayed Deep Deterministic policy gradient algorithm (TD3) with a model-based exploration strategy to enhance both learning efficiency and performance, Xu and Liu introduced a paper that includes experiments on various

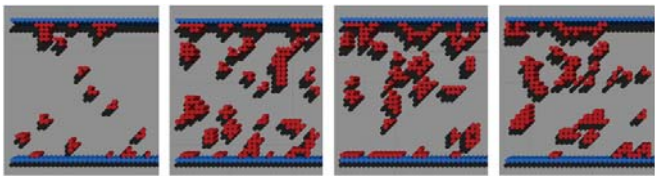

Fig. 1 Static Box Environment

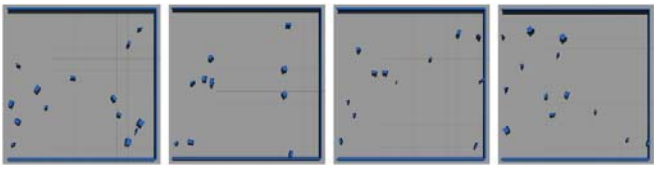

Fig. 2 Dynamic Box Environment

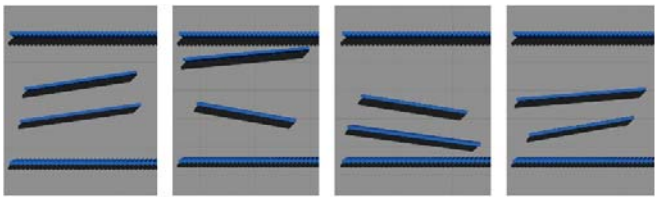

Fig. 3 Dynamic wall Environment

benchmark environments from OpenAI Gym, demonstrating the effectiveness of the proposed approach in addressing the challenges of sample inefficiency and exploration in continuous control tasks [3]. Numerous papers have explored the utility of TD3 in various contexts; for instance, the work by Wu and Wu [9], as well as another paper by Tan, and others on the application of TD3 for dynamic path planning [10]. These studies provide further evidence of TD3's versatility and effectiveness in solving complex real-world problems. Similarly, Soft Actor-Critic (SAC), introduced by Haarnoja et al. in 2018 [4], is a model-free deep reinforcement learning method that utilizes a maximum entropy framework to foster exploration while maximizing expected rewards. SAC's architecture includes two critic networks (Q1 and Q2) and target networks, which enhance stability and reduce overestimation. These features have allowed SAC to achieve consistent performance across different random seeds and surpass other methods on continuous control benchmark tasks. In the paper 'Hierarchical Foresight: Self-Supervised Learning of Long-Horizon Tasks via Visual Subgoal Generation' by Sermanet et al. [2], SAC is integrated with a hierarchical architecture for learning complex, long-horizon tasks through the generation of visual subgoals. Another work, 'Addressing Sample Inefficiency and Exploration in Model-Based Reinforcement Learning' by Srivastava et al. [6], focuses on enhancing sample efficiency and exploration in the broader context of model-based reinforcement learning. Together, these works demonstrate the versatility and robustness of SAC in handling a variety of complex challenges. Numerous researchers have investigated SAC's utility in diverse applications. Papers by Nakhleh and Raza have contributed to the literature on SAC [11], as have works by Martin and Chekroun [12]. Another noteworthy study by Chavali and Gupta has also provided valuable insights into the algorithm's performance and applicability [13]. These studies further underline the versatility and efficacy of SAC in tackling complex real-world challenges.

B. Training and Evaluation Metrics

A comparative analysis of TD3 and SAC will be conducted using a range of performance indicators. Metrics such as success rate, collision rate, episode length and return, along with the time-averaged number of steps, will be employed. These parameters will provide an extensive assessment of the performance and robustness of each algorithm within reinforcement learning environments.

C. Training Procedure and Hyperparameter

This section highlights the application of both the Soft Actor-Critic (SAC) and Twin Delayed DDPG (TD3) algorithms. Both algorithms employ critical hyperparameters like actor and critic learning rates, specific exploration parameters in policy arguments, and distinct reward structures. A unique feature of these implementations is their parallelization strategy, utilizing isolated environments within Singularity containers [19]. Here, up to 10 actors run simultaneously, promoting efficient and extensive exploration. Furthermore, the training, implemented in a ROS operating system, involves the collection of trajectories and subsequent deep neural network calculations at regular intervals, effectively optimizing the policy and enhancing learning robustness.

V. Experiment Results

A. Training Performance Comparison

During the training phase, the TD3 demonstrated a higher average success rate throughout the entire training period compared to the SAC. However, it was the SAC that recorded the peak success rate of 77%, while TD3 reached a peak of 6%. At the conclusion of the training period, the average success rate of SAC exceeded that of TD3, registering at 69% against TD3's 67% in the final logged data but the TD3 is more stable in the average value. Both algorithms exhibited remarkable stability throughout the training, with minor oscillations around a central value, indicating reliable and robust performance. Although the convergence rate was relatively slow for both algorithms, the steadiness of their performance was notable. Interestingly, SAC seemed to initiate the training with a higher success rate, but over time, TD3 gradually surpassed it, as shown in Fig. 4. This underscores the distinct characteristics and capabilities of these two reinforcement learning algorithms in terms of training performance. During the training process, both TD3 and SAC algorithms initially demonstrated a high number of steps required to complete a single path. Particularly, SAC peaked at approximately 257 steps at the 170,000th training step. However, as the training continued, TD3 consistently needed fewer steps to complete a path, thereby exhibiting superior efficiency compared to SAC. The duration required

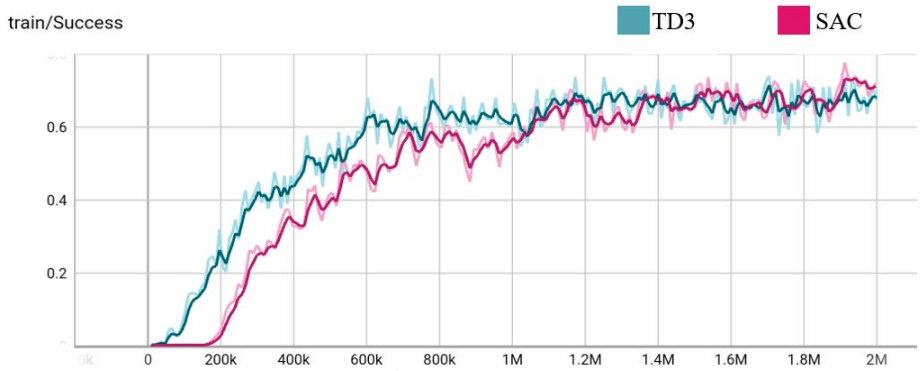

Fig. 4 Success rate

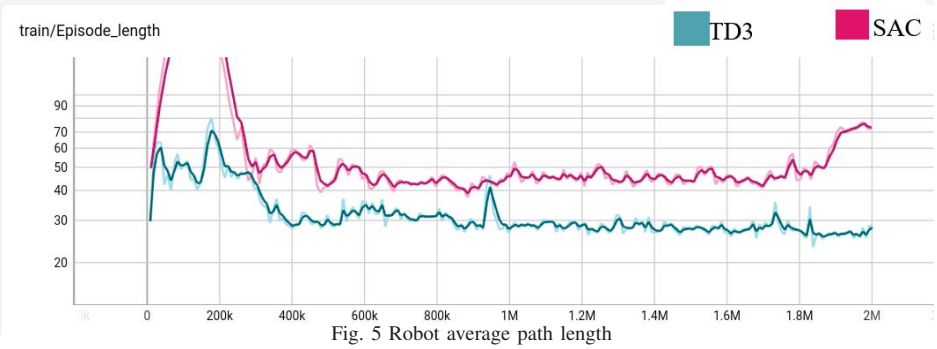

Fig. 5 Robot average path length

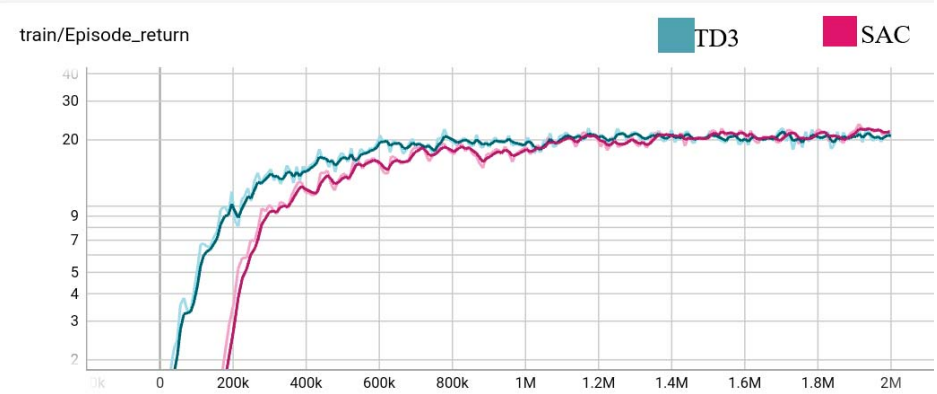

Fig. 6 The average return

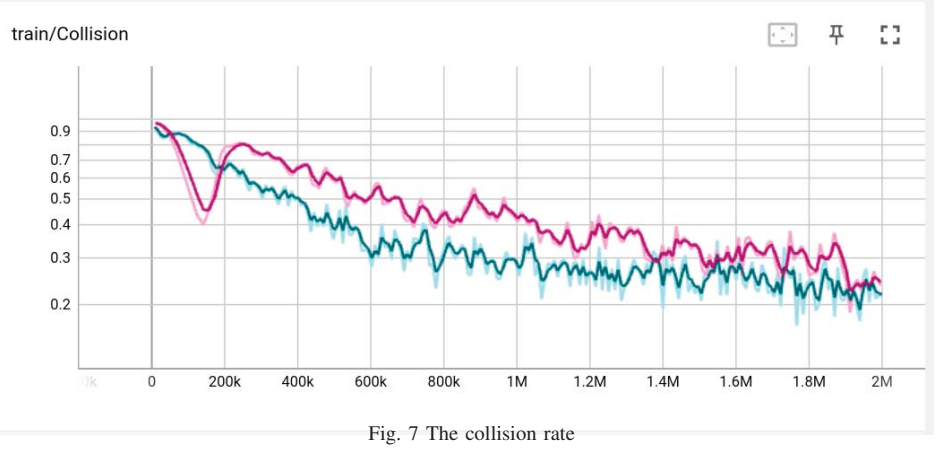

Fig. 7 The collision rate

for each episode mirrored the episode length, showcasing the strong correlation between the number of steps and time. Hence, an increase in steps directly resulted in longer episode durations, as illustrated in Fig. 5. Therefore, TD3's proficiency in minimizing episode length also translated into a reduction in episode duration, effectively demonstrating its time efficiency. When comparing the average return throughout the training process, TD3 demonstrated a swift rise in average return from the beginning of training, consistently achieving higher values throughout. On the other hand, SAC's average return did not show significant growth until after 200Ktraining steps, remaining relatively stable during the initial stages. As shown in Fig. 6, TD3 surpasses SAC in terms of higher average return across the training period. In terms of collision rates, an interesting dynamic was observed between the SAC and TD3 algorithms throughout the training period. SAC exhibited a significant reduction in collisions, reaching an improvement of approximately 40% prior to the 400,000th step. However, this lower collision rate was not consistently maintained beyond that point. Over the entirety of the training process, it was TD3 that demonstrated a lower average collision rate, surpassing SAC's performance. These findings are illustrated in Fig. 7. Table I presents an overview of the metrics, specifically reflecting the averages from the last training logs. This summary table encapsulates essential parameters like success rate, episode length, and collision rate, providing an easy-to-reference comparison between the TD3 and SAC algorithms. By offering a snapshot of the complete training performance.

B. Transferability and Robustness Analysis

This involves using a different world scenario from the repository, which was not included in the training stage see the robot navigation in Figs. 8 and 9. In this unfamiliar environment, 1,000 robot paths are created for evaluation. Key performance indicators, including collision rates, successful path completions, and average time taken for path completion are assessed for each algorithm. This analysis aids in understanding the capability of the algorithms to adjust and perform in new environments.

C. Unseen Environment Navigation Performance

The TD3 algorithm successfully completed 52% of the paths in the new environment, with an average time of 141.58 seconds for each path completion. However, it encountered a collision rate of 48%. On the other hand, the SAC algorithm had a slightly lower success rate, completing 46% of the paths. Despite its lower success rate, it was found to take a significantly longer time for each path completion, averaging at 243.63 seconds. The SAC algorithm also experienced a higher collision rate of 54%. From these findings, the TD3 algorithm demonstrated a higher ability to generalize and perform in the unfamiliar environment, with a higher success rate and lower average time compared to the SAC algorithm, see Table II, Figs. 8 and 9.

TABLE I
SUMMARY OF TRAINING PERFORMANCE METRICS FOR TD3 AND SAC ALGORITHMS

Metric	TD3	SAC
Collision Rate	21%	26%
Steps per path	28.4	72.3
Average return	20.7	21.4
Success Rate	67%	69%
Time per path	5.7	172.0

TABLE II
PERFORMANCE SUMMARY OF TD3 AND SAC IN UNSEEN STATIC ENVIRONMENTS

Algorithm	Success Rate	Collision Rate	Average Time (sec)
TD3	52%	48%	141.58
SAC	46%	54%	243.63

VI. CONCLUSION AND RECOMMENDATION

To sum up, the TD3 algorithm consistently outperformed SAC, making it a more effective choice for tasks that require adaptability, efficiency, and resilience when faced with changes in the environment. While SAC achieved the highest success rate at certain points during the training process, TD3 demonstrated more stability in maintaining a high success rate throughout the entire training. This consistency highlights the robustness of TD3, both in its efficient path completion during training and its superior ability to generalize in new, unseen environments. Importantly, TD3 not only had a higher success rate in the transferability tests but also completed paths in less time on average, demonstrating its strong capability to adapt to and efficiently navigate unseen environments. This time efficiency advantage was a consistent feature of TD3, apparent in both the training and testing stages, suggesting its potential superiority in situations that require fast responses and efficient path planning. Although both algorithms have their unique strengths, these findings suggest a potential preference for TD3 in tasks requiring consistent high performance, quick execution, and adaptability to new environments. In future studies, it would be beneficial to enhance and fine-tune these reinforcement learning algorithms under a more diverse set of scenarios. This would facilitate a deeper understanding of their effective applications and aid in selecting the most suitable algorithm for different tasks.

ACKNOWLEDGMENT

Deep appreciation is conveyed to Professor Marcello Chiaberge for his invaluable supervision and guidance throughout this research. Warm thanks go to the SCUDO Office at Politecnico di Torino for their essential support and assistance. Recognition must also be given to REPLY Concepts company, especially to Mr. Maurizio Griva and Mr. Simone Voto, for their invaluable input and collaboration. Finally, sincere gratitude is extended to the Department of DAUIN at Politecnico di Torino, as well as to the dedicated staff of PIC4SeR, for their cooperation and support in the execution of this work. The contributions of Mr. Zifan Xu from the University of Texas at Austin were particularly

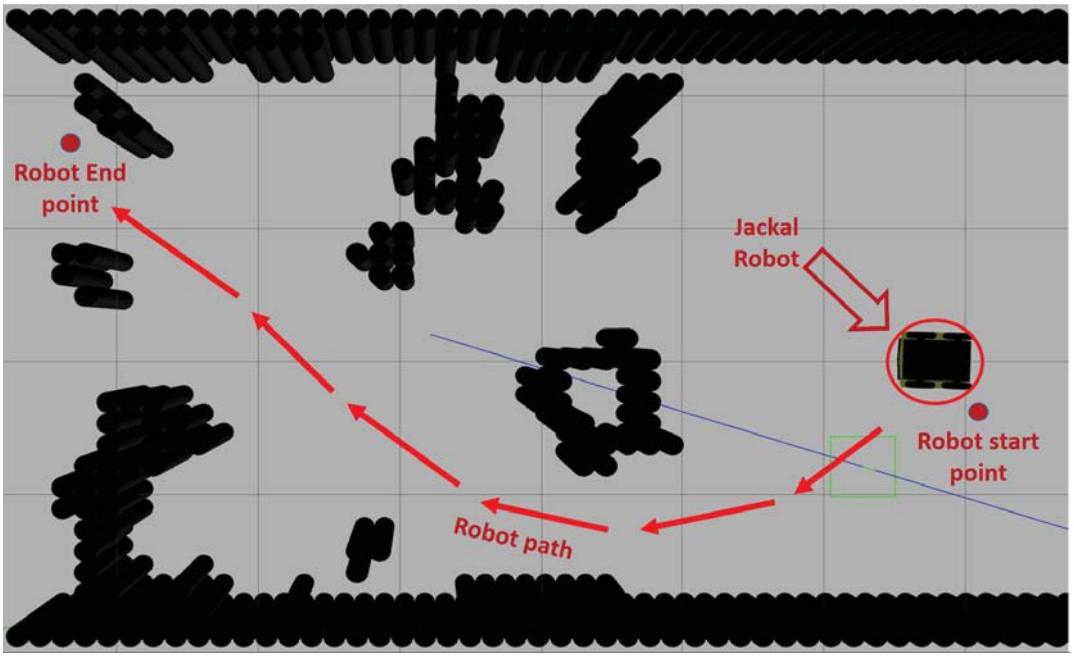

Fig. 8 Start point of the Jackal robot's path in the Gazebo simulation in an unseen environment

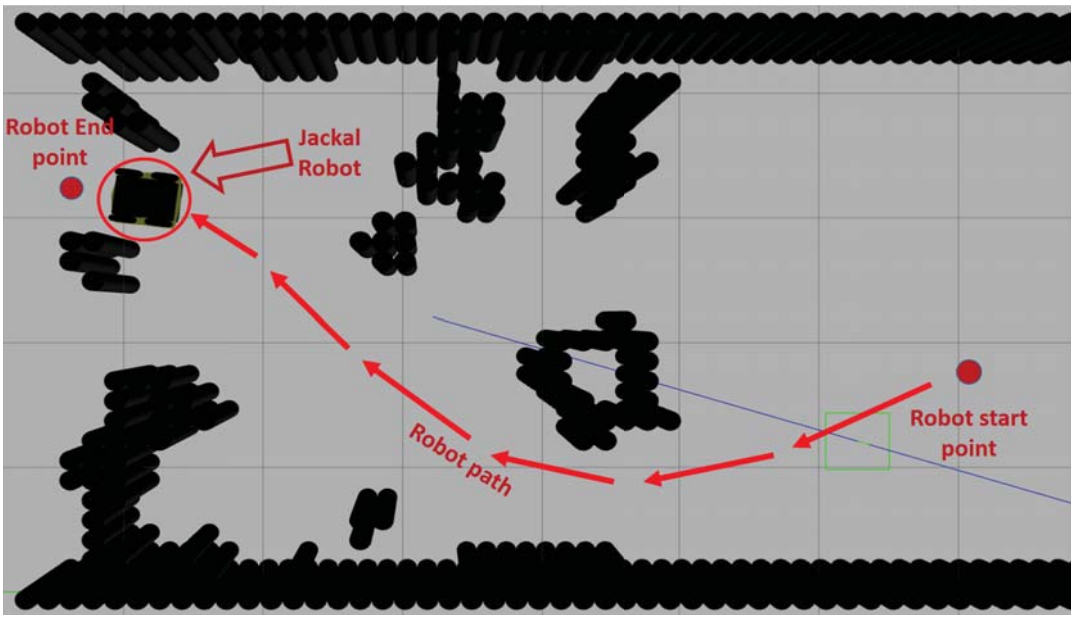

Fig. 9 End point of the Jackal robot's path in the Gazebo simulation in an unseen environment

instrumental to the success of this work and are deeply acknowledged.

REFERENCES

[1] J. D. Johnson, J. Li, and Z. Chen, "Reinforcement Learning: An Introduction: R.S. Sutton, A.G. Barto, MIT Press, Cambridge, MA 1998, 322 pp. ISBN 0-262-19398-1," *Neurocomputing*, vol. 35, no. 1-4, pp. 205–206, 2000.

[2] S. Nair and C. Finn, "Hierarchical Foresight: Self-Supervised Learning of Long-Horizon Tasks via Visual Subgoal Generation," in *8th International Conference on Learning Representations, ICLR 2020*, Addis Ababa, Ethiopia, 2020. Online. Available: https://openreview.net/forum?id=H1gzR2VKDH

[3] Z. Xu, B. Liu, X. Xiao, A. Nair, and P. Stone, "Benchmarking Reinforcement Learning Techniques for Autonomous Navigation," *CoRR*, vol. abs/2210.04839, 2022. Online. Available: https://doi.org/10.48550/arXiv.2210.04839

[4] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, J. G. Dy and A. Krause, Eds., vol. 80, Stockholmsmässan, Stockholm, Sweden, 2018, pp. 1856–1865. Online. Available: http://proceedings.mlr.press/v80/haarnoja18b.html

[5] Z. Xu, B. Liu, X. Xiao, A. Nair, and P. Stone, "Benchmarking Reinforcement Learning Techniques for Autonomous Navigation," in *IEEE International Conference on Robotics and Automation (ICRA)*, London, UK, 2023, pp. 9224–9230. Online. Available: https://doi.org/10.1109/ICRA48891.2023.10160583

[6] A. S. Anand, J. E. Kveen, F. J. Abu-Dakka, E. I. Grøtli, and

J. T. Gravdahl, "Addressing Sample Efficiency and Model-bias in Model-based Reinforcement Learning," in *21st IEEE International Conference on Machine Learning and Applications (ICMLA)*, Nassau, Bahamas, 2022, pp. 1–6. Online. Available: https://doi.org/10.1109/ICMLA55696.2022.00009

[7] N. P. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, 2004, pp. 2149–2154. Online. Available: https://doi.org/10.1109/IROS.2004.1389727

[8] Y. Chen, C. Rastogi, and W. R. Norris, "A CNN Based Vision-Proprioception Fusion Method for Robust UGV Terrain Classification," *IEEE Robotics Autom. Lett.*, vol. 6, no. 4, pp. 7965–7972, 2021. Online. Available: https://doi.org/10.1109/lra.2021.3101866

[9] J. Wu, Q. M. J. Wu, S. Chen, F. Pourpanah, and D. Huang, "A-TD3: An Adaptive Asynchronous Twin Delayed Deep Deterministic for Continuous Action Spaces," *IEEE Access*, vol. 10, pp. 128077–128089, 2022. Online. Available: https://doi.org/10.1109/ACCESS.2022.3226446

[10] Y. Tan, Y. Lin, T. Liu, and H. Min, "PL-TD3: A Dynamic Path Planning Algorithm of Mobile Robot," in *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Prague, Czech Republic, Oct. 9-12, 2022, pp. 3040–3045. Online. Available: https://doi.org/10.1109/SMC53654.2022.9945119

[11] K. Nakhleh, M. Raza, M. Tang, M. Andrews, R. Boney, I. Hadzic, J. Lee, A. Mohajeri, and K. Palyutina, "SACPlanner: Real-World Collision Avoidance with a Soft Actor Critic Local Planner and Polar State Representations," in *IEEE International Conference on Robotics and Automation (ICRA)*, London, UK, May 29 - June 2, 2023, pp. 9464–9470. Online. Available: https://doi.org/10.1109/ICRA48891.2023.10161129

[12] J. B. Martin, R. Chekroun, and F. Moutarde, "Learning from demonstrations with SACR2: Soft Actor-Critic with Reward Relabeling," *CoRR*, vol. abs/2110.14464, 2021. Online. Available: https://arxiv.org/abs/2110.14464

[13] L. Chavali, T. Gupta, and P. Saxena, "SAC-AP: Soft Actor Critic based Deep Reinforcement Learning for Alert Prioritization," in *IEEE Congress on Evolutionary Computation, CEC 2022*, Padua, Italy, July 18-23, 2022, pp. 1–8. Online. Available: https://doi.org/10.1109/CEC55065.2022.9870423

[14] N. P. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, September 28 - October 2, 2004, pp. 2149–2154. Online. Available: https://doi.org/10.1109/IROS.2004.1389727

[15] B. Siciliano and O. Khatib, "Robotics and the Handbook," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds., Springer, 2016, pp. 1–10. Online. Available: https://doi.org/10.1007/978-3-319-32552-1_1

[16] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing Function Approximation Error in Actor-Critic Methods," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, vol. 80, pp. 1582–1591, 2018. Online. Available: http://proceedings.mlr.press/v80/fujimoto18a.html

[17] Daffan. (n.d.). *GitHub - Daffan/ros_jackal: ROS-Jackal environment for RL*. GitHub. https://github.com/Daffan/ros_jackal

[18] Xiao, J. (n.d.). *Benchmarking Reinforcement Learning Techniques for Autonomous Navigation*. Retrieved from https://cs.gmu.edu/~xiao/Research/RLNavBenchmark/

[19] Sylabs, "Admin Guide: Installation on Linux," *Sylabs Documentation*, 2023. Online. Available: https://docs.sylabs.io/guides/latest/admin-guide/installation.html