TD3-Based DRL for Skid-Steering Robot Navigation in Multi-Scale Environments

Carlos J. F. S. Júnior[†] Carlos E. S. Cerqueira[*] André. G. S. Conceição[*†] Tiago T. Ribeiro[*†]
[*]Department of Electrical and Computer Engineering, Federal University of Bahia, Salvador, Brazil
[†]Graduate Program in Electrical and Computer Engineering, Federal University of Bahia, Salvador, Brazil
{carlos.fonseca, cerqueira.carlos, andre.gustavo, tiagotr}@ufba.br

***Abstract*—This work explores the use of Deep Reinforcement Learning (DRL) for autonomous navigation of a skid-steering ground robot. The Twin-Delayed Deep Deterministic Policy Gradient (TD3) algorithm is employed to train a control policy that guides a LiDAR-equipped robot to reach relative targets while avoiding obstacles. The approach is evaluated in environments with varying dimensionality, demonstrating convergence and the ability to generalize navigation strategies across different scenarios. However, training in large-scale environments presented significant challenges, particularly in achieving stable convergence. The results indicate a strong dependence on the relative dimensions of the environment, with policy performance tending to degrade in overly sparse or disproportionate spatial configurations.**

***Index Terms*—Skid-steering robot, Autonomous navigation, Deep reinforcement learning, TD3 algorithm, Obstacle avoidance.**

I. Introduction

Autonomous navigation remains one of the core challenges in mobile robotics, especially in unstructured or dynamic environments where robots must perceive, plan, and act with minimal human intervention. This challenge becomes even more pronounced when dealing with ground robots that exhibit skid-steering kinematics and possess large physical dimensions relative to the operational environment. Such platforms are common in industrial, agricultural, and surveillance scenarios, yet they require more sophisticated control strategies to avoid collisions and maintain smooth trajectories in confined spaces [1].

Traditional navigation architectures rely on modular pipelines involving perception, localization, mapping, planning, and control subsystems [2]. While effective, these systems often depend heavily on accurate environmental models and handcrafted rules, limiting their adaptability in real-world conditions [3]. In contrast, Deep Reinforcement Learning (DRL) methods allow agents to learn control policies directly from sensor interaction, enabling more generalizable and adaptive behaviors [4], [5], [6].

Among DRL algorithms, the Twin-Delayed Deep Deterministic Policy Gradient (TD3) stands out for its stability and robustness in continuous action domains [7]. TD3 addresses common issues found in actor-critic methods, such as overestimation bias and unstable policy updates, by employing double Q-networks, delayed policy updates, and target smoothing.

In this work, we evaluate the applicability of the goal-driven autonomous exploration strategy proposed by Cimurs et al. [8] for the autonomous navigation of a skid-steering ground robot, namely the Clearpath Husky UGV. Specifically, we assess the performance of a TD3-based control policy in environments with varying spatial scales. The robot, a skid-steering ground platform with non-holonomic constraints, operates in scenarios where its physical dimensions may become significant relative to the available space. Simulations are carried out using Gazebo simulator integrated with the Robot Operating System (ROS) framework, which enables modular communication between perception, control, and training components.

Navigation targets are defined relative to the robot's local frame, and a LiDAR sensor provides environmental perception. The training process is carried out across three simulated environments of varying spatial dimensions and obstacle density, allowing us to assess the convergence, robustness, and generalization of the learned policy under different navigation complexities [9], [10].

Results demonstrate that the TD3-based controller successfully enables smooth, goal-oriented, and collision-free navigation. However, the learned policy exhibited greater difficulty in generalizing to certain scenarios, particularly those involving narrow passages or environments with extreme aspect ratios relative to the robot's size. These limitations highlight the influence of spatial configuration on policy transferability and reinforce the need for evaluating DRL-based strategies in diverse and realistic conditions.

The remainder of this paper is organized as follows. Section II reviews related work on the evaluation of DRL methods in mobile robotics, with a particular emphasis on TD3 and its applicability to skid-steering platforms. Section III details the methodology, including the robot model, control policy, reward structure, and system integration and the experimental setup, highlighting the three simulated environments. Section V presents the evaluation results from both training and testing, focusing on convergence behavior, navigation performance, and policy generalization. Finally, Section VI concludes the paper with a summary of the main findings and outlines directions for future evaluation and development.

The authors acknowledges the financial support provided by CNPq-Brazil (grant numbers **201532/2024-7, 442706/2023-5 e 407163/2022-0**), FNDCT-Brazil and MCTI-Brazil and UFBA (EDITAL PROPCI/PROPG-UFBA 007/2022 - JOVEMPESQ, project no. **24387**).

II. Related Work

The use of Deep Reinforcement Learning (DRL) in autonomous navigation has gained significant attention due to its ability to produce adaptive control policies without requiring precise environmental models [11], [12]. In DRL-based navigation, agents learn optimal actions by interacting with the environment and receiving feedback through reward signals, which has shown promising results across a range of robotic applications [13], [14].

Among the various DRL algorithms, the Twin-Delayed Deep Deterministic Policy Gradient (TD3) has demonstrated strong performance in continuous action domains due to its improvements over traditional actor-critic methods, such as double Q-learning, delayed policy updates, and target smoothing [15] [16]. These features make TD3 a compelling candidate for robotic navigation, particularly in scenarios that demand precise and robust control.

Cimurs et al. [8] proposed a goal-driven autonomous exploration strategy using DRL with a non-holonomic differential-drive robot, achieving effective navigation performance in simulated environments. However, their approach has not been explicitly evaluated for skid-steering robots, which introduce unique challenges such as higher slippage and reduced turning precision. Additionally, while the original study demonstrated generalization across several maps, it did not focus on how the robot's physical dimensions relative to the environment might affect navigation performance.

Although DRL has been widely applied to robotic navigation, the literature specifically addressing the use of the TD3 algorithm in skid-steering platforms remains scarce. To the best of our knowledge, only a few works have explored reinforcement learning for such platforms, and even those rely on alternative algorithms. For example, Dai et al. [17] proposed a knowledge-assisted DDPG method for driving torque distribution in skid-steering vehicles, aiming to improve tracking performance. However, no prior study has conducted a systematic evaluation of TD3-based navigation policies on skid-steering robots operating in environments with varied spatial dimensions, as addressed in this work.

This gap is critical, as many real-world applications, such as those in agriculture, mining, and surveillance, deploy large skid-steering platforms in confined or geometrically varied environments, where traditional DRL policies may not generalize effectively. Few studies have addressed how environmental scale, obstacle density, and aspect ratio influence policy robustness when using DRL-based controllers on such platforms [18] [19].

In this work, we build upon the method proposed by Cimurs et al. and conduct a systematic evaluation of its applicability to a skid-steering ground robot with large relative dimensions, simulated in Gazebo. Our goal is to assess the policy's ability to generalize under varying environmental constraints and identify limitations specific to this class of robotic systems.

III. Methodology

This section presents the methodology used to train and evaluate the TD3 algorithm for autonomous navigation of a skid-steering ground robot in simulated environments. The setup includes the robot model, the simulation environments, software stack, TD3 architecture, action and observation spaces, reward function, and training strategy.

A. Robot Kinematics and Action Mapping

Action mapping plays a central role in reinforcement learning for mobile robots, as it defines how the agent's policy outputs translate into physical motion. For wheeled robots, two common configurations are differential drive and skid-steering. Although both are governed by similar control inputs, left and right wheel or tread velocities (v_l, v_r), their underlying kinematics differ substantially, with important implications for policy learning.

In the differential-drive model, it is typically assumed that each wheel maintains a single point of contact with pure rolling and no lateral slip. The linear and angular velocities (v, ω) of the robot's body in its local frame are derived from:

$$\begin{aligned} v &= \frac{v_r + v_l}{2}, \\ \omega &= \frac{v_r - v_l}{2d}, \end{aligned} \tag{1}$$

where d is half the track width. This simple kinematic model provides a deterministic mapping from control to motion, facilitating convergence in actor-critic methods like TD3.

However, in skid-steer vehicles, such as the Husky platform, this assumption does not hold. Each side of the drive system interacts with the ground through multiple contact points, and motion relies on controlled slippage. As a result, the vehicle's movement is governed by tread-dependent Instantaneous Centers of Rotation (ICRs). As per Mandow et al. [20], the robot's local frame velocities (v_x, v_y, ω_z) are related to the tread velocities (v_l, v_r) by:

$$\begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix} = \mathbf{A} \begin{bmatrix} v_l \\ v_r \end{bmatrix}, \tag{2}$$

with the matrix $\mathbf{A}$ defined as:

$$\mathbf{A} = \frac{1}{x_{\text{ICR}r} - x_{\text{ICR}l}} \begin{bmatrix} -y_{\text{ICR}v} \cdot \alpha_l & y_{\text{ICR}v} \cdot \alpha_r \\ x_{\text{ICR}r} \cdot \alpha_l & -x_{\text{ICR}l} \cdot \alpha_r \\ -\alpha_l & \alpha_r \end{bmatrix}, \tag{3}$$

where:

- $x_{\text{ICR}l}$, $x_{\text{ICR}r}$, and $y_{\text{ICR}v}$ are local ICR coordinates for the left tread, right tread, and vehicle centerline, respectively;
- α_l and α_r are correction coefficients reflecting mechanical asymmetries such as tire pressure or wear.

These coordinates are not fixed, but vary with dynamics, mass distribution, terrain, and control inputs. This introduces an indirect and non-stationary relationship between actions and resulting states. Thus, the true motion outcome from an

action (v_l, v_r) depends on latent variables that are not directly observable by the agent. Consequently, value-based methods like TD3 struggle to generalize, as the reward and transition functions vary implicitly with state, making temporal-difference updates unstable.

In their analysis, Mandow et al. introduce two metrics that help quantify slippage and asymmetry:

$$\chi = \frac{L}{x_{\text{ICR}r} - x_{\text{ICR}l}}, \quad 0 < \chi \leq 1, \tag{4}$$

$$e = \frac{x_{\text{ICR}r} + x_{\text{ICR}l}}{x_{\text{ICR}r} - x_{\text{ICR}l}}. \tag{5}$$

Here, L is the lateral distance between tread centerlines. Lower values of χ indicate greater slippage, and $e \approx \pm 1$ signals asymmetric loading or terrain-induced imbalance.

These metrics are particularly relevant for policy evaluation. As $\chi \to 0$ or $e \to \pm 1$, the learning agent must infer and adapt to asymmetric slippage, increasing sample complexity and exploration demands.

In contrast, the differential-drive model assumes fixed ICRs and zero lateral velocity $v_y = 0$, thereby eliminating such sources of uncertainty. Consequently, although TD3 may converge more rapidly in differential-drive robots, skid-steer systems introduce an additional challenge: the need to learn control policies under latent and varying dynamics. This makes them a more demanding, yet more realistic, benchmark for applying DRL to robotics.

a) Challenges for TD3-based Navigation: TD3 is a model-free algorithm that must implicitly learn the dynamics governing the state transitions. In the presence of non-negligible β, the transition function $s_{t+1} = f(s_t, a_t)$ becomes nonlinear and history-dependent, making convergence difficult. The learned Q-function must generalize over a state-action space where similar actions may result in drastically different state transitions due to unmodeled slip.

This lack of deterministic forward consistency increases the variance in the TD3 targets:

$$y = r + \gamma \cdot \min_{i=1,2} Q_{\phi_i'}(s', \pi_{\theta'}(s') + \epsilon), \tag{6}$$

and thus can hinder convergence or lead to unstable learning. These dynamics amplify in confined or low-friction environments, where β is larger or variable, as explored in this paper.

b) Action Mapping: The TD3 actor network outputs two continuous actions $a_1, a_2 \in [-1, 1]$, which are mapped to the robot's linear and angular velocity commands through a scaled hyperbolic tangent function:

$$v = v_{\text{max}} \cdot \tanh(a_1), \quad \omega = \omega_{\text{max}} \cdot \tanh(a_2). \tag{7}$$

This mapping constrains the control signals to a safe, bounded range while preserving the expressiveness of continuous action spaces. The resulting (v, ω) pair is used to command the robot's motion in simulation via Gazebo's physics engine.

B. TD3 Algorithm, State Representation, and Control Model

The Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm is a model-free, off-policy actor-critic method tailored for continuous control tasks. It improves upon prior actor-critic methods (e.g., DDPG) by incorporating three key strategies:

- **Clipped Double Q-Learning:** Two critic networks Q_{ϕ_1} and Q_{ϕ_2} are trained independently, and the minimum of their estimates is used to mitigate overestimation bias. The target computation follows (6), where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is a clipped noise term for target smoothing.
- **Delayed Policy Updates:** The actor is updated less frequently than the critics to enhance stability.
- **Target Policy Smoothing:** Noise is added to the target action during critic updates to improve robustness.

The actor is trained to maximize the expected Q-value predicted by Q_{ϕ_1}:

$$\mathcal{L}_{\text{actor}} = -\mathbb{E}_{s \sim D} \left[Q_{\phi_1}(s, \pi_\theta(s)) \right]. \tag{8}$$

The critics minimize the squared temporal-difference error:

$$\mathcal{L}_{\text{critic}} = \mathbb{E}_{(s,a,r,s') \sim D} \left[\left(Q_\phi(s, a) - y \right)^2 \right]. \tag{9}$$

a) State Representation: The robot uses onboard sensing for navigation. It is equipped with wheel odometry and a Velodyne VLP-16 LiDAR, a 16-channel 3D sensor with a 360° horizontal field of view. To reduce input dimensionality, the LiDAR scan is downsampled to 20 evenly spaced distance readings.

The final 24-dimensional state vector $s \in \mathbb{R}^{24}$ includes:

- 20 LiDAR distance readings;
- 2 polar coordinates (distance and angle) to the goal;
- 2 previous control actions (v_{t-1}, ω_{t-1}).

The target position is defined in the global frame and transformed at each time step to the robot's local frame. The policy operates without global maps, relying only on onboard sensors and relative goal positioning.

b) Network Architecture: The actor network processes the 24-dimensional input through two fully connected layers with 800 and 600 neurons (ReLU activations), followed by a $\tanh$-activated output layer producing normalized linear and angular velocity commands in $[-1, 1]$ (see Fig. 1):

$$v = v_{\text{max}} \cdot v_1, \qquad \omega = \omega_{\text{max}} \cdot v_2. \tag{10}$$

Each of the twin critic networks receives the same 24-dimensional state concatenated with the 2-dimensional action, forming a 26-dimensional input. This passes through a fully connected layer with 800 neurons, followed by a 600-neuron layer and a scalar output layer for the Q-value estimation (Fig. 2).

This design enhances sample efficiency and learning stability, enabling robust policy training for autonomous navigation in unknown environments.

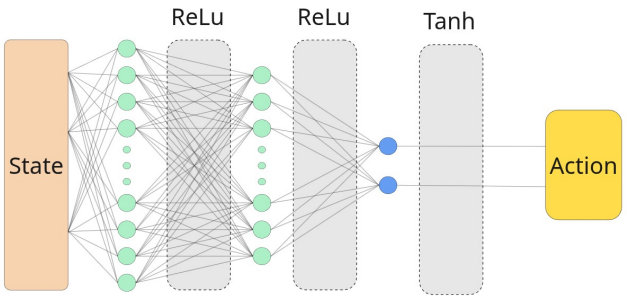


Fig. 1: Architecture of the actor network mapping 24-dimensional states to two continuous control actions.

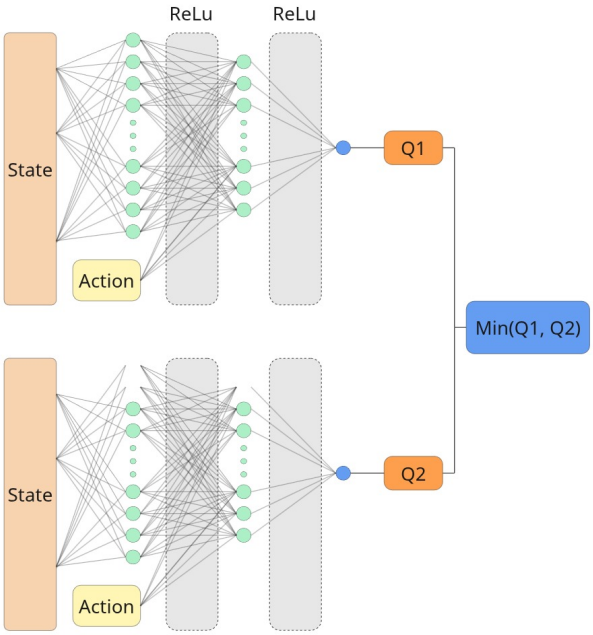


Fig. 2: Architecture of the twin critic networks. The minimum of their outputs is used during target calculation.

C. Simulation Framework and Training Environments

To implement and evaluate the TD3 algorithm in a realistic simulation context, we adopted a software stack composed of ROS, Gazebo, RViz, PyTorch, and TensorBoard. ROS served as the communication backbone, allowing modular development through a publish-subscribe architecture. It facilitated integration between perception, control, and learning components.

Gazebo was used to simulate the robot and its interaction with static environments. As a 3D physics-based simulator compatible with ROS, Gazebo provides accurate models for dynamics and sensor behavior. To visualize robot data, LiDAR readings, and environment feedback in real time, we employed RViz, which integrates seamlessly with ROS and allows debugging through interactive 3D visualization.

The deep reinforcement learning model was implemented and trained using PyTorch, a widely adopted framework for dynamic graph-based deep learning. Training performance and convergence were monitored with TensorBoard, enabling interactive tracking of loss metrics and policy evolution.

To assess generalization capabilities of the TD3 policy, three training environments were developed in Gazebo, each containing static obstacles. Particular care was taken to avoid hollow or narrowly spaced objects that could mislead the LiDAR sensor, potentially suggesting false free paths and degrading learning performance. The environments varied in spatial dimensions and obstacle density:

Three environments were created in Gazebo to evaluate generalization:

- **Environment A**: 24m $\times$ 28m with sparse static obstacles (Figure 3);
- **Environment B**: 20m $\times$ 20m with denser obstacle layout (Figure 4);
- **Environment C**: 14m $\times$ 14m with narrow passages (Figure 5).

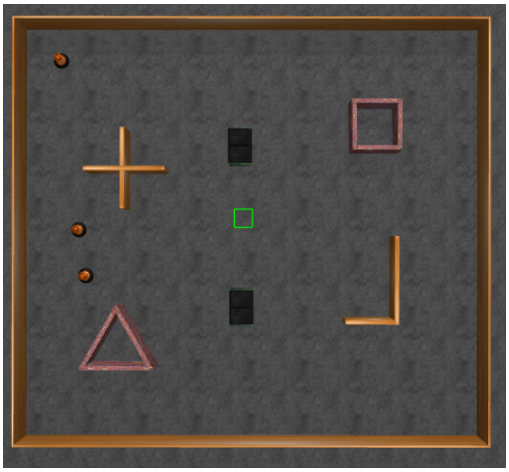


Fig. 3: Simulation (**Environment A**) (24m $\times$ 28m) with sparse obstacles.

Environment B (20m $\times$ 20m) was selected for the majority of training episodes as it offered a balance between challenge and feasibility. It ensured that trajectories were not overly constrained by narrow corridors, while still requiring obstacle negotiation. Choosing an environment too large relative to the robot led to value function convergence in local minima, whereas overly small environments resulted in excessive penalties and increased learning difficulty. The selected setup thus supports both safe exploration and learning efficiency.

D. Reward Function

The reward function, adapted from [8], promotes efficient, goal-directed, and collision-free navigation. It handles three scenarios: reaching the goal, colliding with an obstacle, or progressing normally.

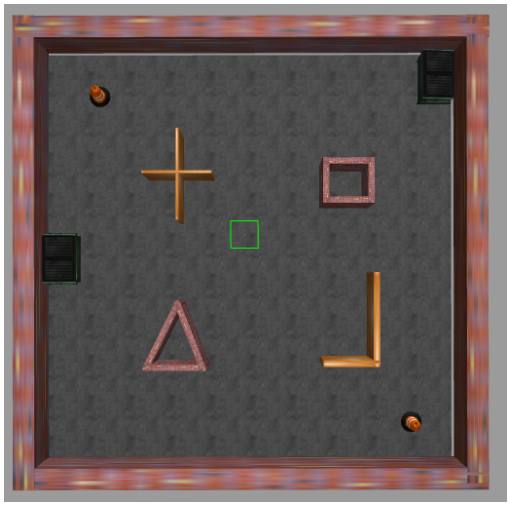


Fig. 4: Simulation (**Environment B**) (20m × 20m) with denser layout.

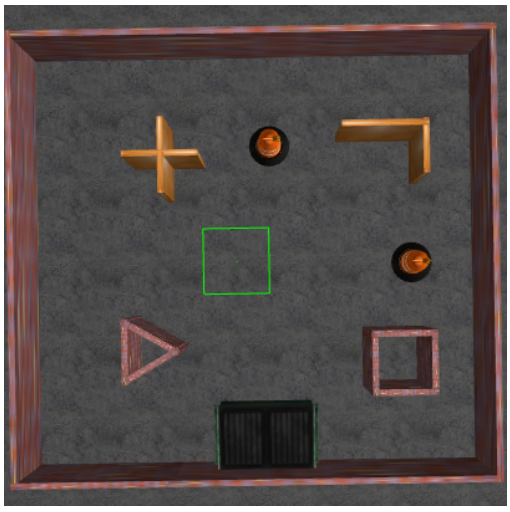


Fig. 5: Simulation (**Environment c**) (14m × 14m) with narrow passages.

For each step, the reward is computed as:

$$r(s_t, a_t) = \begin{cases} +100, & \text{if goal is reached} \\ -100, & \text{if collision occurs} \\ \frac{1}{d} + \cos(\theta) + v - |\omega|, & \text{otherwise} \end{cases} \tag{11}$$

Here, d is the distance to the goal, θ is the heading error, v the linear velocity, and ω the angular velocity. Goal and collision events are triggered when the robot is within 0.5 m of the target or 0.7 m from an obstacle, respectively.

E. Training Procedure

The training process follows the episodic structure described in [8]. Each episode begins with a randomized start and goal position within the environment. The goal positions are sampled from a Gaussian distribution centered around the robot's starting pose, with the sampling radius gradually increased across episodes. This strategy encourages progressive exploration of larger regions and helps prevent premature convergence to local minima.

An episode terminates when the robot reaches the goal, collides with an obstacle, or exceeds a predefined step limit. Transitions of the form (s_t, a_t, r_t, s_{t+1}) are stored in a replay buffer. During training, mini-batches are sampled from this buffer to update both actor and critic networks using the TD3 protocol with soft target updates.

Epochs consist of multiple episodes and are used to periodically evaluate the learning progress and revalidate the network's parameters across different environment instances.

IV. Results and Discussion

Training and testing were carried out in the three previously described simulation environments, each with distinct spatial characteristics and obstacle configurations. These environments were designed to impose varying levels of difficulty, enabling a comprehensive evaluation of the learning process. **Environment C** offered limited free space, increasing the likelihood of collisions and requiring precise control. In contrast, **Environment A** featured large open areas, presenting fewer immediate obstacles. **Environment B** included both open and constrained regions, providing a balanced setup for assessing training robustness and generalization.

All experiments were run on an Ubuntu 20.04 notebook equipped with an Intel i7 processor, NVIDIA Quadro RTX 3000 Max-Q GPU, and 32 GB of RAM.

Training in the largest **Environment A** lasted 14 hours but failed to show convergence in Q-value and loss curves. This behavior is attributed to the abundance of non-penalized trajectories, which led to the reinforcement of local minima. As illustrated in Fig. 6, both the average and maximum Q-values fluctuated without stabilizing, and the critic loss remained high throughout the training process.

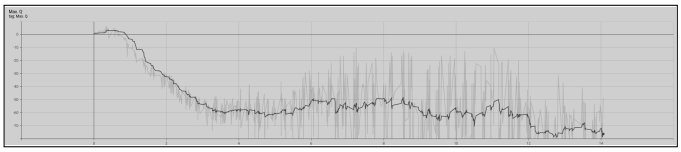
In contrast, **Environment B** showed consistent convergence of Q-values and critic loss within approximately 8 hours of training. As shown in Fig. 7, both the average and maximum Q-values stabilized over time, and the critic loss decreased steadily, indicating effective policy learning in this more balanced scenario.

Similarly, **Environment C** eventually reached convergence, albeit requiring 26 hours of training due to the higher penalty density and tighter navigation margins, as shown in Fig. 8.

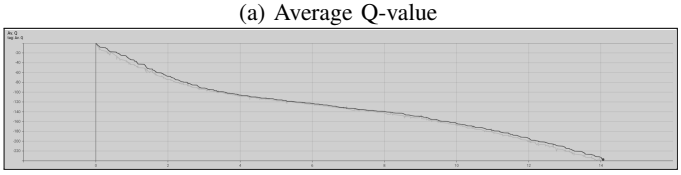
A summary of final training statistics is presented in Table I. An interesting emergent behavior was observed during training: in some cases, the robot would intentionally pause near the goal before reaching it. This behavior appears to exploit the reward function's sensitivity to proximity and heading alignment, accumulating additional reward before triggering episode termination.

A. Testing and Deployment

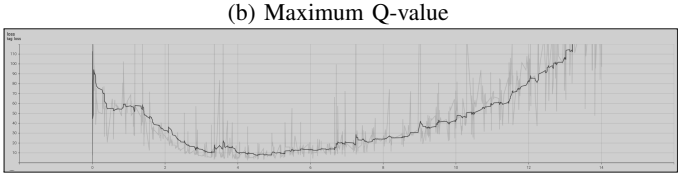
The trained model from the **Environment B** environment was tested across all three simulated environments. Despite



(a) Average Q-value

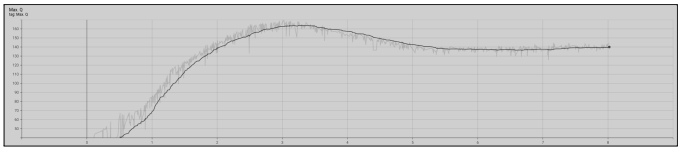


(b) Maximum Q-value

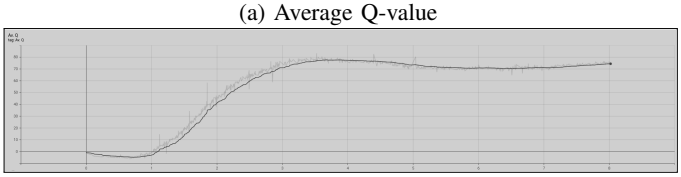


(c) Critic loss

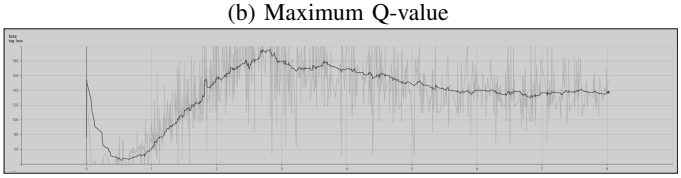
Fig. 6: Training metrics in **Environment A** (24m×28m).



(a) Average Q-value



(b) Maximum Q-value



(c) Critic loss

Fig. 7: Training metrics in **Environment B** (20m×20m).

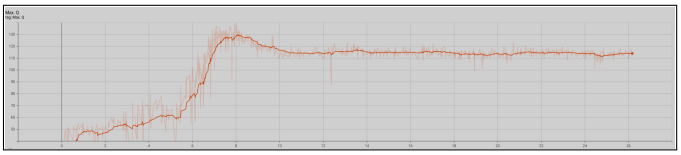
TABLE I: Training Results Summary

Environment	Time (h)	Q_{max}	Q_{avg}	Loss	Converged
A	14	-76.2	-229.8	256.4	No
B	8	140.1	73.8	142.1	Yes
C	26	114.3	32.2	99.1	Yes

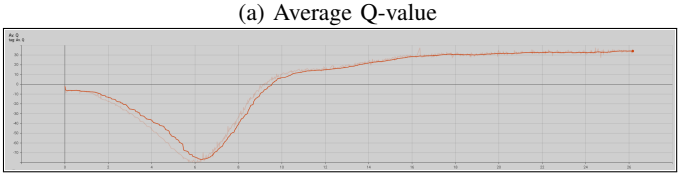
differences in layout and obstacle positioning, the robot consistently reached the defined goal without collisions, demonstrating strong generalization. As illustrated in Figures 9, 10, and 11, the paths were successfully tracked, and all obstacles were appropriately avoided throughout the trajectories.

These results[1] confirm the viability of the trained TD3

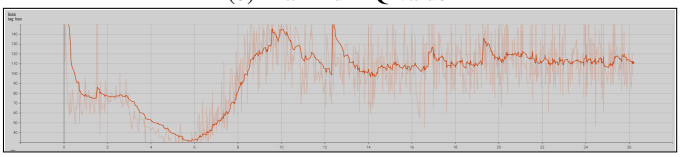
[1]These results can be better visualized through the video available at: https://youtu.be/_VqVAyuaeUc



(a) Average Q-value



(b) Maximum Q-value



(c) Critic loss

Fig. 8: Training metrics in **Environment C** (14m×14m).

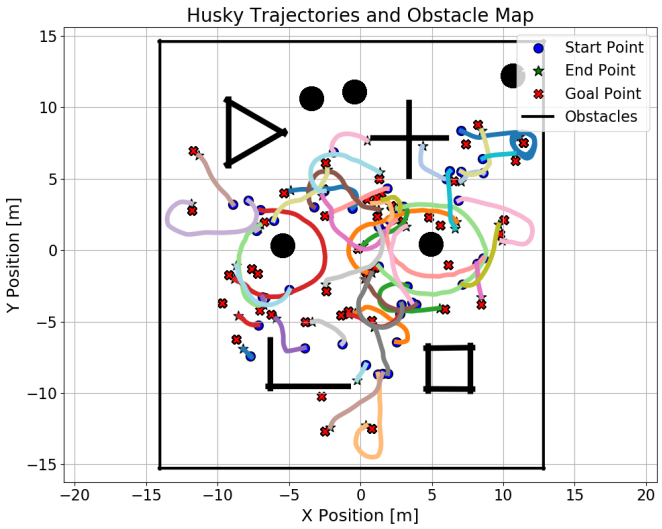


Fig. 9: Test results (**Environment A**).

agent to generalize beyond its original training distribution and adapt to spatial and geometric variations in the environment. **Environment C**, due to its limited free space, posed the most difficult scenario. Conversely, **Environment A** environment had large open areas, which reduced training pressure and led to premature convergence to suboptimal behaviors. The **Environment B** environment provided a balance between challenge and feasibility, containing both open and narrow passages.

V. Conclusion

This work evaluated the application of the TD3 algorithm for autonomous navigation of a skid-steering mobile robot in simulated environments with static obstacles. The study focused on analyzing the algorithm's performance under dif-

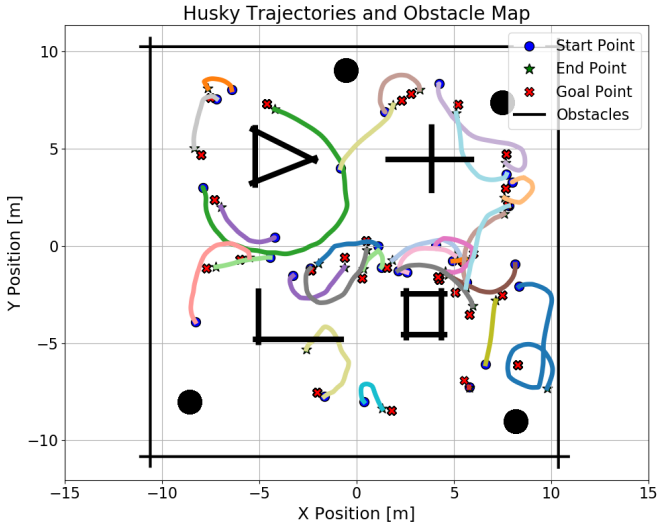


Fig. 10: Test results (**Environment B**).

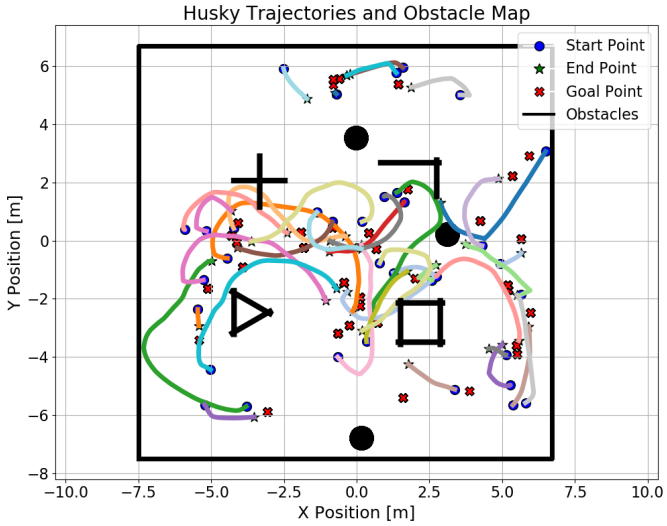


Fig. 11: Test results (**Environment C**).

ferent environmental dimensions, using only onboard sensing and relative target positioning.

Results showed that TD3 is capable of learning collision-free navigation strategies, but its performance varies significantly with the spatial configuration of the environment. These findings emphasize the additional challenges imposed by the nonlinear dynamics of skid-steering systems.

This study contributes to the understanding of how model-free DRL algorithms behave in more realistic robotic platforms, providing a reference for future comparisons and extensions.

References

[1] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, "Introduction to Autonomous Mobile Robots," 2nd ed., Cambridge, MA, USA: MIT Press, 2011.

[2] A. P. Aguiar and J. P. Hespanha, "Trajectory-Tracking and Path-Following of Underactuated Autonomous Vehicles With Parametric Modeling Uncertainty," *IEEE Trans. Autom. Control*, 2007.

[3] J. Sousa et al., "Modeling and Control of Skid-Steering Mobile Robots Using Nonlinear Control Techniques," *Journal of Intelligent & Robotic Systems, vol. 91, no. 3*, 2018.

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[5] L. Tai et al., "Virtual-to-Real Deep Reinforcement Learning: Continuous Control of Mobile Robots for Mapless Navigation," *IROS*, 2017.

[6] M. Pfeiffer et al., "Reinforced Imitation: Sample Efficient Deep Reinforcement Learning for Mapless Navigation by Leveraging Prior Demonstrations," *IEEE RA-L, vol. 3, no. 4*, 2018.

[7] 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," in *IEEE Access, vol. 10, pp. 128077-128089*, 2022

[8] R. Cimurs, I. H. Suh, and J. H. Lee, "Goal-Driven Autonomous Exploration Through Deep Reinforcement Learning," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 730–737, 2022, doi: 10.1109/LRA.2021.3133591.

[9] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, D. Kumaran, and R. Hadsell, "Learning to Navigate in Complex Environments," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2017.

[10] S.-L. Jeng and C.-C. Chiang, "End-to-End Autonomous Navigation Based on Deep Reinforcement Learning with a Survival Penalty Function," *Sensors*, vol. 23, no. 20, p. 8651, 2023.

[11] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2017, pp. 3357–3364.

[12] L. Tai, G. Paolo, and M. Liu, "Virtual-to-Real Deep Reinforcement Learning: Continuous Control of Mobile Robots for Mapless Navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2017, pp. 31–36.

[13] Raj, R., Kos, A. Intelligent mobile robot navigation in unknown and complex environment using reinforcement learning technique. Sci Rep 14, 22852 (2024). https://doi.org/10.1038/s41598-024-72857-3.

[14] H. Liu, Y. Shen, C. Zhou, Y. Zou, Z. Gao and Q. Wang, "TD3 Based Collision Free Motion Planning for Robot Navigation," 2024 6th *International Conference on Communications, Information System and Computer Engineering (CISCE)*, Guangzhou, China, 2024, pp. 247-250.

[15] H. Tai, G. Paolo, F. Manfredi, and J. Bohg, "A survey of deep reinforcement learning in robotic perception and control," *Foundations and Trends in Robotics*, vol. 9, no. 3–4, pp. 211–340, 2021.

[16] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. Int. Conf. on Machine Learning (ICML)*, 2018, pp. 1587–1596.

[17] H. Dai, P. Chen, H. Yang, "Driving Torque Distribution Strategy of Skid-Steering Vehicles with Knowledge-Assisted Reinforcement Learning," *Applied Sciences*, vol. 12, no. 10, p. 5171, 2022, doi: 10.3390/app12105171.

[18] Z. Xie and P. Dames, "DRL-VO: Learning to Navigate Through Crowded Dynamic Scenes Using Velocity Obstacles," in IEEE Transactions on Robotics, vol. 39, no. 4, pp. 2700-2719, Aug. 2023, doi: 10.1109/TRO.2023.3257549.

[19] A. Joglekar, S. Sathe, N. Misurati, S. Srinivasan, M. J. Schmid, and V. Krovi, "Deep reinforcement learning based adaptation of pure-pursuit path-tracking control for skid-steered vehicles," *IFAC-PapersOnLine*, vol. 55, no. 37, pp. 400–407, 2022.

[20] A. Mandow, J. L. Martínez, J. Morales, S. Pedraza, and A. García-Cerezo, "Experimental kinematics for wheeled skid-steer mobile robots," *IEEE Transactions on Robotics*, vol. 23, no. 3, pp. 529–534, Jun. 2007